Simplified

SARVA EDUCATION

e- Book

HTML, DHTML & JAVASCRIPT

SARVA EDUCATION ^{SM} *- An I.T & Skill Advancement Training Programme, Initiated by* ***SITED***®***-India***

An ISO 9001:2015 Certified Organization

UNITS AT A GLANCE

Unit-I

HYPERTEXT MARKUP LANGUAGE (HTML)

INTRODUCTION
Hyper Text Markup Language is the most important aspect in web site development. We write web pages that contain codes written in this particular language. As name implies this is a language like other high level languages, though much easier than those. We shall discuss broadly about different features of html in this chapter in order to get acquainted with writing simple web pages.

CONCEPT OF HYPERTEXT
Web pages are written in a relatively easy computer language. It is known as HTML, short for Hypertext Markup Language. The hypertext concept stems from the Apple Macintosh world. It means mutually connecting text, graphics and other data in a computer file. Simply pointing your mouse to a link and clicking on it, will bring up a different part of the document on your screen as you can see one specimen in figure. This method is very popular for documenting software. You can think HTML as the help files coming with almost all software's.

VERSIONS OF HTML
After the emergence of HTML, several versions came with some upgrading as follows:

- **HTML 2.0 -** It was developed under the supervision of IETF (Internet Engineering Task Force). It came into practice in late 1994.
- **HTML 3.0 -** It was a richer version of HTML. Despite never receiving consensus in standard discussion, it was continuously changing with the new features.
- **HTML 3.2 -** In 1996, it came into existence by the efforts of World Wide Web.
- **HTML 4.0 -** It extends with mechanisms for style sheet scripting, frames, richer tables, forms etc. It is the most current version of HTML.

HTML EDITORS
HTML documents are plain-text (also known as ASCII) files that can be created using any text editor (e.g., Emacs or vi on UNIX machines; SimpleText on a Macintosh; Notepad on a Windows machine). You can also use word-processing software if you remember to save your document as "text only with line breaks".
Some WYSIWYG editors are also available (e.g., Claris Home Page or Adobe PageMill, both for Windows and Macintosh). You may wish to try one of them after you learn some of the basics of HTML tagging. WYSIWYG is an acronym for "What You See Is What You Get"; it means that you design your HTML document visually, as if you were using a word processor, instead of writing the markup tags in a plain-text file and imagining what the resulting page will look like. It is useful to know enough HTML to code a document before you determine the usefulness of a WYSIWYG editor, in case you want to add HTML features that your editor doesn't support.

ELEMENTS OF HTML
An HTML element is a fundamental component of the structure of a text document. Some examples of elements are heads, tables, paragraphs, and lists. Elements can contain plain text, other elements, or both.

TAGS
A web page consists of various types of tags. To denote the various elements in an HTML document, we use tags. HTML tags consist of a left angle bracket **(<)**, a tag name, and a right angle bracket **(>)**. Tags are usually paired (e.g., **<Tag>** and **</Tag>)** to start and end the tag instruction. The end tag looks just like the start tag except a slash **(/)** precedes the text within the brackets?

ATTRIBUTES
Some elements may include an attribute, which is additional information that is included inside the start tag. ***For example,*** you can specify the alignment of text (right, left or center) by including the appropriate attribute with the image source HTML code.

- **HTML is not case sensitive. <title> is equivalent to <TITLE> or <TiTlE>.**
- **Not all tags are supported by all World Wide Web browsers. If a browser does not support a tag, it will simply ignore it. Any text placed between a pair of unknown tags will still be displayed, however.**

TO WRITE A HTML DOCUMENT
To write a web document, follow the instructions as follows:
- Load a text editor like Notepad. To open Notepad click **Start** --- **Programs** --- **Accessories** --- **NotePad.**
- Write codes and save it with a HTML file. HTML files are the files having extensions **.htm** or. **html.**
- View it in a browser of your choice e.g., Internet Explorer or Netscape Navigator.

MINIMUM REQUIRED TAGS IN AN HTML DOCUMENT

Every HTML document should contain certain standard HTML tags. They are as follows-

• HTML tag • Head tag • Title tag • Body tag

HTML Tag

The HTML element tells your browser that the file contains HTML-coded information. The file extension .html also indicates this HTML document and must be used. It comes in a pair: ***<HTML> and </HTML>***

Syntax:

```
<HTML>
…………
…………
</HTML>
```

HEAD Tag

The Head element identifies the first part of your HTML-coded document that contains the title. It contains title tag. It also conies with opening and closing tags: ***<HEAD>and </HEAD>***

Syntax:

```
<HTML>
<HEAD>
………..
………..
</HEAD>
</HTML>
```

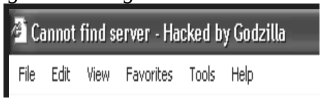
TITLE Tag

The Title element contains your document title and identifies its content in a global context. The title is typically displayed in the title bar at the top of the browser window, but not inside the window. The title is also what is displayed on someone's hot list or bookmark list, so choose something descriptive, unique, and relatively short. A title is also used to identify your page for search engines. Generally you should keep your titles to 64 characters or fewer.

Syntax:

```
<HTML>
<Head>
<TITLE>
DESCRIBE YOUR TITLE
</TITLE>
</HEAD>
</HTML>
```

After implementing the Title tag as above results like this-


Cannot find server - Hacked by Godzilla
File Edit View Favorites Tools Help

If you leave blank between the title tags, some browsers display either UNTITLED or URL.

BODY Tag

The second—and largest—part of your HTML document is the body, which contains the content of your document (displayed within the text area of your browser window). The tags explained below are used within the body of your HTML document.

Syntax:

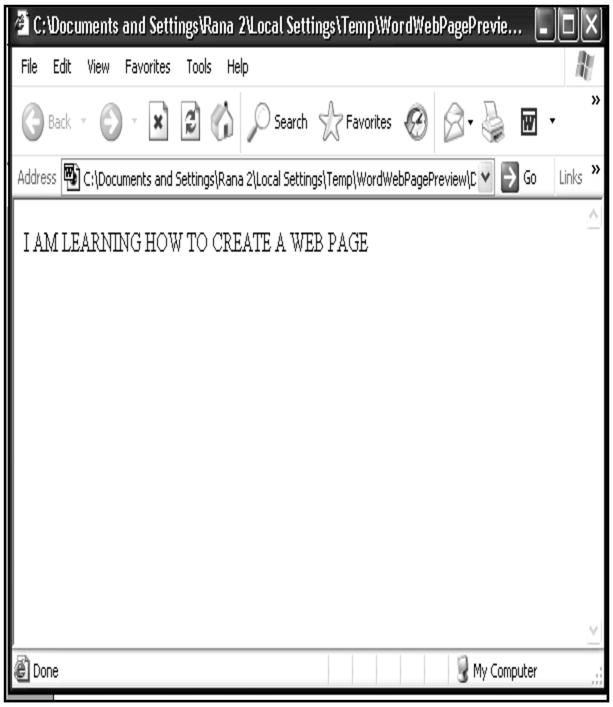
```
<HTML>
<HEAD>
<TITLE>
My First Web Page
<TITLE>
</HEAD>
<BODY>
………..
……….
</BODY>
</HTML>
```

A SAMPLE HTML DOCUMENT:

Each HTML document contains some certain standard HTML tags. It has been illustrated in example given below that shows what a minimal document contains.

```
<HTML>
<HEAD>
<TITLE>
A Basic HTML Page
</TITLE>
</HEAD>
<BODY>
I AM LEARNING HOW TO CREATE A WEB PAGE
</BODY>
</HTML>
```

For implementing the above mentioned example, **save** it with **.htm** or **.html** extension and see the result on the browser. The effects should match with one in figure given below.


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePrevie...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\D
Go Links
Done
My Computer

I AM LEARNING HOW TO CREATE A WEB PAGE

HEADINGS

Headings are used to arrange the contents in a systematic way. HTML has six levels of headings, numbered 1 through 6, with 1 being the largest. Headings are typically displayed in larger and/or bolder fonts than normal body text. The first heading in each document should be tagged <H1>.

The syntax of the heading element is:

<Hn>Heading/Sub-Heading</Hn>

Where **n** is a number between 1 and 6 specifying the level of the headings. You can see the effects of different headings in example given below-

```
                    <HTML>
                    <HEAD>
                    <TITLE>
A Web Page Showing Different Level of Headings.
                    </TITLE>
                    </HEAD>
                    <BODY>
                    <H1> Sunday </H1>
                    <H2> Monday </H2>
                    <H3> Tuesday </H3>
                    <H4> Wednesday </H4>
                    <H5> Thursday >/H5>
                    <H6> Friday </H6>
                    <H7> Saturday </H7>
                    </BODY>
                    </HTML>
```

The result should appear like one in figure given below-

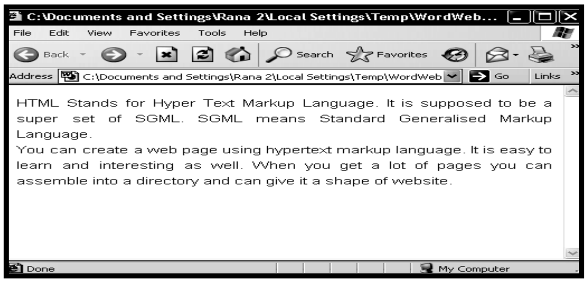
```
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
```

PARAGRAPHS

Paragraph may consist of a single sentence or more than one sentence including punctuation marks and blank space. Paragraph is represented by **<P>…</P> tags. End tag </P>** is not essential to add as the browsers assume the start of paragraph when **<P>** tag is given**.** See example 6.3 to understand the tag well.

```
                    <HTML>
                    <HEAD>
                    <TITLE>
                    A Page Showing Two Paragraphs
                    </TITLE>
                    </HEAD>
                    <BODY>
                    <P>
HTML Stands for Hyper Text Markup Language. It is supposed to be a super set of
SGML. SGML means Standard Generalized Markup Language.
                    <P>
You can create a web page using hypertext markup language. It is easy to learn and
interesting as well. When you get a lot of pages you can assemble into a directory
and can give it a shape of website.
                     </P>
                    </BODY>
                    </HTML>
```

After implementing the example, the browser displays the result like one in figure below-



HTML Stands for Hyper Text Markup Language. It is supposed to be a super set of SGML. SGML means Standard Generalised Markup Language.
You can create a web page using hypertext markup language. It is easy to learn and interesting as well. When you get a lot of pages you can assemble into a directory and can give it a shape of website.

C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links
Done
My Computer

LINE BREAK

This tag is used to change lines with no extra blank spaces. It can be used for short lines of text such as postal addresses. This tag is represented by <BR>. See example -

```
        <HTML>
        <HEAD>
        <TITLE>
        A PAGE WITH BR TAG.
        </TITLE>
        </HEAD>
        <BODY>
        <PR>
        National Informatics Centre <BR>
        Chanakya Puri <BR>
        New Delhi <BR>
        </BODY>
        </HTML>
```

The example results on the browser like this-

```
National Informatics Centre
Chanakya Puri
New Delhi
```

LISTS

HTML supports unnumbered, numbered, and definition lists. You can also nest lists.

NUMBERED LIST

Numbered list is also called ordered list. It is shortened as OL from which the tag name! derives. It is identical to an unnumbered list, except it uses <OL> instead of <UL>. The items are tagged using the same <LI> tag. See example-

```
        <OL>
        <LI> Computer
        <LI> Modem
        <LI> Internet Connection
        <LI> Telephone Line
        </OL>
```

After running the example 6.5, the browser displays result like this-

1. Computer
2. Modem
3. Internet Connection
4. Telephone Line

UNNUMBERED LISTS

Unnumbered list can be referred to as unordered list. To make an unnumbered or bulleted list, do the followings-

1. Start with an opening list <UL> (for unnumbered list) tag

2. Enter the <LI> (List Item) tag followed by the individual item; no closing </LI> tag is needed.

3. End the entire list with a closing list </UL> tag

See example to understand how it is written.

```
<UL>
<LI> Computer
<LI> Modem
<LI> Internet Connection
<LI> Telephone Line
</UL>
```

Example results on the browser like this-

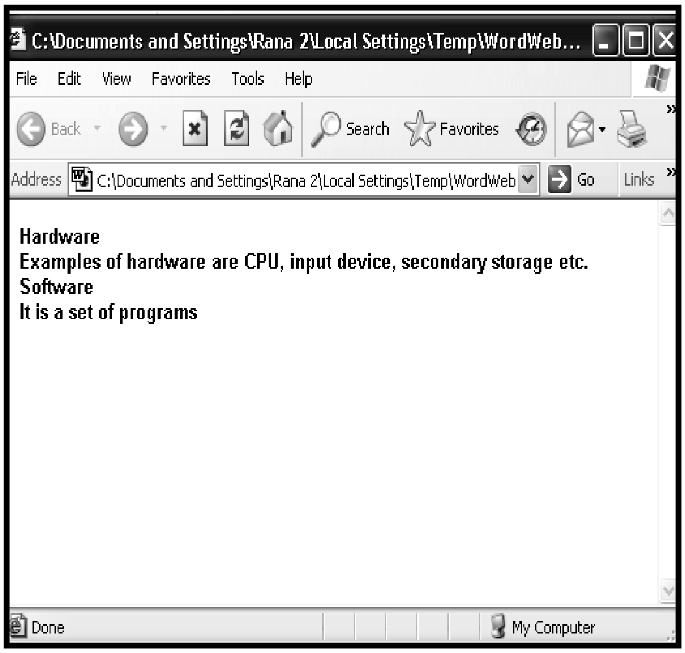
- Computer
- Modem
- Internet Connection.
- Telephone Line

DEFINITION LISTS

A definition list (coded as **<DL>)** usually consists of alternating a definition term (coded as **<DT>)** and a definition description (coded as **<DD>)**. Web browsers generally format the definition on a new line and indent it.

```
<HTML>
<HEAD>
<TITLE>
Definition List
</TITLE>
</HEAD>
<BODY>
<DL>
<DT> Hardware
<DD>Examples of hardware are CPU, input device, secondary storage etc.
<DT> Software
<DD> It is a set of programs
</DL>
</BODY>
</HTML>
```

Codes written in example will affect like one in figure on browser:-


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links
Done My Computer

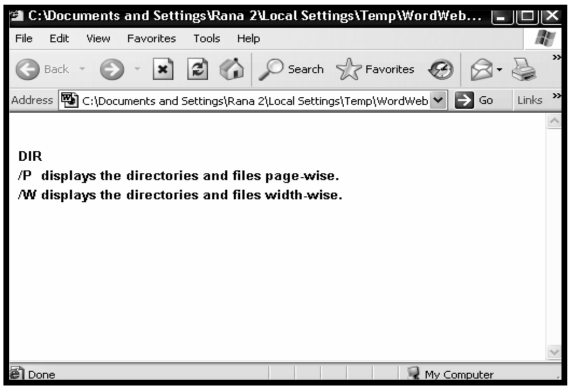
Hardware
Examples of hardware are CPU, input device, secondary storage etc.
Software
It is a set of programs

COMPACT ATTRIBUTE

The COMPACT attribute can be used routinely in case your definition terms are very short. If, for example, you are showing some computer options, the options may fit on the same line as the start of the definition.

```
<P> DIR </P>
<DLCOMPACT>
<DT> IP
<DD> displays the directories and files page-wise.
<DT> /w
<DD> displays the directories and files width-wise.
</DL>
```

See its output in figure below-


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links
Done My Computer

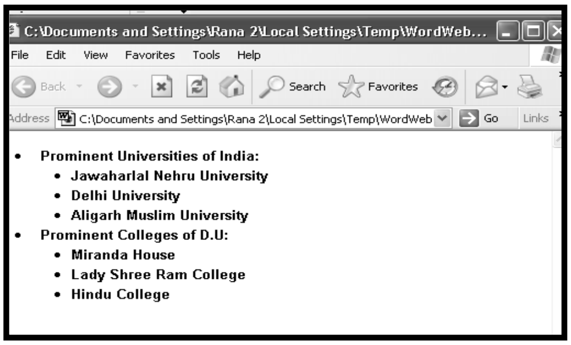
DIR
/P displays the directories and files page-wise.
/W displays the directories and files width-wise.

NESTED LIST

Lists can be nested. You can also have a number of paragraphs, each containing a nested list, in a single list item.

```
<UL>
<LI> Prominent Universities of India:
<UL>
<LI> Jawaharlal Nehru University
<LI> Delhi University
<LI> Aligarh Muslim University
</UL>
<LI> Prominent Colleges of D.U:
<UL>
<LI> Miranda House
<LI> Lady Shree Ram College
<LI> Hindu College
</UL>
</UL>
```

See output in figure below-


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links

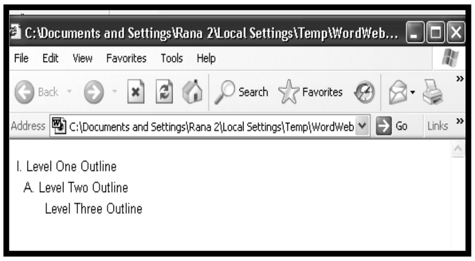
- Prominent Universities of India:
  - Jawaharlal Nehru University
  - Delhi University
  - Aligarh Muslim University
- Prominent Colleges of D.U:
  - Miranda House
  - Lady Shree Ram College
  - Hindu College

TYPE ATTRIBUTE

Type attribute is used with <OL> tag. Using this, you can manipulate the way ordered list displays the effects.

<HTML>
<HEAD>
<TITLE>
My first web page
</TITLE>
</HEAD>
<BODY>
<OLTYPE = "I">
<LI> Level One Outline
<OLTYPE = "A">
<LI> Level Two Outline
<OLTYPE = "1">
</LI> Level Three Outline
</OL>
</OL>
</OL>
</BODY>
</HTML>

See output in figure-


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links

I. Level One Outline
A. Level Two Outline
Level Three Outline

PREFORMATTED TEXT

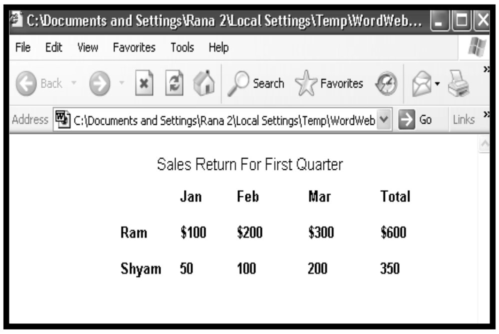
The PRE (Preformatted text) tag is used to display a block of "preformatted" text in a monospace, fixed-pitch font. You use the PRE tag to display a block of text "as it is", Including all spaces and hard returns. One of the primary uses of the PRE tag is to display text in a tabular or columnar format in which you want to make sure that the columns remain properly aligned.

<HTML>
<HEAD>
<TITLE>
My First Web Page
<TITLE>
</HEAD>
<BODY>
<PRE>
Sales Return for First Quarter

Jan	Feb	Mar	Total
$100	$200	$300	$600
50	100	200	350

Ram
Shyam
Gopal
</PRE>
</BODY>
</HTML>

See output in figure:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links

```
Sales Return For First Quarter
        Jan    Feb    Mar    Total
Ram     $100   $200   $300   $600
Shyam   50     100    200    350
```

COMMENT TAG

The comment tag is a stand-alone tag that enables you to include messages for your own or anyone else's future references. A comment always begins with a **<!-** and ends with a **->**.

Example: **<! - Type your comments here ->**

EXTENDED QUOTATION

Use the **<BLOCKQUOTE>** tag to include lengthy quotations in a separate block on the screen. Most browsers generally change the margins for the quotation to separate it from surrounding text.

<P> Note: </P>
<BLOCKQUOTE>
<P>
This page is meant for no commercial purpose. Therefore, no claims whatsoever can be entertained regarding any information published on it.
</P>
Authority
</BLOCKQUOTE>

*After implementing the above mentioned example, the result will be like this-****Note:***

This page is meant for no commercial purpose. Therefore, no claims whatsoever can be entertained regarding any information published on it.
Authority

USING LINKS

The chief power of HTML comes from its ability to link text and/or an image to another document or section of a document. A browser highlights the identified text or image with color and/or underlines to indicate that it is a hypertext link (often shortened to hyperlink or just link). HTML's single hypertext-related tag is **<A>**, which stands for anchor.
Hypertext links can be used to move from one page to the other or to a specific location within the same document or different one. For instance, you can use links to-

- **display an image**
- **download a program**
- **send e-mail**
- **use a database**
- **execute a script**

Hyperlinks are mainly of three kinds –

LINKS TO DIFFERENT WEB PAGES

This type of link is frequently used in web pages. It takes you from one page to the other one.

Links to Specific Sections of *Current* Web Pages

This type of link helps navigation in the current web page. It takes you from one section to another section of the page. Several pages are so big that you can not see it at once and browsing may take time. In that situation, it is found very helpful. Clicking the concerned link available in the beginning directly takes you to the specified section.

Links to Specific Sections of *Different* Web Pages

This is very much identical to the above link except it takes you to a specific section in a different web page directly.

U.R.L.

The *World Wide Web* uses Uniform Resource Locators (URLs) to specify the location of files on other servers. A URL includes the type of resource being accessed (e.g., Web, gopher, FTP), the address of the server, and the location of the file. ***The syntax is:***

scheme://host. domain [:port]/path/ filename

where scheme is one of-

file	-	**a file on your local system**
ftp	-	**a file on an anonymous FTP server**
http	-	**a file on a World Wide Web server**
gopher	-	**a file on a Gopher server**
WAIS	-	**a file on a WAIS server**
news	-	**a Usenet newsgroup**
telnet	-	**a connection to a Telnet-based service**

The port number can generally be omitted. (That means unless someone tells you otherwise, leave it out.)

To include an anchor in your document:

1. Start the anchor with <A (include a space after the A)
2. Specify the document you're linking to by entering the parameter HREF="filename" followed by a closing right angle bracket (>)
3. Enter the text that will serve as the hypertext link in the current document
4. Enter the ending anchor tag: </A> (no space is needed before the end anchor tag) Here is a sample hypertext reference in a file called introduction.html:

<AHREF="introduction.html">Introduction</A>

This entry makes the word Introduction the hyperlink to the document Introduction.html, which is in the same directory as the first document.
You can link to documents in other directories by specifying the relative path from the current document to the linked document. For example, a link to a file introduction.html located in the subdirectory Sarva would be:

<AHREF="Sarva/introduction.html">Introduction</A>

These are called relative links because you are specifying the path to the linked file relative to the location of the current file.

LINKS TO SPECIFIC SECTIONS

Anchors can also be used to move a reader to a particular section in a document (either the same or a different document) rather than to the top, which is the default. This type of an anchor is commonly called a named anchor because to create the links, you insert HTML names within the document. You can also link to a specific section in another document. That information is presented first because understanding that helps you understand linking within one document.

LINKS BETWEEN SECTIONS OF DIFFERENT DOCUMENTS

Suppose you want to set a link from Sarva (sarva.html) to a specific section in another document (books.html).
Enter the HTML coding for a link to a named anchor**:**

sarva.html:
<a href="sarva.html#SITED">Internet & E-Commerce</a>.

Think of the characters after the hash (#) mark as a tab within the sarvaindia.html file. This tab tells your browser what should be displayed at the top of the window when the link is activated. In other words, the first line in your browser window should be the Internet & E-Commerce heading. Next, create the named anchor (in this example "SITED") in introduction.html:

<H2><A NAME="SITED">Internet & E-Commerce</Ax/H2>

With both of these elements in place, you can bring a reader directly to the Internet reference in introduction.html.

LINKS TO SPECIFIC SECTIONS WITHIN THE CURRENT DOCUMENT

The technique of Links to specific sections within different a document is very much identical to the technique of links to specific sections within the current document except the filename is omitted. For example, to link to the IEC anchor from within Sarva, enter:

<A HREF="#SITED">Internet & E-Commerce</a>

After this include the <A NAME=> tag at the place in your document where you want the link to jump to (<A NAME="IEC">Internet & E-Commerce</A>). Named anchors are particularly useful when you think readers will print an entire or when you have a lot of short information you want to place online in one file.

MAILTO

You can make it easy for a reader to send electronic mail to a specific person.

<A HREF="mailto:emailinfo@ host">Name<la>

For example, enter:

<A HREF="mailto:mansoor37 76@ rediffmail.com">
Contact Me</a>

To create a mail window that is already configured to open a mail window for the editor.

INLINE IMAGES

Most Web browsers can display inline images (that is, images next to text) that are in X Bitmap (XBM), GIF, or JPEG format. Other image formats are also being incorporated into Web browsers [e.g., the Portable Network Graphic (PNG) format]. Each image takes additional time to download and slows down the initial display of a document. Carefully select your images and the number of images in a document.

To include an inline image, enter:

<IMG SRC=Image Name>

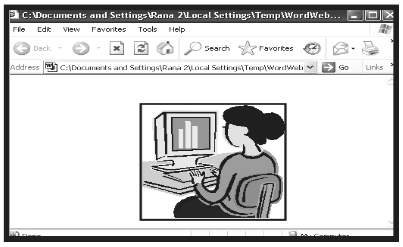
Where *ImageName* is the URL of the image file.

Specifying Size of Inline Images

You should include two other attributes on **<IMG>** tags to tell your browser the size of the images it is downloading with the text. The **HEIGHT** and **WIDTH** attributes let your browser set aside the appropriate space (in pixels) for the images as it downloads the rest of the file.
For example, to include an image named computer in a file along with the image's dimensions, enter**:**

<IMG SRC=computer. if HEIGHT=100 WIDTH=65>

After running the above command, you can see a big picture shrunk to the size (very small) as specified. See figure given below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links

Links to Images

We can link images with the text in a hypertext link. This link appears in a blue border. For example:

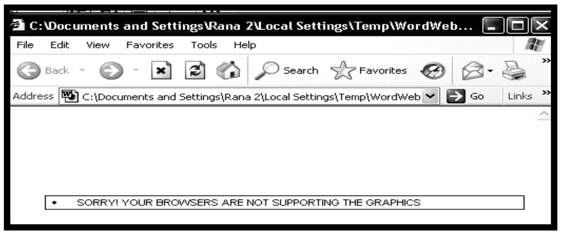
<A HREF="SUB.HTM"> <IMG SRC="FLOWERS.GIF></A>

Using ALT Attribute

Sometimes browsers used by your users may not be supporting the graphics available on your web page. In this situation, you must pass an alternative message to the users that makes the users understand what really the matter is. For example, see the following code

<IMG SRC="COMPUTER.GIF" ALT="SORRY! YOUR BROWSERS ARE NOT SUPPORTING THE GRAPHICS">

In case you confront the same situations as mentioned above, you should see the result in figure below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links

SORRY! YOUR BROWSERS ARE NOT SUPPORTING THE GRAPHICS

HORIZONTAL RULE

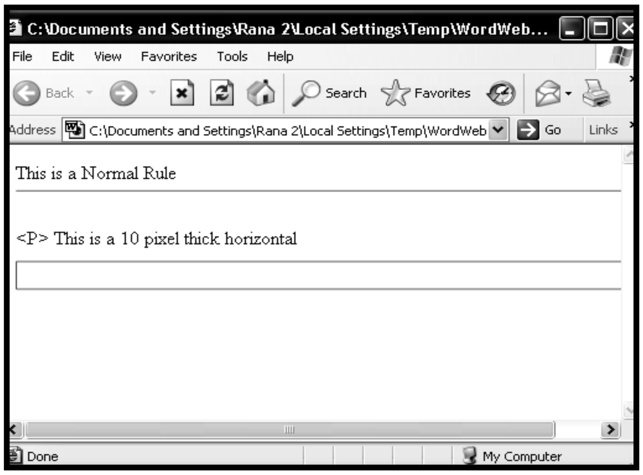
The HR (Horizontal Ruler) tag is a stand-alone, or empty, document element that allows you to add horizontal rules to your Web pages.

CHANGE THE HEIGHT OF A HORIZONTAL RULE

To change the height of a horizontal rule, the SIZE attribute value in the HR tag may be used. The value you set is the rule's height, or thickness, in pixels. The following example is used for creating a horizontal rule that is 10 pixels thick. See example:

<P> This is a Normal Rule
<HR>
<P> This is a 10 pixel thick horizontal
<HR SIZE="10">

The result will be shown as one in figure below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links
Done
My Computer

This is a Normal Rule

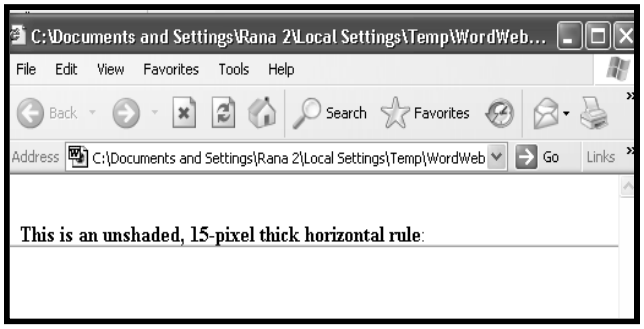
<P> This is a 10 pixel thick horizontal

REMOVING SHADES FROM THE RULE

The default setting for a rule is "shaded". To set an "unshaded" horizontal rule, add the NOSHADE attribute to the HR tag.

<P>This is an unshaded, 15-pixel thick horizontal rule:
<HR SIZE="15" NOSHADE>

The produced result may be seen in figure below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links

This is an unshaded, 15-pixel thick horizontal rule:

CHARACTER FORMATTING

We use some specific formatting attributes to highlight words, phrases or entire sentences in a paragraph or in a page. Some of them are as follows-

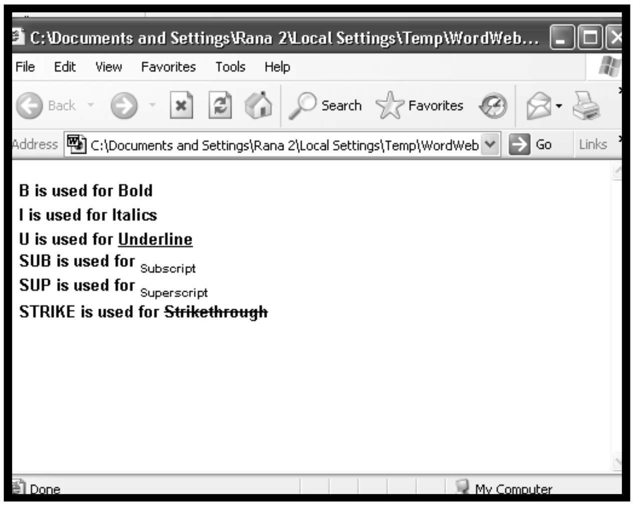
- **<B>…<B>** - This tag is used to set the text bold. For example **Computer**
- **<U>…</U>** - This tag is used to set the text underlined. For example Computer
- **<I>…<I>** - This tag is used to set the text italics. For example Computer
- **<SUB>....</SUB>** - This tag is used to set the text in a subscript form. For example in O_{2}, 2 is in subscript form
- **<sup>....</sup>** - This tag is used to set the text in a superscript form. For example in a^{2},2 is in superscript form
- **<STRIKE>...</STRIKE>** - This tag is used to set the text in a Strikethrough form.

For example: *Computer*

The usage of different formatting in example given below:

```
        <HTML>
        <HEAD>
        <TITLE> Character Formatting </TITLE>
        </HEAD>
        <BODY>
<P> B is used for <B> Bold </B>
<P> I is used for <I> Italics </I>
<P> U is used for <U> Underline </U>
<P> SUB is used for <SUB> Subscript </SUB>
<P> SUP is used for <SUP> Superscript </SUP>
<P> STRIKE is used for <STRIKE> Strikethrough</STRIKE>
        </BODY>
        </HTML>
```

The result will be shown as one in figure below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links
Done
My Computer

B is used for **Bold**
I is used for *Italics*
U is used for Underline
SUB is used for Subscript
SUP is used for Superscript
STRIKE is used for ~~Strikethrough~~

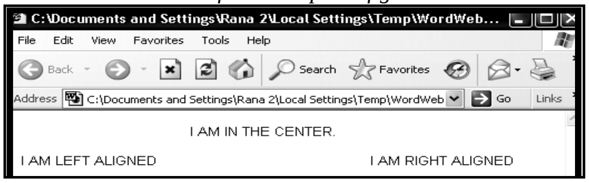
Paragraph, headings or other texts can be aligned using ALIGN attribute. They can be aligned in three ways-

• Center • Right • Left

The syntax for this is as follows-
ALIGN="value"
Where value can be replaced with left, right or center.

```
<HTML>
<HEAD>
<TITLE>
Different Alignments
</TITLE>
</HEAD>
<BODY>
<P ALIGN="CENTER"> I AM IN THE CENTER.
<P ALIGN="LEFT"> I AM LEFT ALIGNED.
<PALIGN="RIGHT"> I AM RIGHT ALIGNED.
</BODY>
</HTML>
```

See the result of the example in figure below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links

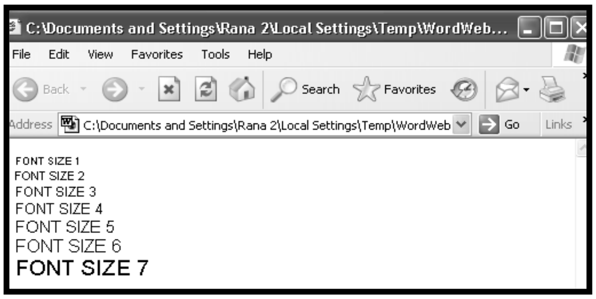
I AM IN THE CENTER.
I AM LEFT ALIGNED I AM RIGHT ALIGNED

CHANGING FONT SIZE AND COLOURS

You can set or change the size and colors of the font as desired. It can be done using **<FONT>** tag.
To specify size of the fonts, Size attribute is used with Font tag. Normally fonts are seven in size. It can be given a number 1 being the smallest through 7 being the largest. By default the font size is 3.

```
<HTML>
<HEAD>
<TITLE>
My web page
</TITLE>
</HEAD>
<BODY>
<FONT SIZE = "1 "> FONT SIZE 1 </FONT> <BR>
<FONT SIZE = "2"> FONT SIZE 2 </FONT> <BR>
<FONT SIZE = "3"> FONT SIZE 3 </FONT> <BR>
<FONT SIZE = "4"> FONT SIZE 4 </FONT> <BR>
 <FONT SIZE = "5"> FONT SIZE 5 </FONT> <BR>
<FONT SIZE = "6"> FONT SIZE 6 </FONT> <BR>
<FONT SIZE = "7"> FONT SIZE 7 </FONT> <BR>
 </BODY>
 </HTML>
```

See the result in figure below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWeb Go Links

FONT SIZE 1
FONT SIZE 2
FONT SIZE 3
FONT SIZE 4
FONT SIZE 5
FONT SIZE 6
FONT SIZE 7

SPECIFYING FONT COLOURS

To change font colour, we use COLOR attribute with the FONT tag. We can set sixteen different colors to make fonts look attractive. They are:

(i) Black (ii) White (iii) Aqua (iv) Blue (v) Fuchsia (vi) Grey (vii) Green (viii) Lime (ix) Maroon (x) Navy (xi) Olive (xii) Purple (xiii) Red (xiv) Silver (xv) Yellow (xvi) Steal

For Example:

```
<P><FONTSIZE = 7>
<FONT COLOR = "AQUA">
This is aqua.
</FONT>
```

When you will run the above mentioned example, you will see the text in aqua colour.

SETTING THE COLOUR OF BODY BACKGROUND, TEXT AND LINK

You change the color of text, links, visited links, and active links (links that are currently being clicked on) using further attributes of the <BODY> tag.

Syntax:

<BODY BGCOLOR="colorname/code"TEXT="colorname/code" LINK="colorname/code">

For example:

<BODY BGCOLOR="#000000" TEXT="#FFFFFF" LINK="#9690CC">

This creates a window with a black background **(BGCOLOR),** white text **(TEXT),** and silvery hyperlinks **(LINK).**

The six-digit number and letter combinations represent colors by giving their RGB (red, green, blue) value. The six digits are actually three two-digit numbers in sequence, representing the amount of red, green, or blue as a hexadecimal value in the range 00-FF. **For example: 000000** is **black** (no color at all), **FFOOOO** is **bright red**, **OOOOFF** is **bright blue**, and **FFFFFF** is **white** (fully saturated with all three colors).

You can use the color name also instead of codes that are a bit difficult like:

<BODYBGCOLOR="BLACK"TEXT="WHITE"LINK="SILVER">BACKGROUNDGRAPHICS

BACKGROUND GRAPHICS

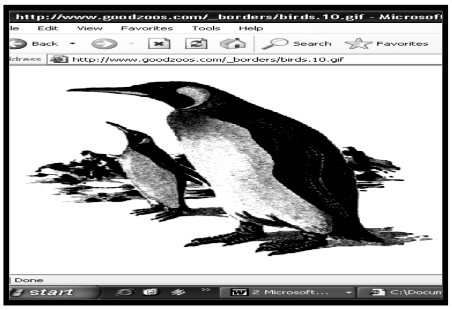
Newer versions of Web browsers can load an image and use it as a background when
displaying a page. Some people like background images and some don't. In general, if you want to include a background, make sure your text can be read easily when displayed on top of the image. Background images can be a texture (linen finished paper, for example) or an image of an object (a logo possibly). You create the background image as you do any image. The tag to include a background image is included in the ***<BODY>*** statement as an attribute:

Syntax:

<BODY BACKGROUND="filename">

For example:

```
<HTML>
<HEAD>
<TITLE>
MY BIRD
</TITLE>
</HEAD>
<BODY background=format.gif>
</BODY>
</HTML>
```


http://www.goodzoos.com/_borders/birds.10.gif - Microsoft
Edit View Favorites Tools Help
Back Search Favorites
http://www.goodzoos.com/_borders/birds.10.gif
Done
start

(See the result in figure)

Unit-II

DYNAMIC HYPERTEXT MARKUP LANGUAGE (DHTML)

INTRODUCTION

In the last unit we had discussed some basics about HTML including different tags and attributes. This Unit discusses mainly table, form and frame that are indispensable in building web pages.

TABLE

Table is used to arrange data especially when it is a combination of text and number. It is based on rows and columns and each column and row has a heading. ***For instance***, Sales-Report of a company or a table presenting information like post, address, phone no. etc.

TABLE TAG

Tag is usually used to instruct the browser what it should do. Like other tags, Table tag is written.
The syntax is as follows:

```
<TABLE>
........... Contents of table........
</TABLE>
```

The <TABLE> tag begins the table, you place what you want inside, and end the table with the </TABLE> tag. To begin adding contents to your table, you will need the <TD> tag. The TD stands for table data, which is what you will place after this tag. You end a table data section with the </TD> tag. Example given below illustrates HTML codes for a basic table with just one cell.

```
<HTML>
<HEAD>
<TITLE>
<TABLE>
<TD>
This is one-cell table.
</TD>
</TABLE>
```

Example results like this:

This is one cell table

The above example does not show clearly whether it is a table or not. To make it look like a table we add border.

ADDING BORDER TO A TABLE

To get the border, we just add the border command to the **<TABLE>** tag, like this-

```
<TABLE BORDER = "number">
 See example to know how it works -
<TABLE BORDER = "2">
<TD>
This is one-cell table.
</TD>
</TABLE>
```

The result will be like this-

This is one-cell table.

The previous result clearly shows that a border has been added around a table. You can set the border width as desired by increasing or decreasing the number like –

<TABLE BORDER = "0"> adds no border.
<TABLE BORDER = "1"> adds border.

The result something like this

This is one-cell table.

ADDING CELLS TO A TABLE

You can add cells as desired to your table. Just add <TD> as many as the number of cells you want in the table. See example.

```
<TABLE BORDER = "2">
<TD>
This is first cell.
</TD>
<TD>
This is the second cell.
</TABLE>
```

Browser will display the output like this –

This is first cell.	This is the second cell.

ADDING ROWS TO A TABLE

Like adding cells to a table, you can add rows using <TR> tag. TR stands for Table Row. Example shows addition of one row to a table.

```
<TABLE BORDER = "2">
<TD>
This is the first cell.
</TD>
<TD>
This is the second cell.
</TD>
<TR>
<TD>
This is 1st cell and 2nd row.
</TD>
<TD>
This is 2nd cell and 2nd row.
</TD>
</TR>
</TABLE>
```

Run it on a browser. You will get the result like this:

This is the first cell.	This is the second cell.
This is 1st cell and 2nd row.	This is 2nd cell and 2nd row

ADDING SPACES TO A TABLE

There are a couple of commands you can add to the <TABLE> tag to get more spacing between cells. They are as follows –

Cellspacing = "   "

Use this command to add more space around each cell. Place a number inside the double quotation marks.
To see how it affects, replace the first line of above mentioned example with this one-

<TABLE BORDER = "2" CELLSPACING = "12">

You will see the result like this –

This is the first cell	**This is the second cell**
This is 1st cell and 2nd row	**This is 2nd cell and 2nd row**

Cellpadding = "   "

This command is used to add more space inside each cell. You can place the number as required with double quotation marks.
Again replace the first line with the one to see the result of the cell padding command.

<TABLE BORDER = "2" CELLPADDING = "12">

The table will look like this –

This is the first cell	**This is the second cell**
This is 1st cell and 2nd row	**This is 2nd cell and 2nd row**

You can also add both commands together inside **<TABLE....>** tag this way –

<TABLE BORDER = "2" CELLSPACING="15" CELLPADDING="15">

Which you can replace with the first line of above mentioned example .It will result like this –

This is the first cell	**This is the second cell**
This is 1st cell and 2nd row	**This is 2nd cell and 2nd row**

You can add anything inside the cell of a table. You can add a link, an image, heading or paragraph.

EXAMPLE shows a hyperlink having added inside the cell.

```
<TABLE>
<TD>
<A HREF = "http://www.yahoo.com"> Go Yahoo!</A>
</TD>
</TABLE>
```

You will see the result like this – Go Yahoo!

DETERMINING THE TABLE SIZE

Sometimes, you need your table in a specific size. You can define the width of your table to help you get the table in the size you would like it to be. To do this, add the width =" " command to your <TABLE> tag. Place the number of pixels wide you would like the table to be between the double quotation marks. So, if you wanted a table 600 pixels long, you would do this:

<TABLE WIDTH = "600" BORDER = "2">
The above command makes table 600 pixels long.
See example**:**

```
<TABLE WIDTH = "600" BORDER = "2">
<TD>
The Table is 600 pixels long.
</TD>
</TABLE>
```

Example Result Below –

The Table is 600 pixels long.

The above output clearly illustrates your table border is much longer than the text it contains.

ALIGNING THE CONTENTS OF THE TABLE

Contents in a table can be arranged from right side, from left side or in the centre with the help of **ALIGN=" "command**. We add this command in **<TD>** tag.

This command can be used in the following three ways -
Align = "left" sets the contents in a table from left side.
Align = "right" sets the contents in a table from right side.
Align = "center" sets the contents in a table in the center.

See Example:

```
<TABLE WIDTH="450" BORDER="2"
CELLSPACING="7" CELLPADDING="7">
<TD ALIGN = "CENTER">
This cell is in the center.
</TD>
<TD ALIGN = "RIGHT">
This cell is right aligned.
</TD>
<TD ALIGN = "LEFT">
This cell is left aligned.
</TD>
</TABLE>
```

Example results like this -

This cell is in the Center.	This cell is right aligned	This cell is left aligned

In the above picture, text in the first cell is quite centered while second and third cells are arranged from right and left side respectively.

Apart from the command mentioned above, there are a few more commands that align the contents vertically. For this, add **VALIGN = " "** command inside **<TD>** tag. Commands that align vertically are as follows –

valign = "top"
arranges the contents quite at the top in the cell.
valign = "middle"
arranges the contents in the centre of the cell.
valign = "bottom"
arranges the contents at bottom in the cell.
You can see the use of these commands in example below:

```
<TABLE WIDTH="550" BORDER="2" CELLSPACING="7" CELLPADDING="0”>
<TD ALIGN = "CENTER" VALIGN = "TOP'S
The cell is at the top. <BR>
Really! It is top.
</TD>
<TD ALIGN = "CENTER" VALIGN = "MIDDLE'S
The cell is in the middle.
</TD>
<TD ALIGN = "CENTER" VALIGN = "BOTTOM'S
The cell is at bottom.
</TD>
</TABLE>
```

Table as a result of example will display like this –

This cell is at the top. Really! It is top.	This cell is in the middle	This cell is at bottom.

The vertical alignment commands only come in useful if your table cells do not have the same number of lines inside each cell. Since the first cell has 2 lines and others have one line, the vertical alignment makes difference in how the table looks. Had all the cells contained one line only, there would have been no use of VALIGN at all.

SPANNING ROWS AND COLUMNS
You can span rows and columns. It is mainly used to write a title of rows or columns. There are two commands for this as follows –

Rowspan = " "
Defines the number of vertical table rows the cell should take up. Place the number as required inside the double quotation marks.

Colspan = " "
Defines the number of horizontal columns the cell should take up.
For instance, the following type of table can be prepared using the codes written in example:

{Hello! Mr.........	Microsoft.Press
How are you?	

```
<TABLE BORDER = "2">
<TD ALIGN = "CENTER”>
 Hello! Mr ..........
</TD>
<TD ROWSPAN = "2" ALIGN = "CENTER'S
<IMG SRC = "PIC1.GIF">
</TD>
<TR>
<TD ALIGN = "CENTER”>
How are you?
 </TD>
</TR>
</TABLE>
```

Similarly, to make a table. We write html codes as written in example.

8PRIDE OF CRICKET		
SACHIN TENDULKAR	BRIAN LARA	IMRAN KHAN

***EXAMPLE*:**

```
<TABLE BORDER = "2" CELLPADDING = "5">
<TD COLSPAN = "3" ALIGN = "CENTER">
<B> PRIDE OF CRICKET </B>
</TD>
<TR>
<TD ALIGN = "CENTERS
SACHINTENDULKAR
</TD>
<TD ALIGN = "CENTERS
BRIAN LARA
</TD>
<TD ALIGN = "CENTERS
IMRAN KHAN
</TD>
</TR>
</TABLE>
```

In example colspan is used. We use the colspan command and set ***colspan="3"*.** We get the first cell to stretch across the other three on the second row below it.

<u>SPECIFYING BACKGROUND COLOUR OF THE TABLE</u>

You can change the background color of the entire table, each row, or each cell. To change the background colour of the table, **bgcolor=" "** command is used like this –

<TABLE BORDER = "5" BGCOLOR = "RED">

In the above command **BGCOLOR** has been used in the <TABLE> tag. To understand the use, see the example:

```
<TABLE BORDER = "5" BGCOLOR = "RED">
<TD>
IT IS QUITE REDDISH !
</TD>
</TABLE>
```

Example results something like this -

IT IS QUITE REDDISH

The entire table gets its background red. To change color of each cell in the table. **BGCOLOR** command is used in each **<TD>** tag. **See example :**

```
<TABLE WIDTH = "75" BORDER = "2">
<TD BGCOLOR = "RED">
RED
</TD>
<TD BGCOLOR = "BLUE">
BLUE
</TD>
</TABLE>
```

After you run the above mentioned example, figure like this appears on the browser –

RED	**BLUE**

Both cells have different colors. The cell that contains RED has Sot red background while the cell that contains BLUE has got blue background. Let us see example in which a table containing two columns and two rows have been created and both rows have been provided two different colors.

```
<TABLE WIDTH = "200" BORDER = "2">
<TR BGCOLOR = "RED">
<TD>RED</TD>
<TD> AGAIN RED </TD>
 </TR>
<TR BGCOLOR = "BLUE">
<TD> BLUE </TD>
 <TD> BLUE AGAIN </TD>
</TR>
 </TABLE>
```

You will see the effects of example like this -

RED	**AGAIN RED**
BLUE	**BLUE AGAIN**

Thus, you can prepare calendar, price-list or whatsoever using tables.

CREATING TABLES WITHIN TABLES

Suppose you had this problem: You need to put two tables on the same line on your page. Oh no! the table tag automatically sends you to the next line! Well, you can set around this by placing your two tables inside one large table, thus keeping them on the same line.
Let us start by placing one table inside another. **See example:**

```
<TABLE WIDTH = "400" BORDER = "6">
<TD ALIGN = "CENTER">
<TABLE WIDTH = "300" BORDER = "2">
<TD ALIGN = "CENTER">
This is a table within a bigger table.
</TD>
</TABLE>
</TD>
</TABLE
```

Result of the Example on the Browser:-

This is a table within a bigger table.

This can be a bit confusing at times. Just remember to keep track of which table you are in while you are writing the code.
As for the problem at the beginning of the section, all we have to do is add another table cell to the big table, and then use a second smaller table inside that cell. To hide the appearance of the big table, we set the border on the big table to zero. **See example:**

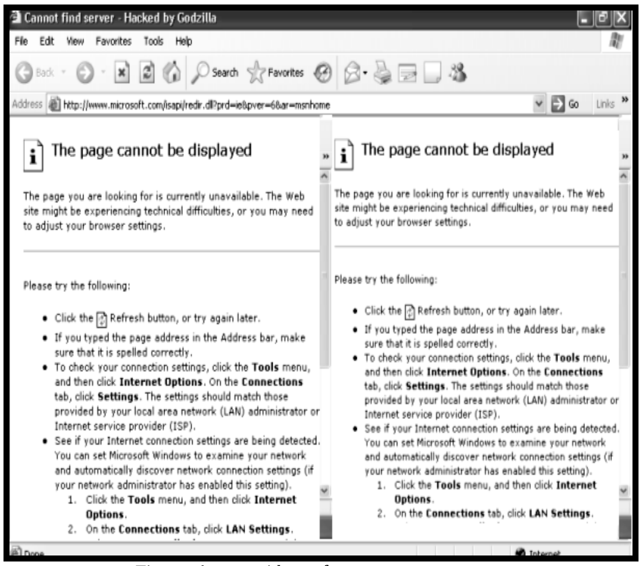
```
<TABLE WIDTH=”600” BORDER= “0”>
<TD ALIGN = "CENTER">
<TABLE WIDTH = "275" BORDER = "4">
<TD ALIGN = "CENTER">
I am in the second small table! Ha!
<TD>
</TABLE>
</TD ALIGN=”CENTER”>
<TABLE WIDTH = “275” BORDER = “4”>
</TD ALIGN=”CENTER”>
I am in the second small table! Wow!
<TD>
</TABLE>
<TD>
</TABLE>
```

Now we see the two inside tables
 on the page side by side like this –

I am in the first small table! Ha!	**I am in the second small table! wow!**

USING FRAMES

Prior to begin using frames on your web page, you will be wanting to know how frames work. A page with frames is really a page split into 2 or more sections, each containing its own html document. As you can see in figure given below, that shows a page with two frames.


Cannot find server - Hacked by Godzilla
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.microsoft.com/isapi/redir.dll?prd=ie&pver=6&ar=msnhome
Go Links
Done
Internet

The page cannot be displayed

The page you are looking for is currently unavailable. The Web site might be experiencing technical difficulties, or you may need to adjust your browser settings.

Please try the following:

- Click the Refresh button, or try again later.
- If you typed the page address in the Address bar, make sure that it is spelled correctly.
- To check your connection settings, click the **Tools** menu, and then click **Internet Options**. On the **Connections** tab, click **Settings**. The settings should match those provided by your local area network (LAN) administrator or Internet service provider (ISP).
- See if your Internet connection settings are being detected. You can set Microsoft Windows to examine your network and automatically discover network connection settings (if your network administrator has enabled this setting).
  1. Click the **Tools** menu, and then click **Internet Options**.
  2. On the **Connections** tab, click **LAN Settings**.

The page cannot be displayed

The page you are looking for is currently unavailable. The Web site might be experiencing technical difficulties, or you may need to adjust your browser settings.

Please try the following:

- Click the Refresh button, or try again later.
- If you typed the page address in the Address bar, make sure that it is spelled correctly.
- To check your connection settings, click the **Tools** menu, and then click **Internet Options**. On the **Connections** tab, click **Settings**. The settings should match those provided by your local area network (LAN) administrator or Internet service provider (ISP).
- See if your Internet connection settings are being detected. You can set Microsoft Windows to examine your network and automatically discover network connection settings (if your network administrator has enabled this setting).
  1. Click the **Tools** menu, and then click **Internet Options**.
  2. On the **Connections** tab, click **LAN Settings**.

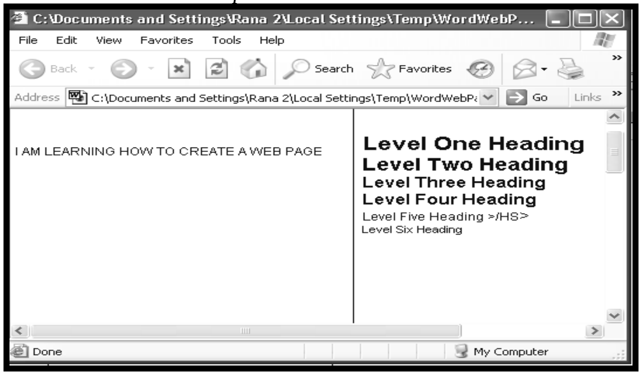
Figure: A page with two frames

As in the figure both frames are actually two different web pages. The page that keeps the two frames has also its own html document. In this section we shall discuss how to create frames. Frame is built using the <FRAMESET> tag rather than <BODY> tag at the beginning of the document.

See example:

```
<HTML>
<HEAD>
<TITLE> My First Page with Frames </TITLE>
</HEAD><FRAMESET COLS = "50%, 50%">
<FRAME SRC = "FRAME1 .HTM">
<FRAME SRC = "FRAME2.HTM">
</FRAMESET>
</HTML>
```

Example result on the browser:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebP...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebP
Go Links
Done
My Computer

I AM LEARNING HOW TO CREATE A WEB PAGE

Level One Heading
Level Two Heading
Level Three Heading
Level Four Heading
Level Five Heading >/HS>
Level Six Heading

Let's have a look at elements used in the above mentioned example -

<FRAMESET>

This tag tells the web browser to expect a series of frames rather than a normal page.

cols ="50%, 50%"

This command inside the FRAMESET tag tells the browser to split the page into two columns. In this case, each column would take up 50% of the space on the screen. You can change the percentage to any number you like. You can also use pixels rather than percentage if you wish. If you use percentages, be sure to keep the % sign after each number, or the browser will read the number as a pixel value.

<FRAME SRC = "FRAME1.HTM">

This tag lets you tell the browser the URL of the document in the frame farthest to the left.

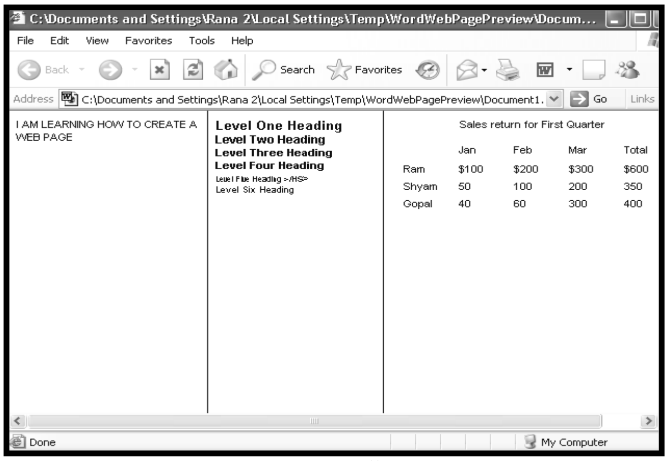
<FRAME SRC = "FRAME2.HTM">

This tag will specify the URL of the next frame, going from left to right. The browser will read your FRAME SRC tags for the columns from left to right. For this everything should be in order.

Example:

```
<HTML>
<HEAD>
<TITLE> MY AGE WITH THREE FRAMES </TITLE>
</HEAD>
<FRAMESET COLS = "33%, 33%, 33%">
<FRAME SRC = "Frame1.htm">
<FRAME SRC = "Frame2.htm">
<FRAME SRC = "Frame3.htm">
</FRAMESET>
</HTML>
```

After you implement example on browser, the effects will appear on the browser like one hi figure given below:
The page now contains three columns and all are almost equal as 33% has been provided to each frame in the code. Rest 1% is made by the browser itself. If you do not want to leave it to browser, change any of the values to 34%.


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Docum...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Document1.
Go Links
Done
My Computer

I AM LEARNING HOW TO CREATE A WEB PAGE

Level One Heading
Level Two Heading
Level Three Heading
Level Four Heading
Level Fibe Heading >/HS>
Level Six Heading

Sales return for First Quarter

	Jan	Feb	Mar	Total
Ram	$100	$200	$300	$600
Shyam	50	100	200	350
Gopal	40	60	300	400

Figure: page with three different documents

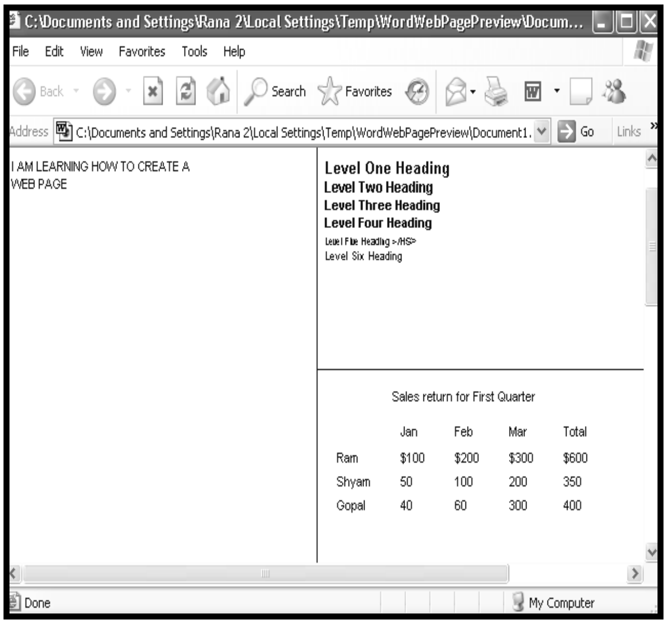
ADDING COLUMNS HORIZONTALLY AND VERTICALLY

Suppose you want two frames vertically and one frame divided into two horizontal frames. For this, we use rows instead of cols for frames that go from top to bottom.

See example:

```
<HTML>
 <HEAD>
<TITLE> My Page with Mixed Frames </TITLE>
</HEAD>
<FRAMESET cols = "50%, 50%">
<FRAME SRC = "FRAME1 .HTM">
<FRAMESET ROWS = "50%, 50%">
<FRAME SRC = "FRAME2.HTM">
<FRAME SRC = "FRAME3.HTM">
</FRAMESET>
</HTML>
```

See result of the above mentioned example on the browser:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Docum...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Document1. Go Links
Done
My Computer

I AM LEARNING HOW TO CREATE A WEB PAGE

Level One Heading
Level Two Heading
Level Three Heading
Level Four Heading
Level File Heading >/HS>
Level Six Heading

Sales return for First Quarter

	Jan	Feb	Mar	Total
Ram	$100	$200	$300	$600
Shyam	50	100	200	350
Gopal	40	60	300	400

Figure: A page based on three frames of mixed structure.

The rows command reads from top to bottom, like the cols command reads from left to right You can have as many columns or rows as you like, but be sure to nest your frameset tags the way you want the frames to appear:

- The first FRAMESET tag tells the browser to divide the page into two columns.
- The FRAME SRC tag following it tells the browser the first columns should be filled withframel.htm.
- The next FRAMESET tag is nested inside the first FRAMESET tag. This tag tells the browser to divide the second column into two rows, rather than using a single html page to fill the column.
- The next two FRAME SRC tags tell the browser to fill the two rows with frame2.htm at the topmost row and frame3.htm on the following row, moving from top to bottom.
- Be sure to close all of your FRAMESET tags after they have been used.

USING LINKING AND ATTRIBUTES ON THE FRAME

Now, you can create a page with frames. But when you place a link inside a frame, the URL you linked to will only appear in that same frame! This is because a link within a frame will default to showing the next page inside the frame the link is. In this case you would certainly like to have the link put the new page in another frame, or come up without any frames around it. For this, just name each frame, and these specify the target frame in your link.

See example:

```
<HTML>
<HEAD>
<TITLE My Page with Linked Frames </TITLE>
<FRAMESET cols = "20%, 80%">
<FRAME SRC = "FRAME1 .HTM" NAME = "Left_frame">
<FRAME SRC = "FRAME2.HTM" NAME = "right_frame">
</FRAMESET>
</HEAD>
</HTML>
```

In example, each frame has got a name, you can use either frames name as a target inside a link tag. Indirectly, you can say you have a link inside the Left_frame (frame1.htm). If you want the new page to show up in the right_frame, you need to add the target="frame_name" command to your link tag. The following example link tag will be in the left_frame, but make the result show up in the right_frame:

<A HREF="sarva.htm" target="right_frame"> About Sarva </A>

Now the left_frame will remain unchanged, while the linked URL will show up in the right frame. You can link to any frame you want to this way. Just remember to name all of your frames so that you can target them.

IMPORTANT ATTRIBUTES USED IN THE FRAME

Some important attributes are used in the frame that you can add to your individual <FRAME> tags to help you control the design of the frames. They are as follows –

scrolling = "no"

This command will let you specify whether or not you want a scroller on the right side of the frame for users to scroll up and down. If you do not add this command, the browser will decide whether or not to add a scroller based on the length of the page inside the frame. If set to "yes", the frame will always have a scroller. If set to "no" the frame will have no scroller.

border = "2"

This command lets you specify the width of the frame border. You can set a number of your choice.

resize = "no"

This command lets you decide whether or not you want your viewers to be able to resize a frame by dragging the border across the page. If set "yes", users can resize the frame and if set to "no" the frame can not be resized. The default is yes.

More size

You may want to use this in place of resize="no", as not all browsers support the resize="no" command. Using both commands, you can safely present the users the scroll bar.

margin width = "2" and marginheight = "2"

These commands let you determine the margins between the frame and the contents of the frame.

USING THE FORM

Forms are created on web pages to get feedback. They can also be used as order form, form for survey on a special subject or for similar purposes. You can create your own form using the techniques being discussed as ahead.

SETTING UP A FORM

To use a form on your page, you will need two things. The HTML for the form and a CGI program to handle the form when it is submitted.
To get a CGI program for you, you can see if your web server has one available for your use and find out from them how to use it in your HTML form code. Most web servers offer CGI programs for you to use for free. The most common program to handle a form is one that sends you the results via email, and this is the program most servers will provide.

To create a form, two elements are specified –

(1) METHOD property within <FORM> tag

(2) ACTION property within <FORM> tag

The **METHOD** command/property will almost always be set to "post". The other value is "get", but if you are using a program from your web server, you will likely be instructed to use method = "post".
The **ACTION** command is asking for the address of a CGI program that will handle the form once it is submitted.

You will replace the:

"/cgi-bin/mansoor.PI"

with the exact address your server gives you to use their cgi program. If you are using your own program, upload the program to your server and use the address of your program. Before you upload a cgi program, be sure your server allows you to use your own cgi programs. If your server doesn't allow you to use your own cgi programs, don't do it.

Let us know how to create some basic elements used on the form. The basic elements on a form are as follows –

- **TEXT BOX**
- **CHECK BOX**
- **RADIO BUTTON**
- **MENU**
- **SUBMIT AND RESET BUTTONS**

CREATING A TEXT BOX

After writing opening tag <FORM>, you can insert any element as required on a form. On your form, the most used element is usually a text box. Text box lets your users enter a line of text as required. To create this, do the followings –

1. Within both form tags <FORM> and </FORM>, type the text you want to use as a label for the text box like <P> Name and press Enter.
2. Then, type <INPUT TYPE = "text" and leave space.
3. Type Name="name". Replace the name within the double quotation marks with a suitable name that describes the text box.
4. Type Size = "n" to specify the width of the text box. Replace the n with a number required as the size of the text box. n is the number of characters used in the text box. For instance size = "20"
5. Type MAXLENGTH="n" to limit the number of maximum characters used in the text box. Change the **n** with a number you want maximum characters in the text box. Like MAXLENGTH = "30"
6. To end it, use the > key. To understand the entire process,

See example:

```
<FORM METHOD = "POST" ACTION =
 "mailto:mansoor3176@rediffmail.com">
<P>Name
<INPUT TYPE="text" Name="text1” SIZE="20"
 MAXLENGTH="30">
</FORM>
```

After you run the example, picture like this will appear on the browser.

CREATING A LARGE TEXT BOX

The text box created in the earlier section has got a limit. Using that text box, You can let your users enter only one line in the text box. What about if your users need enter more than one line in the text box. In this situation, you need a text box that should be large and contains multiple lines. To create a large text box, do the followings **–**

1. Type the text representing a label for the text box within both form tags <FORM> and </FORM> like - <P> Comments/Suggestions and press Enter.
2. Type <TEXTAREA and press spacebar
3. Then, type Name=" ". Within double quotation marks type a suitable word o name the text box. **For example:** Name = "Suggestions" and leave space
4. Type Rows = "n" and replace n with the number you want to allow your users type the lines in the box. eg. Rows = "5"
5. Type Cols="n" to specify the width of text box. Replace the n with the number you want to set the width of text box like COLS="30"
6. Type WRAP> to use word wrap feature in the text box. Word wrap feature Auto-matically changes the line when the column limit of the text box is over.
7. Type </TEXTAREA> to close the tag.

See the example to look at the HTML codes.

```
<HTML>
<HEAD>
<TITLE> FEEDBACK FORM </TITLE>
</HEAD>
<P> Comments/Suggestions
<FORMMETHOD="POST"ACTION="mailto:mansoor3176@yahoo.com">
<TEXTAREA Name="Suggestions" Rows="5" Cols="30" WRAP>
</TEXTAREA>
</HTML>
```

After you run the example the browser will display like this –

CREATING A RADIO BUTTON

Radio button is used when your users have multiple choices to select and you allow them to choose one that is the most appropriate one. Before you create a radio button, you need provide some basic information to the browser as follows –

- Selection of a word representing the whole group of options. It can be done using Name attribute.
- Selection of a word for each radio button. Value attribute is used for this.
- Selection of a word that will work as a label for each radio button. Now follow the instructions to create a radio button –

1. Type <INPUT TYPE = "radio" between both form tags <FORM> and </ FORM>
2. Type Name = " ". Name the word under double quotation marks that will represent the entire group of radio buttons.
3. TYPE VALUE = " " to specify the information for one radio button. Type the word under double quotation marks that describes the radio button.
4. Type CHECKED, if you want the radio button automatically selected. You can have this with only one radio button.
5. TYPE > to complete the radio button.
6. Type the text you want to appear beside the radio button on your web page.
7. Repeat steps from 2, 3, 5 and 6 for as many radio buttons as you want to create.

See the example to understand the HTML codes –

```
<HTML>
<HEAD>
<TITLE> Feedback Form </TITLE>
</HEAD>
<P>Marital Status
<FORM METHOD="POST"
 ACTION="mailto:mansoor3176@rediffmail.com">
<INPUT TYPE="radio" Name="marital" Value="Married"> Married
<INPUTTYPE="radio" Name="marital" Value="Unmarried"
CHECKED> Unmarried
</FORM>
</HTML>
```

After you run the example on the browser, picture like this will display -

Marital Status

○ **Married** ○ **Unmarried**

CREATING A CHECK BOX

You can build a set of check boxes on a form if you want visitors to be able to select one or more options. When creating check boxes, you need to specify the same information, you did while creating radio buttons. After understanding the information to be specified, do the followings to accomplish creation of check boxes –

1. Type <INPUT TYPE = "checkbox" between both form tags <FORM> and </FORM> and then leave space.
2. Type Name = " " and under double quotation marks type the word that defines the group of check boxes like Name = "sports"
3. Then type VALUE = " " to specify the information for each check box. Type the word under double quotation marks that describes the check box such as VALUE = "football".
4. Type CHECKED if you want a special check box to be selected automatically.
5. Type > to complete the check box.
6. Type the text you want to use beside the checkbox.
7. Repeat steps 1, 2, 3, 5 and 6 every time you want a check box be added in the set. See example to understand the html codes.

```
<HTML>
<HEAD>
<TITLE> Feedback Form </TITLE>
</HEAD>
<P> Hobbies
<FORMMETHOD="POST"
ACTION="mailto:mansoor3176@rediffmail.com">
<INPUT TYPE="checkbox" Name="hobbies" Value="Cricket"
CHECKED>
Cricket<BR>
<INPUT TYPE="checkbox" Name="hobbies" Value="Music">
Music<BR>
<INPUT TYPE="checkbox" Name="hobbies"
Value="Travelling">
Traveling<BR>
</FORM>
</HTML>
```

After you run the example, the picture like this comes up on the browser

Hobbies
Cricket
Music
Travelling

CREATING A MENU

You can create a menu that offers visitors a list of options to choose from. Normally menus are used for displaying lists of products, states or age groups. While creating radio buttons, you must specify the same thing specified in the earlier radio buttons and check boxes and then proceed to the following steps –

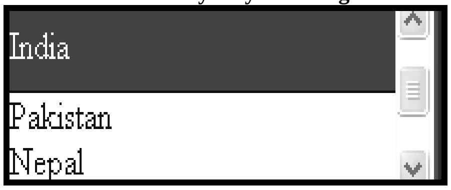
1. Type <SELECT NAME = " " between both form tags <FORM> and </ FORM>. Insert a word under double quotations that describes the menu you want to create. And then press SPACEBAR. For example <SELECT NAME = "COUNTRY"
2. Type SIZE = "n" and replace n with the number of options you want readers to see in the menu without using the scroll bar. eg. SIZE = "3"
3. Type <OPTION VALUE = " " and insert a word under double quotations that describe^ the menu option.
4. Type the text you want to appear for the menu option.
5. Repeat steps 3 and 4 for each menu option you want to add.
6. Type SELECTED if you want a particular menu selected. You can have this with one.

See example to look through the HTML codes.

```
<HTML>
<HEAD>
<TITLE> FEEDBACK FORM </TITLE>
<P> Which country do you belong to?
<FORM METHOD = "POST"
ACTION="mailto:mansoor3176@rediffmail.com”>
<SELECT NAME="Country" SIZE="3">
<OPTION VALUE = "India" Selected> India
<OPTION VALUE = "Pakistani”> Pakistan
<OPTION VALUE = "Nepal"> Nepal
<OPTION VALUE = "Bangladesh"> Bangladesh
<OPTION VALUE = "Bhutan"> Bhutan
<OPTION VALUE = "Sri Lanka"> Sri Lanka
<OPTION VALUE = "Malaysia"> Malaysia
</SELECT>
</FORM>
</HTML>
```

After implementing the codes written in example on a browser, the picture that appears will match with this –

Which country do you belong to?



CREATING A SUBMIT BUTTON

You can build a SUBMIT button which lets the user to click in order to store the information an your web server or on your e-mail account. To create SUBMIT button, do the followings –

1. Type <INPUT TYPE = "Submit" between both form tags <FORM> and </ FORM> and leave spaces.
2. Type VALUE = " ". Put the word under double quotation marks that appears on the button like VALUE = "ACCEPT"

See example:

```
<HTML>
<HEAD>
<TITLE> FEEDBACK FORM </TITLE>
</HEAD>
<FORMMETHOD="POST"ACTION="mailto:mansoor3176@yahoo.com">
<INPUT TYPE="SUBMIT" VALUE="ACCEPT">
</FORM>
</HTML>
```

After you run the example, the output on the browser will appear like this-

CREATING A RESET BUTTON

You can create a RESET button that allows your users to refresh the information entered by the user to create this button, do the followings –

1. Type <INPUT TYPE = "reset" between both form tags <FORM> and </ FORMx
2. Type VALUE = " " and put a word that will appear on this button like VALUE = "RESET"

See example:

```
<HTML>
<HEAD>
<TITLE> FEEDBACK FORM </TITLE>
</HEAD>
<FORM METHOD="POST"
ACTION="mailto:mansoor3176 ©yahoo. com">
<INPUTTYPE="RESET" VALUE="REFRESH">
</FORM>
</HTML>
```

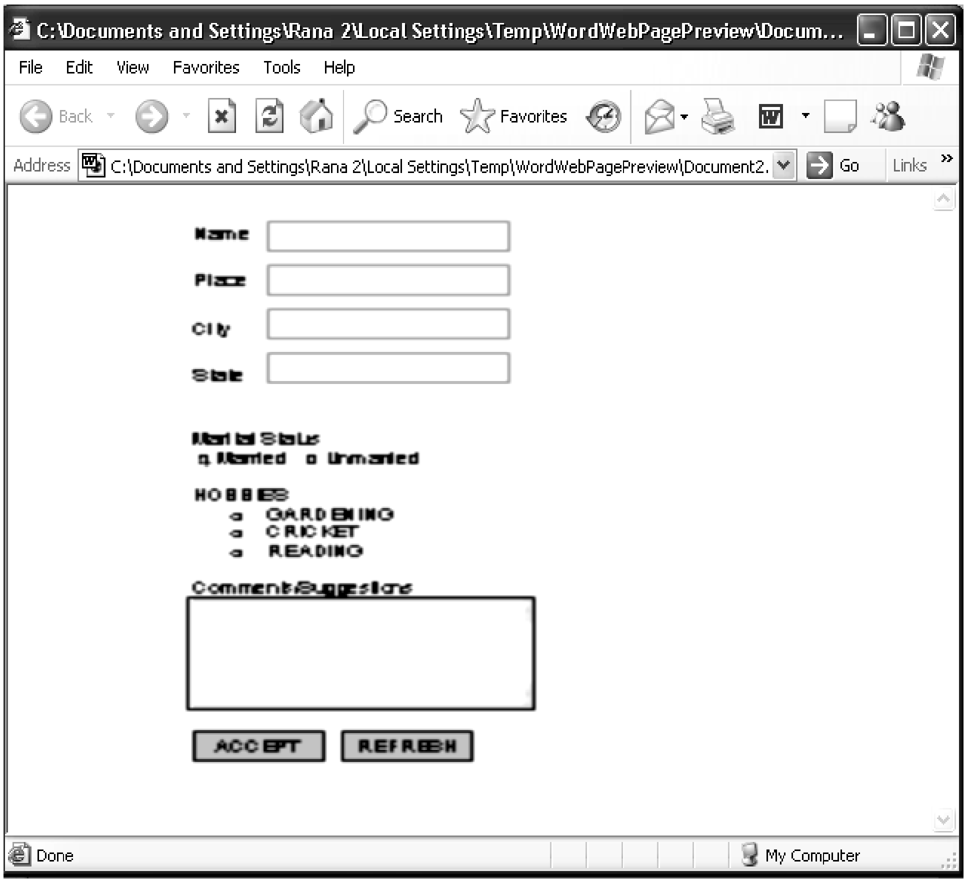
After you run the example, the result will be shown like this on the browser -

So far, you have seen how to create various elements on a form. Now we write codes together in example to create a form incorporating all the elements discussed earlier.
***Example*:**

```
<HTML>
<HEAD>
</HEAD>
<FORM METHOD="POST"
ACTION="mailto:mansoor3176@rediffmail.com">
<P> Name <INPUT TYPE="TEXT" NAME="prof»
SIZE="20"
MAXLENGTH="30">
<P> Place <INPUTTYPE="TEXT» NAME="prof"
SIZE="20"
MAXLENGTH="30">
<P> City <INPUTTYPE=" TEXT" NAME="prof"
SIZE="20"
MAXLENGTH="30">
<P> State <INPUTTYPE="TEXT" NAME="prof"
SIZE="20"
MAXLENGTH="30">
<P> Marital Status<BR>
<INPUT TYPE="radio" Name=”marital”
Value="Married"> Married
<INPUT TYPE="radio" Name="marital"
Value="Unmarried" CHECKED>Unmarried
```

```
<P>HOBBIES <BR>
<INPUT TYPE=»CHECKBOX" NAME="HOB" VALUE="GARDENING"
CHECKED>GARDENING<BR>
                    <INPUTTYPE="CHECKBOX» NAME="HOB" VALUE="CRICKET»>CRICKET<BR> <INPUTTYPE=»CHECKBOX"
                    NAME="HOB" VALUE="READING">READING<BR>
<P> Comments/Suggestions<BR>
<TEXTAREA Name="Suggestions" Rows="5" Cols="30" WRAP>
</TEXTAREA> <BR>
<BR>
<BR>
<INPUT TYPE="SUBMIT" VALUE="ACCEPT">
<INPUT TYPE="RESET" VALUE="REFRESH">
</FORM>
</HTML>
```

After you implement the example the effects on the browser can be shown in figure below:-


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Docum...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Document2. Go Links
Name
Place
City
State
Marital Status
Married Unmarried
HOBBIES
GARDENING
CRICKET
READING
Comments/Suggestions
ACCEPT REFRESH
Done My Computer

Unit-III

JAVA SCRIPT

INTRODUCTION

As you might have come across a lot of programming languages that help in writing web pages with a lot of features. Unlike all those programming languages, JavaScript has got its own charm, hence received overwhelming acceptance from the developers community. This chapter covers various aspects of this very language.

JAVASCRIPT- AN OVERVIEW

JavaScript is a scripting language which is similar in syntax to Java. It was developed by Netscape Communications Corporation. Netscape Communications Corporation is the same enterprise that had developed once the most popular Internet browser i.e. Netscape Navigator. Netscape Navigator is a part of Netscape suite. JavaScript was originally called **Livescript** when it was first introduced in Netscape Navigator 2.0 in 1995. However, it changed to JavaScript in order to setup relations with Java language.

DIFFERENCE BETWEEN JAVASCRIPT AND JAVA

JavaScript and Java are similar in some ways but fundamentally different in others.
The JavaScript and Java are similar in some ways but fundamentally different in others. The JavaScript language resembles Java but does not have Java's static typing and strong type checking. JavaScript supports most Java expression syntax and basic control-flow constructs. In contrast to Java's compile-time system of classes built by declarations, JavaScript supports a runtime system based on a small number of data types representing numeric, Boolean, and ring values. JavaScript has a simple, instance-based object model that still provides significant capabilities. JavaScript also supports functions without any special declarative requirements. Functions can be properties of objects, executing as loosely typed methods.
Java is an object-oriented programming language designed for fast execution and type safety. Type safety means, for instance, that you can't cast a Java integer into an object reference or access private memory by corrupting Java byte codes. Java's object-oriented model means that programs consist exclusively of classes and their methods. Java's class inheritance and strong typing generally require tightly coupled object hierarchies. These requirements make Java programming more complex than JavaScript authoring.

In contrast, JavaScript descends in spirit from a hue of smaller, dynamically typed languages like Hyper Talk and dBASE. These scripting languages offer programming tools to a much wider audience because of their easier syntax, specialized built-in functionality, and minimal requirements for object creation. The basic differences between JavaScript and Java are summarized in table given below.

JAVA SRIPT	JAVA
Interpreted (not compiled) by client.	Compiled byte codes downloaded from server, executed on client.
Object-based. Uses built-in, extensible objects, but no classes or inheritance.	Object-oriented. Applets consist of object classes with inheritance.
Code integrated with, and embedded in, HTML.	Applets distinct from HTML (accessed from HTML pages).
Variable data types not declared (loose typing).	Variable data types must be declared (strong typing).
Dynamic binding. Object references checked at runtime.	Static binding. Object references must exist at compile-time.
Cannot automatically write to hard disk.	Cannot automatically write to hard disk.

VERSIONS OF JAVASCRIPT

The JavaScript language has evolved since its original release in **netscape 2.0**. There have been several versions of JavaScript:
JavaScript 1.0 - This was the original version, supported by Netscape 2.0 and Internet Explorer 3.0.
JavaScript 1.1 - This version was supported by Netscape 3.0 and mostly supported by Internet Explorer 4.0.
JavaScript 1.2 - This version of JavaScript was supported by Netscape 4.0 and partially supported by Internet Explore 4.0.
JavaScript 1.3 - This is supported by Netscape 4.5 and Internet Explorer 5.0.
JavaScript 1.5 - This version is supported by Netscape 6.0 and most of its features are supported by Internet Explorer 5.5 and later.

SYNTAX AND CONVENTIONS

JavaScript lets you embed programs right in your web pages and run these programs using the web browser. You place these programs in a **<SCRIPT>** element, usually within the **<HEAD>** element. If you want the script to write directly to the web page, place it in the **<BODY>**

element. The basic syntax-of a JavaScript program/script is as follows:

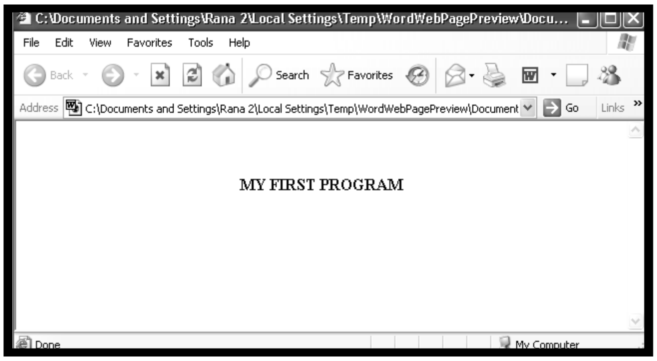
```
<HTML>
     <HEAD>
           <TITLE>
                 Write title of the page here
           <TITLE>
     </HEAD>
     <BODY>
        <SCRIPT   LANGUAGE="JavaScript">
             .
             .
.
.
        </SCRIPT>
    </BODY>
</HTML>
```

WRITING A SIMPLE JAVASCRIPT PROGRAM

Writing a script in JavaScript is similar to writing codes in HTML. Start text editor whatsoever available on your computer. Most likely, all of you prefer using Microsoft Notepad. **OK,** start Notepad and start typing following script as you have written HTML codes in the previous chapters.

```
<HTML>
    <HEAD>
        <TITLE>
             The First JScript Program
        </TITLE>
    </HEAD>
    <BODY>
        <Script Language=``JavaScript``>
        <!--
            document. writeln("My First Program")
        //-->
        </SCRIPT>
        <CENTER>
             <H1>
                  MY FIRST PROGRAM
             <H1>
        </CENTER>
     </BODY>
</HTML>
```

After typing all the lines in the Notepad, save it with **jscript I.htm** and see on the browser. The browser after having the script run should look like as shown in figure given below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Docu...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Document
Go Links
Done
My Computer

MY FIRST PROGRAM

In the previous program, beginning lines are according to the HTML standards. Normally we start writing the same way. However, when we write something of JavaScript in an HTML document, we should not forget to enclose them between **<SCRIPT>...... </SCRIPT>** tag pair as you can see in the above example. These tags instruct browser to deal with the lines between the very tag pair as JavaScript.

VARIABLES AND THEIR NAMING RULES

Variables are just place holders that can store data. Data may be a number, a text string, or an object. Like in any programming language, naming variables has got some specific rules here in JavaScript too. They are as follows:

- Variable names may be alphabetical and alphanumeric. They can be written in uppercase and lower case also. You can use digits 0 to 9 and can contain an underscore (_).
- Variables may start with an underscore or a letter. Remember variable names must not start with a digit.
- Variable names should include no spaces or special characters except underscore.
- Variable names are case sensitive means age, Age and AGE are three different variable names.

Some examples of legal names are *Number_hits*, *temp99*, *and _name*.

VARIABLE SCOPES

We can declare a variable in two ways:

- By simply assigning it a value; for example, **x=42**
- With the keyword **var:** for example, **var x = 42**

When we set a variable identifier by assignment outside of a function, it is called a global variable, because it is available everywhere in the current document. When we declare a variable within a function, it is called a local variable, because it is available only within the inside a function that has already been declared as a global variable.

We can access global variables declared in one window or frame from another window or frame by specifying the window or frame name. For example, if a variable called ***phone Number*** is declared in a **FRAMESET** document, we can refer to this variable from a child frame as parent phone Number.

EXPRESSIONS

An *expression* is any valid set of literals, variables, operators, and expressions that evaluates to a single value; the value can be a number, a string, or a logical value.

Conceptually, there are two types of expressions: those that assign a value to a variable, and those that simply have a value. For example, the expression, **x = 7** is an expression that assigns **x** the value **seven**. This expression itself evaluates to seven. Such expressions use assignment operators. On the other hand, the expression **3+4** simply evaluates to **seven**; it does not perform an assignment. The operators used in such expressions are referred to simply as ***operators.***

JavaScript has the following types of expressions:

- Arithmetic: evaluates to a number, for example 2.1019
- String: evaluates to a character string, for example, "John" or "234"
- Logical: evaluates to true or false

The special keyword null denotes a null value. In contrast, variables that have not been assigned a value are undefined and will cause a runtime error if used as numbers or as numeric variables. Array elements that have not been assigned a value, however, evaluate to false. For example, the following code executes the function **myFunction** because the array element is not defined:

```
 MyArray=newArray()
if(!myArray["notThere"]
myFunction
```

CONDITIONAL EXPRESSIONS

A conditional expression can have one of two values based on a condition. The syntax is-

(*condition*) ? *val1* : *val2*

If ***condition i***s true, the expression has the value of **val1**. Otherwise it has the value of **val2.** You can use a conditional expression anywhere you would use a standard expression.

For example,

status = (age >= 18) ? "adult" : "minor"

This statement assigns the value **"adult"** to the variable status if age is eighteen or greater. Otherwise, it assigns the value **"minor"** to *status*.

LOOP STATEMENTS

A loop is a set of commands that executes repeatedly until a specified condition is met. JavaScript supports two loop statements: **for** and **while**. In addition, you can use the **break** and **continue** statements within loop statements.

for STATEMENT

A for loop repeats until a specified condition evaluates to false. The JavaScript for loop is similar to the Java and C for loop. A for statement looks as follows**:**

for ([*initial-expression*]; [*condition*]; [*increment-expression*]) {
statements
}

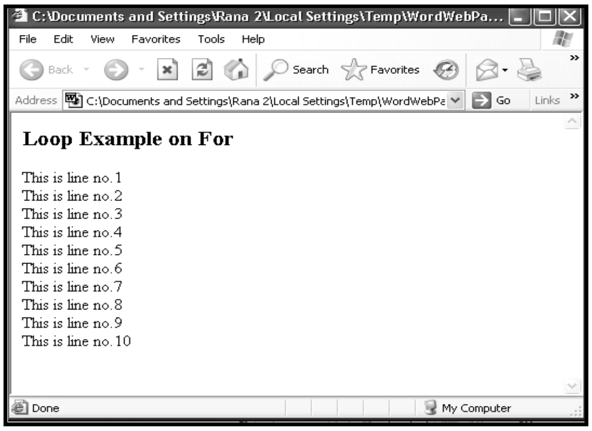
When a for loop executes, the following occurs:

- The initializing expression initial-expression, if any, is executed. This expression usually initializes one or more loop counters, but the syntax allows an expression of any degree of complexity.
- The condition expression is evaluated. If the value of condition is true, the loop statements execute. If the value of condition is false, the **for** loop terminates.
- The update expression increment-expression executes.
- The statements execute, and control returns to step 2.

***Example*:**

```
<html>
    <head>
        <title> Loops Example 1 </title>
    </head>
<body>
    <h1> Loop Example on For </h1>
<SCRIPT LANGUAGE= "JavaScript">
    for  (i=1; i<=10) {
    document.write("This is line no"+ i)
}
</script>
</body>
</html>
```

The above example shows a message with the loop's counter during each alteration. The loop is executed ten times. The counter is incremented every time until the expression is true i.e. i<=10. You can see the output as shown in figure given below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPa...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPa
Go Links
Done
My Computer

```
Loop Example on For

This is line no. 1
This is line no. 2
This is line no. 3
This is line no. 4
This is line no. 5
This is line no. 6
This is line no. 7
This is line no. 8
This is line no. 9
This is line no. 10
```

while STATEMENT

A while statement repeats a loop as long as a specified condition evaluates to true. A while statement looks as follows:

while(*condition*){
    statements
}

If the condition becomes false, the statements within the loop stop executing and control passes to the statement following the loop. The condition test occurs only when the statements in the loop have been executed and the loop is about to be repeated. That is, the condition test is not continuous but is performed once at the beginning of the loop and again just following the last statement in statements, each time control passes through the loop.

Example1. The following **while** loop iterates as long as n is less than three:

```
n=0
x=0
while(n<3){
n++
x+= n
}
```

With each iteration, the loop increments n and adds that value to x.

Therefore, x and n take on the following values:

- After the first pass: n = 1 and x = 1
- After the second pass: n = 2 and x = 3
- After the third pass: n = 3 and x = 6

After completing the third pass, the condition ***n* < *3*** is no longer true, so the loop terminates.

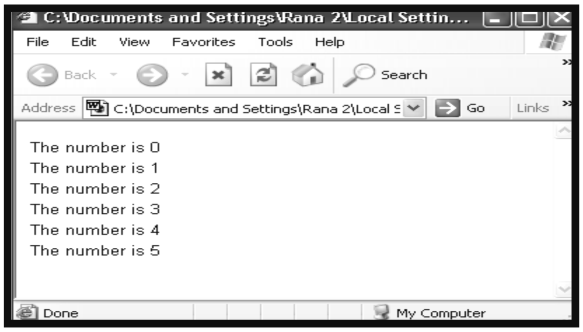
Example 2: infinite loop. Make sure the condition in a loop eventually becomes false;

Otherwise, the loop will never terminate. The statements in the following while loop execute forever because the condition never becomes false:

```
while(true) {
      alert("Hello, world")}
```

Example 3: This is a complete script on while

```
<html>
<head>
<title>While Example</title>
</head>
<body>
<script language="JavaScript">
i = 0
while (i <= 5)
{
document.write("The number is " + i)
document.write("<br>")
i + +
3
</script>
</body>
</html>
```


C:\Documents and Settings\Rana 2\Local Settin...
File Edit View Favorites Tools Help
Back Search
Address C:\Documents and Settings\Rana 2\Local S Go Links
Done My Computer

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

Explanation:

In this example i has been assigned to 0. The while loop continues to run till i is less than or equal to 5 and i will increase by 1 each time the loop runs. You can see in above mentioned figure how it results on the browser.

break STATEMENT

The break statement terminates the current while or for loop and transfers program control to the statement following the terminated loop. A break statement looks as follows:

break

Example. The following function has a break statement that terminates the **while** loop when *i* is three, and then returns the value 3 * x.

```
function testbreak(x) {
      var i = 0
        while(i<6) {
        if(i==3)
              break
        i++
      }
      return i*x
  }
```

continue STATEMENT

A **continue** statement terminates execution of the block of statements in a **while** or **for** loop and continues execution of the loop with the next iteration. A **continue** statement looks as follows**:**

continue

In contrast to the break statement, continue does not terminate the execution of the loop entirely. Instead,

- In a **while** loop, it jumps back to the condition.
- In a **for** loop, it jumps to the increment-expression.

Example: The following example shows a while loop with a continue statement that executes when the value of i is three. Thus, n takes on the values one, three, seven, and twelve.

```
i=0
n=0
while(i<5) {
     i++
     if(i==3)
          continue
n+=i
}
```

FUNCTIONS

A function is a definition of a set of deferred actions. Functions are invoked by event handlers or by statements elsewhere in the script. Whenever possible, good functions are designed to be reusable in other documents. They can become building blocks you can use over and over again. To use a function, you must first define it; then your script can call it.

DEFINING FUNCTIONS

A function definition consists of the function keyword, followed by:-

- The name of the function.
- A list of arguments to the function enclosed hi parentheses and separated by commas.
- The JavaScript statements that define the function, enclosed in curly braces, { }. The statements in a function can include calls to other functions defined in the current application.

In Navigator JavaScript, it is good practice to define all your functions in the HEAD of a page. so that when a user loads the page, the functions are loaded first.

For example, here is the definition of a simple function named **pretty_print:**

```
function pretty_print(str) {
        document.write("<HR><P>" + str)
    }
```

This function takes a string, *str*, as its argument, adds some **HTML** tags to it using the concatenation operator (+), and then displays the result to the current document using the **write** method.
In addition to defining functions as described here, you can also define Function objects, as described in **Function object.**

USING FUNCTIONS

In a Navigator application, you can use (or call) any function defined in the current page. You can also use functions defined by other named windows or frames.
Defining a function does not execute it. You have to call the function for it to do its work. **For example,** if you defined the example function **pretty_print** in the HEAD of the document, you could call it as follows:

```
<SCRIPT>
 pretty_print("This is some text to display")
 </SCRIPT>
```

The arguments of a function are not limited to strings and numbers. You can pass whole objects to a function, too. The **show_props** function (defined in "Objects and properties") is an example of a function that takes an object as an argument.
A function can even be recursive, that is, it can call itself. **For example,** here is a function that computes factorials:

```
    function factorial(n) {
          if(n==0) II (n=1))
               return 1
    else {
          result=(n*factorial)(n-1))
     return result
    }
 }
```

You could then display the factorials of one through five as follows:

```
 for(x=0; x<5; x++) {
document.write("<BR>", x, "factorial is", factorial(x))
}
```

The results are:

```
0 factorial is 1
1 factorial is 1
2 factorial is 2
3 factorial is 6
4 factorial is 24
5 factorial is 120
```

ARRAYS

An array is a numbered group of data items that you can treat as a single unit. **For example,** you might use an array called scores to store several scores for a game. Arrays can contain string, numbers, objects or other types of data.

CREATING AN ARRAY

Unlike most other types of JavaScript variables, you must declare an array before you use it The following example creates an array with 30 elements:

```
scores = new Array(30);
```

To assign values to the array, use brackets and an index. Indices begin with 0, so the elements of the array in this example would be numbered 0 to 29. These statements assign values to the first four elements of the array**:**

```
scores[0] =  39;
scores[1] =  40;
scores[2] = 100;
scores[3] =  49;
………
scores[29] = 52;
```

Like strings, arrays have a length property. This tells you the number of elements in the array, usually the same number you used when creating the array. For example, this statement would print the number 30;

```
document.write (scores.length);
```

ACCESSING ARRAY ELEMENTS

You can read the contents of an array using the same notation you used when assigning values. For example, the following statements would display the values of the first four elements of the *scores* array:

```
scoredisp = "Scores:" + scores[0] +"," +scores[1]
+"," + scores[2] +"," + scores[3];
document.write(scoredisp);
```

OBJECTS

JavaScript is based on a simple object-oriented paradigm. An object is a construct with properties that are JavaScript variables or other objects. An object also has functions associated with it that are known as the object's methods. In addition to objects that are built into the Navigator, you can define your own objects.

OBJECTS AND PROPERTIES

A JavaScript object has properties associated with it. You access the properties of an object with a simple notation:

```
objectName.propertyName
```

Both the object name and property name are case sensitive. You define a property by assigning it a value. For example, suppose there is an object named *my Car* (for now, just assume the object already exists). You can give it properties named *make, model,* and *year* as follows:

```
myCar.make = "Ford"
myCar.model = "Mustang"
myCar.year = 69;
```

An array is an ordered set of values associated with a single variable name. Properties and arrays in JavaScript are intimately related; in fact, they are different interfaces to the same data structure. So, for example, you could access the properties of the myCar object as follows**:**

```
myCar["make"] = "Ford"
 myCar["model"] = "Mustang"
  myCar["year"] = 67
```

This type of array is known as an associative array, because each index element is also associated with a string value. To illustrate how this works, the following function displays the properties of the object when you pass the object and the object's name as arguments to the function:

```
function show_props(obj, obj_name) {
var result =""
for (var i in obj)
        result += objjname +"." + ! + " = " + obj[i] + "\n"
 return result
 }
```

So, the function call show_props(myCar, "myCar") would return the following:

```
myCar.make = Ford
 myCar.model = Mustang
 myCar.year = 67
```

CREATING NEW OBJECTS

Both client and server JavaScript have a number of predefined objects. In addition, you can create your own objects. Creating your own object requires two steps:

- Define the object type by writing a constructor function.
- Create an instance of the object with new.

To define an object type, create a function for the object type that specifies its name, properties, and methods. **For example,** suppose you want to create an object type for cars. You want this type of object to be called car, and you want it to have properties for make, model, year, and color. To do this, you would write the following function:

```
function car(make, model, year) {
      this.make = make
      this.model = model
      this.year = year
   }
```

Notice the use of this to assign values to the object's properties based on the values passed to the function.
Now you can create an object called mycar as follows:

```
mycar = new car("Eagle", 'Talon TSi", 1993)
```

This statement creates mycar and assigns it the specified values for its properties. Then the value of *mycar*.make is the siring "Eagle," mycar.year is the integer 1993, and so on.

You can create any number of car objects by calls to new. For example,

```
kenscar = new car("Nissan", "300ZX", 1992)
```

An object can have a property that is itself another object. For example, suppose you define an object called person as follows:

```
function person(name, age, sex) {
        this.name = name
        this.age = age
        this.sex = sex
 }
```

and then instantiate two new person objects as follows:

```
rand = new person("Rand McKinnon", 33, "M")
ken = new person("Ken Jones", 39, "M")
```

Then you can rewrite the definition of car to include an owner property that takes a. per son object, as follows**:**

```
function car(make, model, year, owner) {
                this.make = make
                this.model = model
                this.year = year
                this.owner = owner
        }
```

To instantiate the new objects, you then use the following:

```
car1 = new car("Eagle", 'Talon TSi", 1993, rand)
 car2 = new car("Nissan", "300ZX", 1992, ken)
```

Notice that instead of passing a literal string or integer value when creating the new objects, the above statements pass the objects rand and ken as the arguments for the owners. Then if you want to find out the name of the owner of car2, you can access the following property: **car2.owner.name**

Note that you can always add a property to a previously defined object. For example, the statement
car1 .color = "black"
adds a property *color* to car 1, and assigns it a value of "black." However, this does not affect any other objects. To add the new property to all objects of the same type, you have to add the property to the definition of the car object type.

USING THIS FOR OBJECT REFERENCES

JavaScript has a special keyword, this, that you can use within a method to refer to the current object. For example, suppose you have a function called validate that validates an object's value property, given the object and the high and low values:

```
function validate(obj, lowval, hival) {
  if ((obj.value < lowval) II (obj.value > hival))
    alert("lnvalid Value!")
  }
```

Then, you could call **validate** in each form element's on Change event handler, using this to pass it the form element, as in the following example:

```
<INPUT TYPE = "text" NAME = "age" SIZE = 3
 onChange="validate(this, 18, 99)">
```

In general, **this** refers to the calling object in a method.
When combined with the *form* property, **this** can refer to the current object's parent form,
In the following example, the form *myForm* contains a Text object and a button. When the user clicks the button, the value of the Text object is set to the form's name. The button's onClick event handler uses this.form to refer to the parent form, myForm.

```
<FORM NAME="myForm">
  Form namexINPUT 7vpE="text" NAME="text1"
VALUE="Beluga">
    <P>
  <INPUT NAME="button1" TYPE="button"
VALUE="Show Form Name"
    onClick="this.form.text1.value=this.form.name">
    </FORM>
```

OBJECT DELETION

In JavaScript for Navigator 2.0, you cannot remove objects — they exist until you leave the page containing the object. In JavaScript for Navigator 3.0, you can remove an object by setting its object reference to null (if that is the last reference to the object). JavaScript finalizes the object immediately, as part of the assignment expression.

DOCUMENT OBJECT MODEL

Before you can truly start scripting, you should have a good feel for the kinds of objects you will be scripting. A scriptable browser does a lot of the work creating software objects that generally represent the visible objects you see in an **HTML** page in the browser window. Obvious objects include images and form elements. However, there may be other objects that aren't so obvious by looking at a page, but make perfect sense when you consider the HTML tags used to generate a page's content.

To help scripts control these objects - and to help authors see some method to the madness of potentially dozens of objects on a page - the browser makers define a document object model. A model is like a prototype or plan for the organization of objects in a page. Figure given below shows the document object model that Netscape has defined for Navigator 4. Internet Explorer contains almost all the objects as in figure given below At this stage of the learning process, it is not important to memorize every last object in the model, but rather to get a general feel for what's going on. One misconception you must avoid at the outset is that the model shown in figure is the model for every document that loads into the browser. On the contrary - it represents an idealized version of a document that includes one of every possible type of object that Navigator 4 knows about, hi a moment, I will show you how the document object model stored in the browser at any given instant reflects the HTML in the document. But for now, I want to impress an important aspect of the structure of the idealized model: its hierarchy.

MODEL HEIRARCHY

Notice in figure 9.4 that objects are grouped together in various levels designated by the density of the gray background. Objects are organized in a hierarchy, not unlike the hierarchy of a company's organization chart of job positions. At the top is the President. Reporting to the president are several vice-presidents. One of the vice presidents manages a sales force that is divided into geographical regions. Each region has a manager who reports to the vice president of sales; each region then has several salespeople. If the president wanted to communicate to a salesperson who handles a big account, the protocol would call for the president to route the message through the hierarchy - to the vice president of sales; to the sales manager, to the salesperson. The objects that generally represent the visible objects you see in an **HTML** page in the browser window. Obvious objects include images and form elements. However, there may be other objects that aren't so obvious by looking at a page, but make perfect sense when you consider the HTML tags used to generate a page's content.

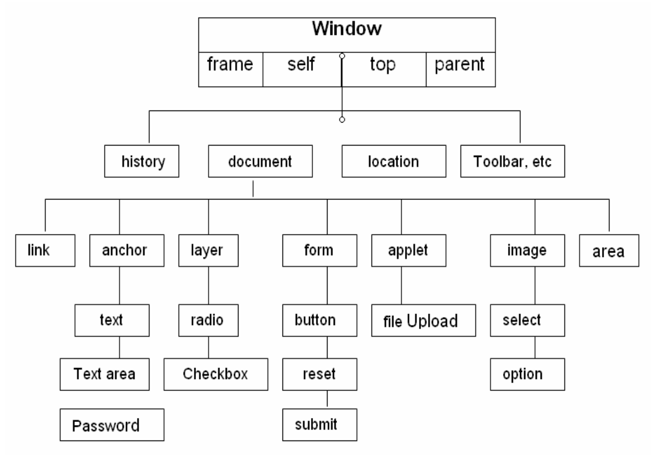


Figure shows. : Model Hierarchy-

Document Object-
Each HTML document that gets loaded into a window becomes a document object. Its position in the object hierarchy is an important one, as you can see in above mentioned figure. The document object contains by far the most other kinds of objects in the model. This makes perfect sense when you think about it: The document contains the content that you are likely to script.

Form object –
Users do not see the beginning and ending of forms on a page, only their elements. But a form is a distinct grouping of content inside an HTML document. Everything that is inside the <FORM>..... </FORM> tag set is part of the form object. A document might have more than one pair of <FORM> tags if that's what the page design calls for. If so, the map of the objects for that particular document would have two form objects instead of the one that appears in above mentioned figure.

Form elements-
Just as your HTML defines form elements within the confines of the <FORMx...</FQRM> tag pair, so does a form object contain all the elements defined for that object. Each one of those form elements - text fields, buttons, radio buttons, checkboxes and the like - is a separate object. Unlike the one of everything model shown in figure 9.4, the precise model for any document depends on the HTML tags in the document.

EVENT HANDLERS

As you must know that an event is something that happens sometime with somebody. For example your birthday, your parents' wedding anniversary are events for you. Sometimes, events may not be informed in advance means, they fall at random or unexpectedly. However, if the events occur, you have to handle anyway. The tools you use to handle the events are said to be event handlers.

The concept of events handlers are similar in JavaScript. The event handlers inform the browser what to do in case events occur. Like when somebody clicks the mouse button, when the page completes loading are the events for which we use handlers to tell how the browser will react. Event handlers are an important part of JavaScript.
Event handlers are associated with particular browser objects, and you specify the event handler in the tag that defines the object. For example, images and text links have an event **onMouseOver**, that happens when the mouse pointer moves over the object. You can write a simple event handler in this way –

<1MG SCR = "pic1.gif" onMouseOver = "highlight();">
Event handlers often used while creating web pages are being discussed as ahead-

onClick
A click event occurs when an object on a form is clicked. The onClick event handler executes JavaScript code when a click event occurs.
For checkboxes, links, radio buttons, reset buttons, and submit buttons, the onClick event handler can return false to cancel the action normally associated with a click event.
For example, the following code creates a hyperlink that, when clicked, displays a confirm dialog box. If the user clicks the hyperlink and then chooses cancel, the page specified by the hyperlink is not loaded.

```
<A HREF = "http://home.netscape.com/"
onClick="return confirm('Load Netscape home page?')">Netscape</A>
```

Returning false in an onClick event handler for a button has no effect.

Examples
Call a function when a user clicks a button. Suppose you have created a JavaScript function called compute. You can execute the compute function when the user clicks a button by calling the function in the onClick event handler, as follows:

```
<INPUT TYPE="button" VALUE="Calculate"
onClick="compute(this.form)">
```

In the preceding example, the keyword this refers to the current object; in this case, the Calculate button. The construct this.form refers to the form containing the button. For another example, suppose you have created a JavaScript function called **pickRandomURL** that lets you select a URL at random. You can use the onClick event handler of a link to specify a value for the HREF attribute of the <A> tag dynamically, as shown in the following example:

```
<A HREF=""
 onClick=="this.href=pickRandomURL()"
onMouseOver="window.status='Pick a random URL'; return true">
 Go!</A>
```

In the above example, the onMouseOver event handler specifies a custom message for the Navigator status bar when the user places the mouse pointer over the Go! anchor. As this example shows, you must return true to set the window, status property in the onMouseOver event handler.
Cancel the checking of a checkbox. The following example creates a checkbox with an onClick event handler. The event handler displays a confirm that warns the user that checking the checkbox purges all files. If the user chooses Cancel, the onClick event handler returns false and the checkbox is not checked.

```
<INPUTTYPE="checkbox" NAME="check1" VALUE="check1"
onClick="return confirm('This purges all your files. Are you sure?')"> Remove
files
```

onMouseOver
A mouseOver event occurs once each time the mouse pointer moves over an object or area from outside that object or area. The onMouseOver event handler executes JavaScript code when a mouseOver event occurs.
If the mouse moves from one area into another in a client-side image map, you'll get onMouseOut for the first area, then onMouseOver for the second.
Area objects that use the onMouseOver event handler must include the **HREF** attribute within the**<AREA>tag.**

You must return true within the event handler if you want to set the *status* or *defaultStatus* properties with the onMouseOver event handler.

Examples
By default, the HREF value of an anchor displays in the status bar at the bottom of the Navigator when a user places the mouse pointer over the anchor. In the following example, the onMouseOver event handler provides the custom message "Click this if you dare."

```
<A HREF="http://home.netscape.com/"
onMouseOver="window.status='Click this if you dare!';
return true">
 Click me</A>
```

onSubmit
A submit event occurs when a user submits a form. The onSubmit event handler executes JavaScript code when a submit event occurs.
You can use the onSubmit event handler to prevent a form from being submitted; to do so, put a **return** statement that returns false in the event handler. Any other returned value lets the form submit. If you omit the **return** statement, the form is submitted.

***Examples*:**
In the following example, the onSubmit event handler calls the validate function to evaluate the data being submitted. If the data is valid, the form is submitted; otherwise, the form is not submitted.

```
<FORM onSubmit="return validate(this)">
…
</FORM>
```

onFocus
A focus event occurs when a window, frame, or frameset receives focus or when a form element receives input focus. The focus event can result from a **focus** method or from the user clicking the mouse on an object or window or tabbing with the keyboard. Selecting within a field results in a select event, not a focus event. The onFocus event handler executes JavaScript code when a focus event occurs.

A frame's onFocus event handler overrides an onFocus event handler in the <BODY> tag of the document loaded into frame.

Note that placing an alert in an onFocus event handler results in recurrent alerts: when you press OK to dismiss the alert, the underlying window gains focus again and produces another focus event.

***Examples*:**

The following example uses an onFocus handler in the *valueFieldTextarea* object to call the **valueCheck** function.

```
<INPUTTYPE="textarea" VALUE="" NAME="valueField"
onFocus="valueCheck()">
```

onChange

A change event occurs when a select, text, or textarea field loses focus and its value has been modified. The onChange event handler executes JavaScript code when a change event occurs. Use the onChange event handler to validate data after it is modified by a user.

***Examples*:**

In the following example, *userName* is a text field. When a user changes the text and leaves the field, the onChange event handler calls the **checkValue** function to confirm that userName has a legal value.

```
<INPUTTYPE="text" VALUE="" NAME="userName"
        onChange="checkValue(this.value)">
```

onBlur

A blur event occurs when a form element loses focus or when a window or frame loses focus.The blur event can result from a **blur** method or from the user clicking the mouse on another object or window or tabbing with the keyboard. The onBlur event handler executes JavaScript code when a blur event occurs.

For windows, frames, and framesets, the onBlur event handler specifies JavaScript code to execute when a window loses focus.

A frame's onBlur event handler overrides an onBlur event handler in the <BODY> tag of the document loaded into frame.

***Examples*:**

Validate form input. In the following example, userName is a required text field. When a

user attempts to leave the field, the onBlur event handler calls the required function to confirm that userName has a legal value.

```
<INPUTTYPE="text" VALUE=”” NAME="userName"
onBlur="required(this.value)">
```

Change the background color of a window. In the following example, a window's onBlur and onFocus event handlers change the window's background color depending on whether the window has focus.

```
<BODY BGCOLOR="lightgrey"
  onBlur="document.bgColor='lightgrey'"
  onFocus="document.bgColor='antiquewhite'">
```

Change the background color of a frame. The following example creates four frames. The source for each frame, onblur2.html has the <BODY> tag with the onBlur and onFocus event handlers shown in Example 1. When the document loads, all frames are "lightgrey". When the user clicks a frame, the onFocus event handler changes the frame's background color to "antiquewhite". The frame that loses focus is changed to "lightgrey". Note that the onBlur and onFocus event handlers are within the <BODY> tag, not the <FRAME> tag.

```
<FRAMESET ROWS="50%,50%" COLS="40%,60%">
<FRAME SRC=onblur2.html NAME="frame1">
<FRAME SRC=onblur2.html NAME="frame2">
<FRAME SRC=onblur2.html NAME="frame3">
<FRAME SRC=onblur2.html NAME="frame4">
</FRAMESET>
```

The following code has the same effect as the previous code, but is implemented differently. The onFocus and onBlur event handlers are associated with the frame, not the document.

The onBlur and onFocus event handlers for the frame are specified by setting the ***onblur*** and ***onfocus*** properties. For information on using new to specify a string of JavaScript code to be compiled as a function, see the *Function* object.

```
      <SCRIPT>
      function setUpHandlers() {
       for (var i = 0; i < frames.length; i++) {
frames[i].onfocus=newFunction("document.bgColor='antiquewhite'
")          frames[i].onblur=new
Function("document.bgColor='lightgrey'")
      }
    }
    </SCRIPT>
    <FRAMESET ROWS="50%,50%" COLS="40%,60%"
onLoad=setUpHandlers()>
    <FRAME SRC=onblur2.html NAME="frame1">
    <FRAME SRC=onblur2.html NAME="frame2">
    <FRAME SRC=onblur2.html NAME="frame3">
    <FRAME SRC=onblur2.html NAME="frame4">
    </FRAMESET>
```

Close a window. In the following example, a window's onBlur event handler closes the window when the window loses focus.

```
   <BODY onBlur="window.close()">
    This is some text
   </BODY>
```

onLoad

A load event occurs when Navigator finishes loading a window or all frames within a <FRAMESET> tag. The onLoad event handler executes JavaScript code when a load event occurs.

Use the onLoad event handler within either the BODY or the <FRAMESET> tag, for example, **<BODY onLoad="...">.**

In a FRAMESET and FRAME relationship, an onLoad event within a frame (placed in the <BODY> tag) occurs before an onLoad event within the FRAMESET (placed in the **<FRAMESET>tag)**.

For images, the onLoad event handler indicates the script to execute when an image is displayed. Do not confuse displaying an image with loading an image. You can load several images, then display them one by one in the same *Image* object by setting the object's *src* property. If you change the image displayed in this way, the onLoad event handler executes every time an image is displayed, not just when the image is loaded into memory.

If you specify an onLoad event handler for an *Image* object that displays a looping GIF animation (multi-image GIF), each loop of the animation triggers the onLoad event, and the event handler executes once for each loop. You can use the onLoad event handler to create a JavaScript animation by repeatedly setting the *src* property of an *Image* object.

***Examples*:**

Display message when page loads. In the following example, the onLoad event handler displays a greeting message after a Web page is loaded.

```
<BODY onLoad="window.alert("Welcome to the Brave New World
home page!")>
```

Display alert when image loads. The following example creates two *Image* objects, one with the Image() constructor and one with the <JMG> tag. Each *Image* object has an onLoad event handler that calls the *displayAlert* function, which displays an alert. For the image created with the <IMG> tag, the alert displays the image name. For the image created with the Image() constructor, the alert displays a message without the image name. This is because the onLoad handler for an object created with the Image() constructor must be the name of a function, and it cannot specify parameters for the *displayAlert* function.

```
        <SCRIPT>
  imageA = new lmage(50,50)
  imageA.onload=displayAlert
  imageA.src="cyanball.gif "
  function displayAlert(thelmage) {
     if (thelmage==null) {
       alert('An image loaded')
     }
    else alert(thelmage.name + ' has been loaded.')
  }
  </SCRIPT>
  <IMG NAME="imageB" SRC="greenball.gif" ALIGN="top"
     onLoad=displayAlert(this)><BR>
```

Looping GIF animation: The following example displays an image, birdie.gif, that is a looping GIF animation. The onLoad event handler for the image increments the variable cycles, which keeps track of the number of times the animation has looped. To see the value of cycles, the user clicks the button labeled Count Loops.

```
      <SCRIPT>
      var cycles=0
        </SCRIPT>
    <IMG ALIGN="top" SRC="birdie.gif BORDER=0
        onload="++cycles">
    <INPUTTYPE="button" VALUE="Count Loops"
     onClick="alert(The animation has looped ' + cycles + ' times.')">
```

Change **GIF** animation displayed. The following example uses an **onLoad** event handler to rotate the display of six GIF animations. Each animation is displayed in sequence in one Image object. When the document loads, !anim0.html is displayed. When that animation completes, the onLoad event handler causes the next file, laniml .html, to load in place of the first file. After the last animation, !anim5.html, completes, the first file is again displayed. Notice that the *changeAnimation* function does not call itself after changing the src property of the Image object. This is because when the src property changes, the image's onLoad event handler is triggered and the *changeAnimation* function is called.

```
          <SCRIPT>
            var whichlmage=0
           var maxlmages=5
           function changeAnimation(thelmage) {
              ++whichlmage
    if (whichlmage <= maxlmages) {
    var imageName="!anim" + whichlmage + ".gif
    thelmage.src=imageName
    } else {
          whichlmage=-1
          return
          }
    }
    </SCRIPT>
    <IMG NAME="changingAnimation" SRC="!anim0.gif BORDER=0
ALIGN="top"
    onLoad="changeAnimation(this)">
```

onUnload

An unload event occurs when you exit a document. The **onUnload** event handler executes JavaScript code when an unload event occurs. Use the **onUnload** event handler within either the BODY or the **<FRAMESET> tag**, for example, **<BODY onUnload="...">.**

In a frameset and frame relationship, an onUnload event within a frame (placed in the <BOD Y> tag) occurs before an onUnload event within the frameset (placed in the <FRAMESET> tag).

***Examples*:**

In the following example, the onUnload event handler calls the **cleanUp** function to perform some shutdown processing when the user exits a Web page:

```
            <BODY onUnload="cleanUp()">
```

ALERT, CONFIRM AND PROMPT DIALOG BOXES

There are three methods for displaying messages and interacting with the users. They are alert, confirm and prompt. The alert method displays an alert dialog box that gives a message to the user. The confirm method displays a confirmation box that contains OK and Cancel buttons normally. The prompt method displays a message and prompts the user for input.

alert

Displays an Alert dialog box with a message and an **OK** button.

Syntax

Alert("*message*")

Parameters

message is any string or a property of an existing object.

Description

An alert dialog box looks. Use the **alert** method to display a message that does not require a user decision. The message argument specifies a message that the dialog box contains.

Although **alert** is a method of the *window* object, you do not need to specify a *window Reference* when you call it. For example, window Reference.alert() is unnecessary. You cannot specify a title for an alert dialog box, but you can use the open method to create your own "alert" dialog.

***Examples*:**

In the following example, the **testValue** function checks the name entered by a user in the Text object of a form to make sure that it is no more than eight characters hi length.

This example uses the **alert** method to prompt the user to enter a valid value.

```
  function testValue(textElement) {
    if (textElement. length > 8) {
      alert("Please enter a name that is 8 characters or less")
    }
  }
```

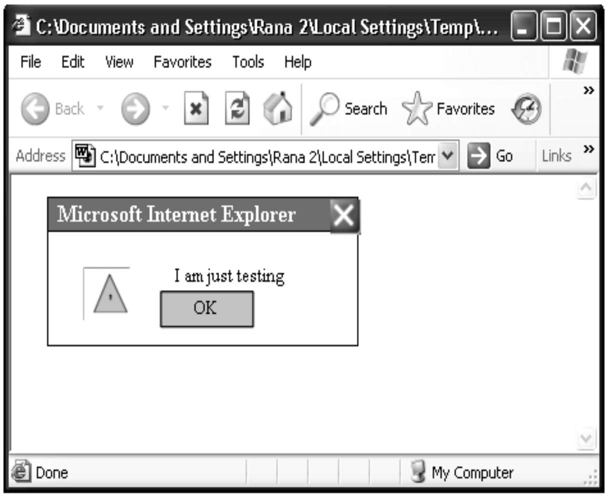
You can call the testValue function in the onBlur event handler of a form's Text object, as shown in the following example:

```
            Name: <INPUT TYPE="text" NAME="userName"
              onBlur="testValue(username.value)">
```

SCRIPT FOR CREATING AN ALERT DIALOG BOX

```
<HTML>
   <HEAD>
   <TITLE>
   Using An Alert Dialog Box
   </TITLE>
   </HEAD>
<BODY>
   <Script Language = "JavaScript">
    window.alert("I am just testing alert")
</SCRIPT>
</BODY>
 </HTML>
```

The result of the above program on the browser will look like one in figure given below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\...

Address C:\Documents and Settings\Rana 2\Local Settings\Tem

Microsoft Internet Explorer

I am just testing

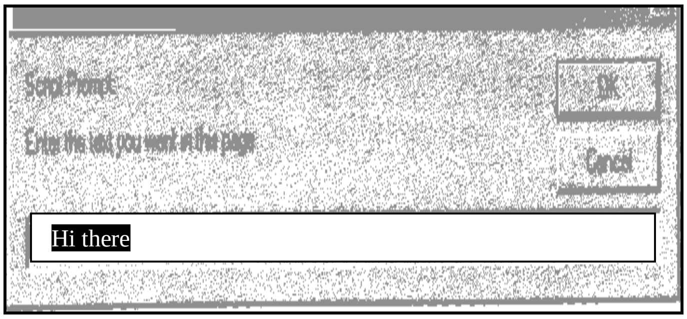
Prompt:

Displays a Prompt dialog box with a message and an input field.
Syntax: prompt(*message*, [*inputDefault*])

Parameters

message is any string or a property of an existing object; the string is displayed as the message. *inputDefault* is a string, integer, or property of an existing object that represents the default value of the input field.

Description
A prompt dialog box looks as follows:


Script Prompt:

Enter the text you want in the page

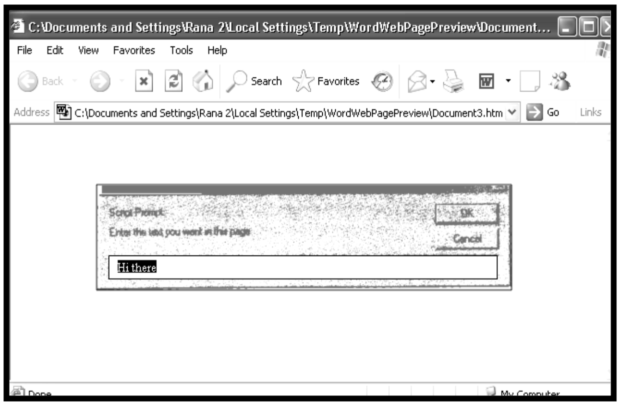
Hi there

Use the **prompt** method to display a dialog box that receives user input. If you do not specify an initial value for *inputDefault*, the dialog box displays "<undefined>." Although prompt is a method of the window object, you do not need to specify a window Reference when you call it. For example, *windowReference*.prompt() is unnecessary.
***Examples*:** prompt("Enter the number of cookies you want to order:", 12)

SCRIPT FOR CREATING A PROMPT DIALOG BOX

```
<HTML>
   <HEAD>
<TITLE>
     Using An Alert Dialog Box
    <TITLE>
   </HEAD>
<BODY>
<Script Language = "JavaScript">
var text = prompt("Enter the text you want in this page",
             "Hi there!")
if (text =="") {
alert ("You didn't enter anything.")
}
    else {
         document.writeln("<CENTER><H1>" +
text+"</H1></CENTER>")
     }
</SCRIPT>
</BODY>
 </HTML>
```

The result of the above program on the browser will look like one in figure below:


C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Document...

Address C:\Documents and Settings\Rana 2\Local Settings\Temp\WordWebPagePreview\Document3.htm

Script Prompt:

Enter the text you want in the page

Hi there

SOME SCRIPTS FOR PRACTICE IN JAVASCRIPT
In the following section, we are citing some simple programs for the students to learn how to develop them. Just write the programs and execute them on the browser to find how they look like.

WRITING TEXT WITH JAVASCRIPT

```
<html>
     <body>
          <script language="javaScript">
 document.write("Hello World!")
             </script>
    </body>
</html>
```

```
<html>
    <head>
        <script language="javaScript”>
        function message()
        {
        alert('This alert box was called with the onload event")
        }
        </script>
    </head>
    <body onload="message()">
    </body>
</html>
```

JAVASCRIPT IN THE BODY SECTION

```
<html>
    <head>
    </head>
        <body>
        <script language"javaScript”>
 document.write(“This message is written when the page loads")
        </script>
        </body>
</html>
```

CALLING A FUNCTION

```
<html>
    <head>
    <script language="javaScript">
    function myfunction()
    {
    alert("HELLO")
    }
   </script>
   </head>
        <body>
        <form>
        <input type="button"
         onclick="myfunction()"
         value="Call function">
        </form>
        <p>By pressing the button, a function will be called. The
function will alert a
         message.</p>
        </body>
</html>
```

FOR LOOP

```
            <html>
            <head>
           <title>For Loop</title>
            <head>
            <body>
            <script language="javaScript” >
            for(i = 0;i <= 5;i++)
            {
            document.write(“The number is" + i)
            document.write("<br>")
            }
            </script>
            </body>
            </html>
```

JAVASCRIPT IN THE HEAD SECTION

WHILE LOOP

```
<html>
<body>
<script language="javaScript” >
i = 0
while (i <= 5){
document.write('The number is"+i)
document. write("<br>")
i++
}
</script>
</body>
<html>
```

DISPLAYING MESSAGE ON SCROLLBAR

```
</html>
<head><title>Scrolling message Example<?title>
<script language = "javaScript">
pos = 0;
function ScrollMessage() {
pos++;
if (pos > msg.length) pos = 0;
window.setTimeout("ScrollMessage()" ,200);
}
ScrollMessage();
</script>
 </head>
 <body>
<center>
<Marquee><h1> Have a Nice Day<h1></Marquee>
</center>
</body>
</html>
```

INSERTING WISH ACCORDING TO TIME

```
<html>
<head>
<script language="javaScript">
   <!-- Hide this code from non JavaScript browsers
currentTime = new Date();
if (currentTime.getHours() < 12)
      document.write("Good morning");
else if (currentTime.getHours() < 17)
     document.write("Good afternoon");
else document.write("Good evening");
// End of javaScript code -->
</script>
</head>
</html>
```