

Class: - BCA (Semester IV)

Subject: - WDA

QUESTION BANK

HTML

HTML is made up of elements (often called tags) that build the contents of a web page. The differences between HTML and other programming languages include:

- HTML is not compiled. It is written and used without any changes being done to it. I start out a text file and is still a text file when a browser or user agent interprets it.
- HTML is human readable. While some other programming languages can be read by people (and not just machines), many times you have to learn the language to really understand it. In comparison, most HTML beginners can at least guess what an tag does, for example.

HTML is made up of elements with attributes, some of the most common ones you would see include:

- <p> for paragraphs
- <a> for links
- <div> for dividing up sections of a page

The basic structure for all HTML documents is simple and should include the following minimum **elements or tags**:

- <html> - The main container for HTML pages
- <head> - The container for page header information
- <title> - The title of the page
- <body> - The main body of the page

Remember that before an opening <html> tag, an XHTML (eXtensible HyperText Markup Language.) document can contain the optional XML declaration, and it should always contain a DOCTYPE declaration indicating which version of XHTML it uses.

Now we will explain each of these tags one by one. In this tutorial you will find the terms element and tag are used interchangeably.

HTML Basic Syntax:

- HTML Element names and attribute names are not case sensitive.
- HTML Documents start with a <!doctype...> statement, followed by a *header* and a text body all enclosed in <html>...</html>.
- HTML Header is enclosed in <head>....</head> tags.
- HTML Body is enclosed in <body>....</body> tags.
- HTML Comments are written as <!-- A comment -->.

HTML Basic Document:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4.0//EN">
<html>
<head>
```

```
<title>Document Title like HTML Tutorial</title>
</head>
<body>
  Document Text with other tags will come here.
</body>
</html>
```

HTML Versions

There are a number of versions of HTML that have been supported by web browsers:

- **HTML2.0**
This was published as an IETF RFC in 1995. It was supported by some browsers like Mosaic.
- **HTML3.2**
this version was the first W3C recommendation. It had wider browser support (partially because there were a lot more browsers). It became a recommendation in January 1997.
- **HTML4.0andHTML4.01**
In December 1997, the W3C released this upgrade to HTML 3.2 as a recommendation. It added more features and tags and provided three versions: transitional, strict, and frameset. This was updated to 4.01 in December 1999 with a few changes to the specification.
- **XHTML 1.0**
XHTML 1.0 is a reformulation of HTML 4.01 under XML rules, and it was published as a recommendation by the W3C in January 2000. It has much stricter syntax and requires that any XHTML be valid and well-formed in order to display correctly. Most web browsers render XHTML 1.0 documents the same as they render HTML 4.01 documents.
- **HTML5**
HTML5 began being developed in 2004, when the W3C HTML working group decided to merge the HTML and XHTML tree to make HTML a purely XML-based language. This left designers and browser manufacturers who wanted a more flexible solution with the choice to give up or create their own new specification. They created a new group called the Web Hypertext Application Technology Working Group or WHATWG. HTML5 became a W3C working draft in 2008.

HTML Elements

1. The <html> Element:

The <html> element is the containing element for the whole HTML document. Each HTML document should have one <html> and each document should end with a closing </html> tag.

Following two elements appear as direct children of an <html> element:

- <head>
- <body>

As such, start and end HTML tags enclose all the other HTML tags you use to describe the Web page.

2. The <head> Element:

The <head> element is just a container for all other header elements. It should be the first thing to appear after the opening <html> tag.

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

Each <head> element should contain a <title> element indicating the title of the document, although it may also contain any combination of the following elements, in any order:

- The <base> tag is used to create a "base" url for all links on the page. Check HTML Base tag.
- The <object> tag is designed to include images, JavaScript objects, Flash animations, MP3 files, QuickTime movies and other components of a page. Check HTML Object tag.
- The <link> tag is used to link to an external file, such as a style sheet or JavaScript file. Check HTML Link tag.
- The <style> tag is used to include CSS rules inside the document. Check HTML Style tag.
- The <script> tag is used to include JAVAScript or VBScript inside the document. Check HTML Script tag.
- The <meta> tag includes information about the document such as keywords and a description, which are particularly helpful for search applications. Check HTML Meta tag.

Example:

```
<head>
<title>HTML Basic tags</title>
<meta name="Keywords" content="HTML, Web Pages" />
<meta name="description" content="HTML Basic Tags" />
<base href="http://www.tutorialspoint.com" />
<link rel="stylesheet" type="text/css" href="tp.css" />
<script type="text/javascript">
</script>
</head>
```

3. The <title> Element:

You should specify a title for every page that you write inside the <title> element. This element is a child of the <head> element). It is used in several ways:

- It displays at the very top of a browser window.
- It is used as the default name for a bookmark in browsers such as IE and Netscape.
- It is used by search engines that use its content to help index pages.

Therefore it is important to use a title that really describes the content of your site. The <title> element should contain only the text for the title and it may not contain any other elements.

Example:

```
<head>
<title>HTML Basic tags</title>
</head>
```

4. The <body> Element:

The <body> element appears after the <head> element and contains the part of the Web page that you actually see in the main browser window, which is sometimes referred to as body content. A <body> element may contain anything from a couple of paragraphs under a heading to more complicated layouts containing forms and tables. Most of what you will be learning in

this and the following five chapters will be written between the opening <body> tag and closing </body> tag.

Example:-

```
<body>
  <p>this is a paragraph tag. </p>
</body>
```

HTML Attributes

Attributes are another important part of HTML markup. An attribute is used to define the characteristics of an element and is placed inside the element's opening tag. All attributes are made up of two parts: a name and a value:

- The *name* is the property you want to set. For example, the element in the example carries an attribute whose name is *face*, which you can use to indicate which typeface you want the text to appear in.
- The *value* is what you want the value of the property to be. The first example was supposed to use the Arial typeface, so the value of the *face* attribute is Arial.

The value of the attribute should be put in double quotation marks, and is separated from the name by the equals sign. You can see that a color for the text has been specified as well as the typeface in this element:

```
<font face="arial" color="#CC0000">
```

Any HTML tags have a unique set of their own attributes. These will be discussed as each tag is introduced throughout the tutorial. Right now we want to focus on a set of generic attributes that can be used with just about every HTML Tag in existence.

The **four core attributes** that can be used on the majority of HTML elements (although not all) are:

- **id**
- **title**
- **class**
- **style**

1. The id Attribute:

The *id* attribute can be used to uniquely identify any element within a page (or style sheet). There are two primary reasons that you might want to use an id attribute on an element:

- If an element carries an id attribute as a unique identifier it is possible to identify just that element and its content.
- If you have two elements of the same name within a Web page (or style sheet), you can use the id attribute to distinguish between elements that have the same name.

We will discuss style sheet in separate tutorial. For now, the id attribute could be used to distinguish between two paragraph elements, like so:

```
<p id="html">This para explains what is HTML</p>
<p id="css">This para explains what is Cascading Style Sheet</p>
```

Note that there are some special rules for the value of the id attribute, it must:

- Begin with a letter (A.Z or a.z) and can then be followed by any number of letters, digits (0.9), hyphens, underscores, colons, and periods.
- Remain unique within that document; no two attributes may have the same value within that HTML document.

2. The title Attribute:

The *title* attribute gives a suggested title for the element. The syntax for the *title* attribute is similar as explained for *id* attribute:

The behavior of this attribute will depend upon the element that carries it, although it is often displayed as a tooltip or while the element is loading.

For example: `<h4 title="Hello HTML!">Titled Heading Tag Example</h4>`

3. The class Attribute:

The *class* attribute is used to associate an element with a style sheet, and specifies the class of element. You learn more about the use of the class attribute when you will learn Cascading Style Sheet (CSS). So for now you can avoid it.

The value of the attribute may also be a space-separated list of class names.

For example: `class="className1 className2 className3"`

4. The style Attribute:

The style attribute allows you to specify CSS rules within the element.

For example: `<p style="font-family:arial; color:#FF0000;">Some text...</p>`

HTML Formatting Tags

If you want people to read what you have written, then structuring your text well is even more important on the Web than when writing for print. People have trouble reading wide, long, paragraphs of text on Web sites unless they are broken up well.

This section will teach you basic text formatting elements like heading elements and paragraph elements.

Create Headings - The <h> Elements:

Any document starts with a heading. You use different sizes for your headings. HTML also has six levels of headings, which use the elements `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, and `<h6>`. While displaying any heading, browser adds one line before and after that heading.

Example:

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
```

Answer:-

This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

Create Paragraph - The <p> Element:

The <p> element offers a way to structure your text. Each paragraph of text should go in between an opening <p> and closing </p> tag as shown below in the example:

<p>Here is a paragraph of text.</p>

<p>Here is a second paragraph of text.</p>

<p>Here is a third paragraph of text.</p>

Result:

Here is a paragraph of text.

Here is a second paragraph of text.

Here is a third paragraph of text.

**Create Line Breaks - The
 Element:**

Whenever you use the
 element, anything following it starts on the next line. This tag is an example of an **empty** element, where you do not need opening and closing tags, as there is nothing to go in between them.

Example:

Hello

You come most carefully upon your hour.

Thanks

Mahnaz

Result:

Hello

You come most carefully upon your hour.

Thanks

Mahnaz

Centering Content - The <center> Element:

You can use <center> tag to put any content in the center of the page or any table cell.

Example:

<p>This is not in the center.</p>

<center>

<p>This is in the center.</p>

</center>

Result:

This is not in the center.

This is in the center.

Horizontal Rules - The <hr /> Element

Horizontal rules are used to visually break up sections of a document. The <hr> tag creates a line from the current position in the document to the right margin and breaks the line accordingly.

For example you may want to give a line between two paragraphs as follows:

<p>This is paragraph one and should be on top</p>

<hr />

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

<p>This is paragraph two and should be at bottom</p>

Result: This is paragraph one and should be on top

This is paragraph two and should be at bottom

Again <hr /> tag is an example of an empty element, where you do not need opening and closing tags, as there is nothing to go in between them.

Bold Text - The Element:

Anything that appears in a ... element is displayed in bold, like the word bold here:

<p>The following word uses a bold typeface.</p>

Result: The following word uses a bold typeface.

Italic Text - The <i> Element:

Anything that appears in a <i>...</i> element is displayed in italicized, like the word italicized here:

<p>The following word uses a <i>italicized</i> typeface.</p>

Result: The following word uses a italicized typeface.

Underlined Text - The <u> Element:

Anything that appears in a <u>...</u> element is displayed with underline, like the word underlined here:

<p>The following word uses a <u>underlined</u> typeface.</p>

Result: The following word uses a underlined typeface.

Strike Text - The <strike> Element:

Anything that appears in a <strike>...</strike> element is displayed with strikethrough, which is a thin line through the text:

<p>The following word uses a <strike>strikethrough</strike> typeface.</p>

Result: The following word uses a strikethrough typeface.

Superscript Text - The <sup> Element:

The content of a <sup> element is written in superscript; the font size used is the same size as the characters surrounding it but is displayed half a characters height above the other characters.

<p>The following word uses a ^{superscript} typeface.</p>

Result: The following word uses a superscript typeface.

Subscript Text - The <sub> Element:

The content of a <sub> element is written in subscript; the font size used is the same as the characters surrounding it, but is displayed half a characters height beneath the other characters.

Ex: - <p>The following word uses a _{subscript} typeface. </p>

Result: The following word uses a subscript typeface.

Larger Text - The <big> Element:

The content of the <big> element is displayed one font size larger than the rest of the text surrounding it.

Ex:- <p>The following word uses a <big>big</big> typeface.</p>

Result: The following word uses a big typeface.

Smaller Text - The <small> Element:

The content of the <small> element is displayed one font size smaller than the rest of the text surrounding it. **Ex:-** <p>The following word uses a <small>small</small> typeface.</p>

Result: The following word uses a small typeface.

Emphasized Text - The Element:

The content of an element is intended to be a point of emphasis in your document, and it is usually displayed in italicized text. The kind of emphasis intended is on words such as "must" in the following sentence:

<p>You must remember to close elements in XHTML.</p>

Result: You *must* remember to close elements in XHTML.

Strong Text - The Element:

The element is intended to show strong emphasis for its content; stronger emphasis than the element. As with the element, the element should be used only when you want to add strong emphasis to part of a document.

<p>You must remember to close elements in XHTML.</p>

Result: You **must** remember to close elements in XHTML.

Short Quotations - The <q> Element :

The <q> element is intended to be used when you want to add a quote within a sentence rather than as an indented block on its own.

<p>Amit is in Spain, <q>He is there at my home. I think I am wrong</q>.</p>

Result: Amit is in Spain, He is their at my home. I think I am wrong.

Block and Inline Elements:

We can categories all the elements into two sections:

- **Block-level elements** - Block-level elements appear on the screen as if they have a carriage return or line break before and after them. For example the <p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, , , <dl>, <pre>, <hr />, <blockquote>, and <address> elements are all block level elements. They all start on their own new line, and anything that follows them appears on its own new line.
- **Inline elements** - Inline elements, on the other hand, can appear within sentences and do not have to appear on a new line of their own. The , <i>, <u>, , , <sup>, <sub>, <big>, <small>, , <ins>, , <code>, <cite>, <dfn>, <kbd>, and <var> elements are all inline elements.

HTML Marquees Tag:-

A HTML marquee is a scrolling piece of text displayed either horizontally across or vertically down your web site page depending on the settings. This is created by using HTML tag <marquee>.

Syntax:

A simple syntax to use marquee is as follows:

<marquee attribute_name="attribute_value"....more attributes>

One or more lines or text message or image

</marquee>

Attributes:

A HTML marquee can have following attributes:

- **width:** how wide the marquee is. This will have a value like 10 or 20% etc.
- **height:** how tall the marquee is. This will have a value like 10 or 20% etc.
- **direction:** which direction the marquee should scroll. This will have value either *up, down, left* or *right*.
- **behavior:** what type of scrolling. This will have value *scroll, slid* and *alternate*.
- **scrolldelay:** how long to delay between each jump. This will have a value like 10 etc.
- **scrollamount:** how far to jump. This will have a value like 10 etc.
- **loop:** how many times to loop. The default value is INFINITE, which means that the marquee loops endlessly.
- **bgcolor:** background color. This will have any color name or color hex value.
- **hspace:** horizontal space around the marquee. This will have a value like 10 or 20% etc.
- **vspace:** vertical space around the marquee. This will have a value like 10 or 20% etc.

Examples: <marquee>This is basic example of marquee</marquee>

Example: <marquee direction="up">This text will scroll from bottom to up</marquee>

HTML Images

Images are very important to beautify as well as to depicts many concepts on your web page. Its is true that one single image is worth than thousands of words. So as a Web Developer you should have clear understanding on how to use images in your web pages.

Insert Image - The Element:

You will insert any image in your web page by using tag. Following is the simple syntax to use this tag.

Image Attributes:

Following are most frequently used attributes for tag.

- **width:** sets width of the image. This will have a value like 10 or 20% etc.
- **height:** sets height of the image. This will have a value like 10 or 20% etc.
- **border:** sets a border around the image. This will have a value like 1 or 2 etc.
- **src:** specifies URL of the image file.
- **alt:** this is an alternate text which will be displayed if image is missing.
- **align:** this sets horizontal alignment of the image and takes value either *left, right* or *center*.
- **valign:** this sets vertical alignment of the image and takes value either *top, bottom* or *center*.
- **hspace:** horizontal space around the image. This will have a value like 10 or 20% etc.
- **vspace:** vertical space around the image. This will have a value like 10 or 20% etc.
- **name:** name of the image with in the document.
- **id:** id of the image with in the document.
- **style:** this will be used if you are using CSS.
- **title:** specifies a text title. The browser, perhaps flashing the title when the mouse passes over the link.

- **ismap and usemap:** These attributes for the tag tell the browser that the image is a special mouse-selectable visual map of one or more hyperlinks, commonly known as an **image map**. We will see how to use these attributes in [Image Links](#) chapter.

Example: - Image Attributes - width, height, title, border and align:

```

```

Example: ``

Which image format is suitable for you?

The images in Graphics Interchange Format - **GIF** format are best used for banners, clip art, and buttons. The main reason for this is that gifs can have a transparent background which is priceless when it comes to web design. On the down side, gifs are usually larger files, not as compressed as a jpeg, which calls for slow load times and large transfer rates. Gifs are also limited to the 256 color scheme.

Ths images in Joint Photographic Experts Group - **JPEG** format have an unlimited color wheel, and have a high compression rate downsizing your load times and saving hard drive space. JPEGs don't allow for transparent backgrounds, but their size/quality ratio is outstanding. Its best to use JPEG format for photo galleries, or artwork to allow the viewer to catch that extra bit of detail. Avoid Jpegs for graphical design, stick to using them for thumbnails and backgrounds.

The images in Portable Network Graphics - **PNG** format are an extensible file format for the lossless, portable, well-compressed storage of raster images. PNG provides a patent-free replacement for GIF and can also replace many common uses of TIFF. Indexed-color, grayscale, and true color images are supported, plus an optional alpha channel. Sample depths range from 1 to 16 bits. PNG also compresses better than GIF in almost every case (5% to 25% in typical cases).

Linking Documents - The <a> Element:

A link is specified using the <a> element. This element is called **anchor tag** as well. Anything between the opening <a> tag and the closing tag becomes part of the link and a user can click that part to reach to the linked document.

Following is the simple syntax to use this tag.

```
<a href="Document URL" attr_name="attr_value"...more attributes />
```

Anchor Attributes:

Following are most frequently used attributes for <a> tag.

- **href:** specifies the URL of the target of a hyperlink. Its value is any valid document URL, absolute or relative, including a fragment identifier or a JavaScript code fragment.
- **target:** specify where to display the contents of a selected hyperlink. If set to "_blank" then a new window will be opened to display the loaded page, if set to "_top" or "_parent" then same window will be used to display the loaded document, if set to "_self" then loads the new page in current window. By default its "_self".
- **name & id:** attributes places a label within a document. When that label is used in a link to that document, it is the equivalent of telling the browser to goto that label.
- **event:** attributes like *onClick*, *onMouseOver* etc. are used to trigger any Javascript or VBscript code.

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

- **title:** attribute lets you specify a title for the document to which you are linking. The value of the attribute is any string, enclosed in quotation marks. The browser might use it when displaying the link, perhaps flashing the title when the mouse passes over the link.
- **accesskey:** attribute provides a keyboard shortcut that can be used to activate a link. For example, you could make the T key an access key so that when the user presses either the Alt or Ctrl key on his keyboard (depending on his operating system) along with the T key, the link gets activated.

A Simple Example:

```
<a href="http://www.tutorialspoint.com/" target="_blank" >TP Home</a>
|
<a href="http://www.amrood.com/" target="_self" >AMROOD Home</a>
|
<a href="http://www.change-images.com/" target="_top" >Change Images
Home</a>
```

HTML Email Tag:

HTML <a> tag provides you facility to specify an email address to send an email. While using <a> tag as an email tag then you will use **mailto:email address** along with *href* attribute. Following is the syntax of using mailto instead of using http.

```
<a href= "mailto:abc@example.com">Send Email</a>
```

This code will generate following link: Send Email

HTML Tables

Tables are very useful to arrange in HTML and they are used very frequently by almost all web developers. Tables are just like spreadsheets and they are made up of rows and columns.

You will create a table in HTML/XHTML by using <table> tag. Inside <table> element the table is written out row by row. A row is contained inside a <tr> tag . which stands for table row. And each cell is then written inside the row element using a <td> tag . which stands for table data.

Example:

```
<table border="1">
<tr>
<td>Row 1, Column 1</td>
<td>Row 1, Column 2</td>
</tr>
<tr>
<td>Row 2, Column 1</td>
<td>Row 2, Column 2</td>
</tr>
</table>
```

Result:

Row 1, Column 1	Row 1, Column 2
Row 2, Column 1	Row 2, Column 2

Table Heading - The <th> Element:

Table heading can be defined using <th> element. This tag will be put to replace <td> tag which is used to represent actual data. Normally you will put your top row as table heading as shown below, otherwise you can use <th> element at any place:

```
<table border="1">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
</tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
```

This will produce following result. You can see its making heading as a bold one:

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

Table Cellpadding and Cellspacing:

There are two attributes called *cellpadding* and *cellspacing* which you will use to adjust the white space in your table cell. Cellspacing defines the width of the border, while cellpadding represents the distance between cell borders and the content within. Following is the example:

```
<table border="1" cellpadding="5" cellspacing="5">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Ramesh Raman</td>
<td>5000</td>
</tr>
<tr>
<td>Shabbir Hussein</td>
<td>7000</td>
</tr>
</table>
```

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

This will produce following result:

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

Colspan and Rowspan Attributes:

You will use *colspan* attribute if you want to merge two or more columns into a single column. Similar way you will use *rowspan* if you want to merge two or more rows. Following is the **example**:

```
<table border="1">
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
<tr><td rowspan="2">Row 1 Cell 1</td>
<td>Row 1 Cell 2</td><td>Row 1 Cell 3</td></tr>
<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
<tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>
```

This will produce following result:

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

Tables Backgrounds

You can set table background using of the following two ways:

- Using *bgcolor* attribute - You can set background color for whole table or just for one cell.
- Using *background* attribute - You can set background image for whole table or just for one cell.

NOTE: You can set border color also using *bordercolor* attribute.

Here is an **example** of using *bgcolor* attribute:

```
<table border="5" bordercolor="green" bgcolor="gray">
<tr>
```

```
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
<tr><td rowspan="2">Row 1 Cell 1</td>
<td bgcolor="red">Row 1 Cell 2</td><td>Row 1 Cell 3</td></tr>
<tr><td>Row 2 Cell 2</td><td>Row 2 Cell 3</td></tr>
<tr><td colspan="3">Row 3 Cell 1</td></tr>
</table>
```

This will produce following result:

Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

Table height and width:

You can set a table width and height using *width* and *height* attributes. You can specify table width or height in terms of integer value or in terms of percentage of available screen area. Following is the example:

```
<table border="1" width="400" height="150">
<tr>
<td>Row 1, Column 1</td>
<td>Row 1, Column 2</td>
</tr>
<tr>
<td>Row 2, Column 1</td>
<td>Row 2, Column 2</td>
</tr>
</table>
```

This will produce following result:

	Row 1, Column 1	Row 1, Column 2
	Row 2, Column 1	Row 2, Column 2

Nested Tables:

You can use one table inside another table. Not only tables you can use almost all the tags inside table data tag <td>.

Following is the example of using another table and other tags inside a table cell.

```
<table border="1">
<tr>
<td>
      <table border="1">
        <tr>
          <th>Name</th>
          <th>Salary</th>
        </tr>
        <tr>
          <td>Ramesh Raman</td>
          <td>5000</td>
        </tr>
        <tr>
          <td>Shabbir Hussein</td>
          <td>7000</td>
        </tr>
      </table>
    </td>
    <td>
      <ul>
        <li>This is another cell</li>
        <li>Using list inside this cell</li>
      </ul>
    </td>
  </tr>
  <tr>
    <td>Row 2, Column 1</td>
    <td>Row 2, Column 2</td>
  </tr>
</table>
```

This will produce following result:

Name	Salary	This is another cell
Ramesh Raman	5000	Using list inside this cell
Shabbir Hussein	7000	
Row 2, Column 1	Row 2, Column 2	

HTML Frames

Frames divide a browser window into several pieces or panes, each pane containing a separate XHTML/HTML document. One of the key advantages that frames offer is that you can then load and reload single panes without having to reload the entire contents of the browser window. A collection of frames in the browser window is known as a frameset.

The window is divided up into frames in a similar pattern to the way tables are organized: into rows and columns. The simplest of framesets might just divide the screen into two rows, while a complex frameset could use several rows and columns.

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

There are few drawbacks also you should be aware of with frames are as follows:

- Some browsers do not print well from framesets.
- Some smaller devices cannot cope with frames, often because their screen is not big enough to be divided up.
- Some time your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back button* might not work as the user hopes.
- There are still few browsers who do not support frame technology.

To create a frameset document, first you need the `<frameset>` element, which is used instead of the `<body>` element. The frameset defines the rows and columns your page is divided into, which in turn specify where each individual frame will go. Each frame is then represented by a `<frame>` element.

You also need to learn the `<noframes>` element, which provides a message for users whose browsers do not support frames.

Now we will discuss these tags in detail one by one.

Creating Frames - The `<frameset>` Element:

- The `<frameset>` tag replaces the `<body>` element in frameset documents.
- The `<frameset>` tag defines how to divide the window into frames.
- Each frameset defines a set of rows **or** columns. If you define frames by using rows then horizontal frames are created. If you define frames by using columns then vertical frames are created.
- The values of the rows/columns indicate the amount of screen area each row/column will occupy.
- Each frame is indicated by `<frame>` tag and it defines what HTML document to put into the frame.

Example:

Following is the example to create three horizontal frames:

```
<html>
<head>
<title>Frames example</title>
</head>
  <frameset rows="10%,80%,10%">
    <frame src="/html/top_frame.htm" />
    <frame src="/html/main_frame.htm" />
    <frame src="/html/bottom_frame.htm" />
  </frameset>
  <noframes>
    <body>
      Your browser does not support frames.
    </body>
  </noframes>
</html>
```

Now create three HTML files called *top_frame.htm*, *main_frame.htm* and *bottom_frame.htm* to be loaded into three frames with some content.

The `<frameset>` Element Attributes:

Following are important attributes of `<frameset>` and should be known to you to use frameset.

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

- **cols:** specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of four ways:
 - Absolute values in pixels. For example to create three vertical frames, `usecols="100, 500, 100"`.
 - A percentage of the browser window. For example to create three vertical frames, use `cols="10%, 80%, 10%"`.
 - Using a wildcard symbol. For example to create three vertical frames, `usecols="10%, *, 10%"`. In this case wildcard takes remainder of the window.
 - As relative widths of the browser window. For example to create three vertical frames, use `cols="3*, 2*, 1*"`. This is an alternative to percentages. You can use relative widths of the browser window. Here the window is divided into sixths: the first column takes up half of the window, the second takes one third, and the third takes one sixth.
- **rows:** attribute works just like the cols attribute and can take the same values, but it is used to specify the rows in the frameset. For example to create two horizontal frames, use `rows="10%, 90%"`. You can specify the height of each row in the same way as explained above for columns.
- **border:** attribute specifies the width of the border of each frame in pixels. For example `border="5"`. A value of zero specifies that no border should be there.
- **frameborder:** specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example `frameborder="0"` specifies no border.
- **framespacing:** specifies the amount of space between frames in a frameset. This can take any integer value. For example `framespacing="10"` means there should be 10 pixels spacing between each frames.

Loading Content - The <frame> Element:

The <frame> element indicates what goes in each frame of the frameset. The <frame> element is always an empty element, and therefore should not have any content, although each <frame> element should always carry one attribute, `src`, to indicate the page that should represent that frame.

From the above example, lets take small snippet:

```
<frame src="/html/top_frame.htm" />
<frame src="/html/main_frame.htm" />
<frame src="/html/bottom_frame.htm" />
```

The <frame> Element Attributes:

Following are important attributes of and should be known to you to use frames.

- **src:** indicates the file that should be used in the frame. Its value can be any URL. For example, `src="/html/top_frame.htm"` will load an HTML file available in html directory.
- **name:** attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when you want to create links in one frame that load pages into a second frame, in which case the second frame needs a name to identify itself as the target of the link.
- **frameborder:** attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the `frameborder` attribute on the <frameset> element if one is given, and the possible values are the same. This can take values either 1 (yes) or 0 (no).

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

- **marginwidth:** allows you to specify the width of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example `marginwidth="10"`.
- **marginheight:** allows you to specify the height of the space between the top and bottom of the frame's borders and its contents. The value is given in pixels. For example `marginheight="10"`.
- **noresize:** By default you can resize any frame by clicking and dragging on the borders of a frame. The `noresize` attribute prevents a user from being able to resize the frame. For example `noresize="noresize"`.
- **scrolling:** controls the appearance of the scrollbars that appear on the frame. This takes values either "yes", "no" or "auto". For example `scrolling="no"` means it should not have scroll bars.
- **longdesc:** allows you to provide a link to another page containing a long description of the contents of the frame. For example `longdesc="framedescription.htm"`

Browser Support - The `<noframes>` Element:

If a user is using any old browser or any browser which does not support frames then `<noframes>` element should be displayed to the user.

In XHTML you must place a `<body>` element inside the `<noframes>` element because the `<frameset>` element is supposed to replace the `<body>` element, but if a browser does not understand the `<frameset>` element it should understand what is inside the `<body>` element contained in the `<noframes>` element.

You can put some nice message for your user having old browsers. For example *Sorry!! your browser does not support frames.*

Frame's name and target attributes:

One of the most popular uses of frames is to place navigation bars in one frame and then load the pages with the content into a separate frame.

As you have already seen, each `<frame>` element can carry the *name* attribute to give each frame a name. This name is used in the links to indicate which frame the new page should load into. Consider this very simple example, create following content in `index.htm` file:

```
<frameset cols="200, *">
  <frame src="/html/menu.htm" name="menu_page" />
  <frame src="/html/main.htm" name="main_page" />
</frameset>
```

There are two columns in this example. The first is 200 pixels wide and will contain the navigation bar. The second column or frame will contain the main part of the page. The links on the left side navigation bar will load pages into the right side main page.

Keep some content in `main.htm` file and the links in the `menu.htm` file look like this:

```
<a href="http://www.google.com" target="main_page">Google</a>
<br /><br />
<a href="http://www.microsoft.com" target="main_page">Microsoft</a>
<br /><br />
<a href="http://news.bbc.co.uk/" target="main_page">BBC News</a>
```

Inline Frames - The `<iframe>` Element:

You can define an inline frame with the `<iframe>` tag. The `<iframe>` tag is not used within a `<frameset>` tag. Instead, it appears anywhere in your document. The `<iframe>` tag defines a rectangular region within the document in which the browser displays a separate document,

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

including scrollbars and borders. Use the *src* attribute with `<iframe>` to specify the URL of the document that occupies the inline frame. All of the other, optional attributes for the `<iframe>` tag, including *name*, *class*, *frameborder*, *id*, *longdesc*, *marginheight*, *marginwidth*, *name*, *scrolling*, *style*, and *title* behave exactly like the corresponding attributes for the `<frame>` tag. Following is the example to show how to use the `<iframe>`. This tag is used along with `<body>` tag:

```
<body>
...other document content...
<iframe src="/html/menu.htm" width="75" height="200" align="right">
Your browser does not support inline frames. To view this
<a href="/html/menu.htm">document</a> correctly, you'll need
a copy of Internet Explorer or the latest Netscape Navigator.
</iframe>
...subsequent document content...
</body>
```

HTML Lists Formatting

You can list out your items, subjects or menu in the form of a list. HTML gives you three different types of lists.

- `` - An unordered list. This will list items using bullets
- `` - A ordered list. This will use different schemes of numbers to list your items
- `<dl>` - A definition list. This is arrange your items in the same way as they are arranged in a dictionary.

HTML Unordered Lists:

An unordered list is a collection of related items that have no special order or sequence. The most common unordered list you will find on the Web is a collection of hyperlinks to other documents.

This list is created by using `` tag. Each item in the list is marked with a butllet. The bullet itself comes in three flavors: squares, discs, and circles. The default bullet displayed by most web browsers is the traditional full disc.

One Movie list is given below:

```
<center>
<h2>Movie List</h2>
</center>
<ul>
<li>Ram Teri Ganga Meli</li>
<li>Mera Naam Jocker</li>
<li>Titanic</li>
<li>Ghost in the ship</li>
</ul>
```

This will produce following result:

Movie List

- Ram Teri Ganga Meli
- Mera Naam Jocker
- Titanic
- Ghost in the ship

You can use *type* attribute to specify the type of bullet you like. By default its is a disc. Following are the possible way:

```
<ul type="square">  
<ul type="disc">  
<ul type="circle">
```

<ul type="square">

- Hindi
- English
- Maths
- Physics

<ul type="disc">

- Hindi
- English
- Maths
- Physics

<ul type="circle">

- Hindi
- English
- Maths
- Physics

HTML Ordered Lists:

The typical browser formats the contents of an ordered list just like an unordered list, except that the items are numbered instead of bulleted. The numbering starts at one and is incremented by one for each successive ordered list element tagged with ``

This list is created by using `` tag. Each item in the list is marked with a number.

One Movie list is given below:

```
<center>  
<h2>Movie List</h2>  
</center>  
<ol>  
<li>Ram Teri Ganga Meli</li>  
<li>Mera Naam Jocker</li>  
<li>Titanic</li>  
<li>Ghost in the ship</li>  
</ol>
```

This will produce following result:

Movie List

1. Ram Teri Ganga Meli
2. Mera Naam Jocker
3. Titanic
4. Ghost in the ship

You can use *type* attribute to specify the type of numbers you like. By default its is a generic numbers. Following are the other possible way:

```
<ol type="I"> - Upper-Case Numerals.  
<ol type="i"> - Lower-Case Numerals.  
<ol type="a"> - Lower-Case Letters.
```

`<ol type="A">` - Upper-Case Letters.

<code><ol type="I"></code>	<code><ol type="i"></code>	<code><ol type="a"></code>	<code><ol type="A"></code>
I. Hindi	i. Hindi	a. Hindi	A. Hindi
II. English	ii. English	b. English	B. English
III. Maths	iii. Maths	c. Maths	C. Maths
IV. Physics	iv. Physics	d. Physics	D. Physics

You can use *start* attribute to specify the beginning of any index. By default its is a first number or character. In the following example index starts from 5:

```
<center>
<h2>Movie List</h2>
</center>
<ol start="5">
<li>Ram Teri Ganga Meli</li>
<li>Mera Naam Jocker</li>
<li>Titanic</li>
<li>Ghost in the ship</li>
</ol>
```

This will produce following result:

Movie List

5. Ram Teri Ganga Meli
6. Mera Naam Jocker
7. Titanic
8. Ghost in the ship

HTML Definition Lists:

HTML and XHTML also support a list style entirely different from the ordered and unordered lists we have discussed so far - definition lists . Like the entries you find in a dictionary or encyclopedia, complete with text, pictures, and other multimedia elements, the Definition List is the ideal way to present a glossary, list of terms, or other name/value list.

Definition List makes use of following three tags.

- `<dl>` - Defines the start of the list
- `<dt>` - A term
- `<dd>` - Term definition
- `</dl>` - Defines the end of the list

Example:

```
<dl>
<dt><b>HTML</b></dt>
<dd>This stands for Hyper Text Markup Language</dd>
<dt><b>HTTP</b></dt>
<dd>This stands for Hyper Text Transfer Protocol</dd>
</dl>
```

This will produce following result:

HTML

This stands for Hyper Text Markup Language
HTTP
This stands for Hyper Text Transfer Protocol

HTML Page Layouts

Web page layout is very important to give better look to your website. You should design your webpage layout very carefully. You may have noticed that there are many websites which have put their content in multiple columns - they are formatted like a magazine or newspaper. This is easily achieved by using tables or division or span tags. Sometime you use CSS as well to position various elements or to create backgrounds or colorful look for the pages.

HTML Layout - Using Tables:

The simplest and most popular way of creating layouts is using HTML <table> tag. These tables are arranged in columns and rows, so you can utilize these rows and columns in whatever way you like.

For example:-

The following HTML layout example is achieved using a table with 3 rows and 2 columns - but the header and footer column spans both columns using the colspan attribute:

```
<table width="100%" border="0">
<tr>
  <td colspan="2" style="background-color:#CC99FF;">
    <h1>This is Web Page Main title</h1>
  </td>
</tr>
<tr valign="top">
  <td style="background-color:#FFCCFF;
    width:100px;text-align:top;">
    <b>Main Menu</b><br />
    HTML<br />
    PHP<br />
    PERL...
  </td>
  <td style="background-color:#eeeeee;height:200px;
    width:300px;text-align:top;">
    Technical and Managerial Tutorials
  </td>
</tr>
<tr>
  <td colspan="2" style="background-color:#CC99FF;">
    <center>
      Copyright © 2007 Tutorialspoint.com
    </center>
  </td>
</tr>
</table>
```

This will produce following result:

This is Web Page Main title	
Main Menu HTML PHP PERL...	Technical and Managerial Tutorials
Copyright © 2007 Tutorialspoint.com	

Multiple Columns Layouts - Using Tables

You can design your webpage to put your web content in multiple pages. You can keep your content in middle column and you can use left column to use menu and right column can be used to put advertisement or some other stuff. It will be very similar to our site tutorialspoint.com.

Here is an example to create three column layout:

```
<table width="100%" border="0">
<tr valign="top">
  <td style="background-color:#FFCCFF;width:20%;
    text-align:top;">
    <b>Main Menu</b><br />
    HTML<br />
    PHP<br />
    PERL...
  </td>
  <td style="background-color:#eeeeee;height:200px;
    width:60%;text-align:top;">
    Technical and Managerial Tutorials
  </td>
  <td style="background-color:#FFCCFF;
    width:20%;text-align:top;">
    <b>Right Menu</b><br />
    HTML<br />
    PHP<br />
    PERL...
  </td>
</tr>
</table>
```

This will produce following result:

Main Menu HTML PHP PERL...	Technical and Managerial Tutorials	Right Menu HTML PHP PERL...
--	------------------------------------	---

HTML Layouts - Using DIV, SPAN:-

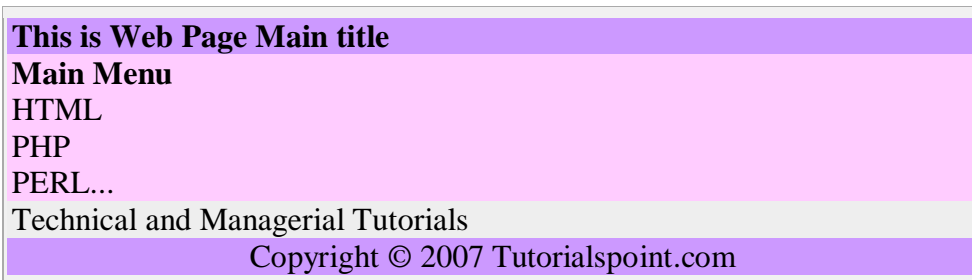
The div element is a block level element used for grouping HTML elements. While the div tag is a block-level element, the HTML span element is used for grouping elements at an inline level.

Although we can achieve pretty nice layouts with HTML tables, tables weren't really designed as a layout tool. Tables are more suited to presenting tabular data.

You can achieve same result whatever you have achieved using <table> tag in previous example.

```
<div style="width:100%">
  <div style="background-color:#CC99FF;">
    <b style="font-size:150%">This is Web Page Main title</b>
  </div>
  <div style="background-color:#FFCCFF;
    height:200px;width:100px;float:left;">
    <b>Main Menu</b><br />
    HTML<br />
    PHP<br />
    PERL...
  </div>
  <div style="background-color:#eeeeee;
    height:200px;width:300px;float:left;">
    Technical and Managerial Tutorials
  </div>
  <div style="background-color:#CC99FF;clear:both">
  <center>
    Copyright © 2007 Tutorialspoint.com
  </center>
  </div>
</div>
```

This will produce following result:



HTML Forms

HTML Forms are required when you want to collect some data from the site visitor. For example registration information: name, email address, credit card, etc.

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

A form will take input from the site visitor and then will post your back-end application such as CGI, ASP Script or PHP script etc. Then your back-end application will do required processing on that data in whatever way you like.

Form elements are like text fields, text area fields, drop-down menus, radio buttons, checkboxes, etc. which are used to take information from the user.

A simple syntax of using <form> is as follows:

```
<form action="back-end script" method="posting method">  
  form elements like input, textarea etc.  
</form>
```

Most frequently used form attributes are:

- **name:** This is the name of the form.
- **action:** Here you will specify any script URL which will receive uploaded data.
- **method:** Here you will specify method to be used to upload data. It can take various values but most frequently used are GET and POST.
- **target:** It specifies the target page where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
- **enctype:** You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server.

There are different types of form controls that you can use to collect data from a visitor to your site.

- Text input controls
- Buttons
- Checkboxes and radio buttons
- Select boxes
- File select boxes
- Hidden controls
- Submit and reset button

HTML Forms - Text Input Controls:

There are actually three types of text input used on forms:

- **Single-line text input controls:** Used for items that require only one line of user input, such as search boxes or names. They are created using the <input> element.
- **Password input controls:** Single-line text input that mask the characters a user enters.
- **Multi-line text input controls:** Used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created with the <textarea> element.

Single-line text input controls:

Single-line text input controls are created using an <input> element whose type attribute has a value of text.

Here is a basic example of a single-line text input used to take first name and last name:

```
<form action="/cgi-bin/hello_get.cgi" method="get">  
First name:  
<input type="text" name="first_name" />  
<br>  
Last name:
```

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

```
<input type="text" name="last_name" />
<input type="submit" value="submit" />
</form>
```

This will produce following result:



First name:

Last name:

Following is the list of attributes for <input> tag.

- **type:** Indicates the type of input control you want to create. This element is also used to create other form controls such as radio buttons and checkboxes.
- **name:** Used to give the name part of the name/value pair that is sent to the server, representing each form control and the value the user entered.
- **value:** Provides an initial value for the text input control that the user will see when the form loads.
- **size:** Allows you to specify the width of the text-input control in terms of characters.
- **maxlength:** Allows you to specify the maximum number of characters a user can enter into the text box.

Password input controls:

This is also a form of single-line text input controls are created using an <input> element whose type attribute has a value of password.

Here is a basic example of a single-line password input used to take user password:

```
<form action="/cgi-bin/hello_get.cgi" method="get">
Login :
<input type="text" name="login" />
<br>
Password:
<input type="text" name="password" />
<input type="submit" value="submit" />
</form>
```

This will produce following result:



Login :

Password :

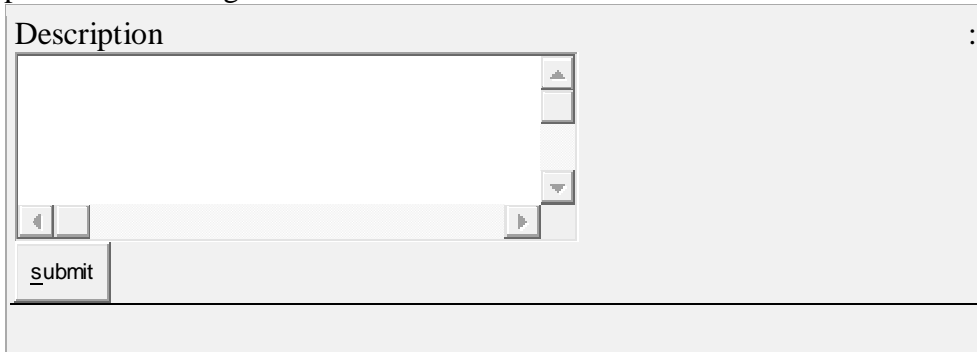
Multiple-Line Text Input Controls:

If you want to allow a visitor to your site to enter more than one line of text, you should create a multiple-line text input control using the <textarea> element.

Here is a basic example of a multi-line text input used to take item description:

```
<form action="/cgi-bin/hello_get.cgi" method="get">
Description : <br />
<textarea rows="5" cols="50" name="description">
Enter description here...
</textarea>
<input type="submit" value="submit" />
</form>
```

This will produce following result:

A screenshot of a web browser displaying a form. The form has a label 'Description' followed by a colon. Below the label is a multi-line text area with a light gray background and a vertical scrollbar on the right. At the bottom left of the text area, there are small navigation icons (back, forward, home, etc.). Below the text area is a button labeled 'submit'.

Following is the detail of above used attributes for <textarea> tag.

- **name:** The name of the control. This is used in the name/value pair that is sent to the server.
- **rows:** Indicates the number of rows of text area box.
- **cols:** Indicates the number of columns of text area box.

HTML Forms - Creating Button:

There are various ways in HTML to create clickable buttons. You can create clickable button using <input> tag.

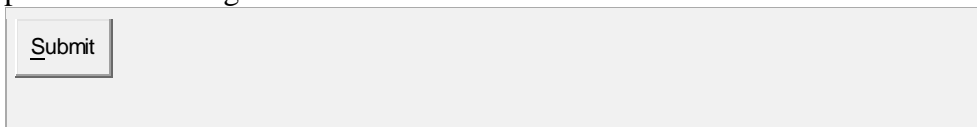
When you use the <input> element to create a button, the type of button you create is specified using the type attribute. The type attribute can take the following values:

- **submit:** This creates a button that automatically submits a form.
- **reset:** This creates a button that automatically resets form controls to their initial values.
- **button:** This creates a button that is used to trigger a client-side script when the user clicks that button.

Here is the example:

```
<form action="http://www.example.com/test.asp" method="get">
<input type="submit" name="Submit" value="Submit" />
<br /><br />
<input type="reset" value="Reset" />
<input type="button" value="Button" />
</form>
```

This will produce following result:

A screenshot of a web browser displaying a form. The form contains a single button labeled 'Submit'.

Reset

You can use an image to create a button. Here is the syntax:

```
<form action="http://www.example.com/test.asp" method="get">
<input type="image" name="imagebutton" src="URL" />
</form>
```

Here *src* attribute specifies a location of the image on your webserver.

You can use `<button>` element to create various buttons. Here is the syntax:

```
<form action="http://www.example.com/test.asp" method="get">
<button type="submit">Submit</button>
<br /><br />
<button type="reset"> Reset </button>
<button type="button"> Button </button>
</form>
```

This will produce following result:

Submit
Reset Button

HTML Forms - Checkboxes Control:

Checkboxes are used when more than one option is required to be selected. They are created using `<input>` tag as shown below.

Here is example HTML code for a form with two checkboxes

```
<form action="/cgi-bin/checkbox.cgi" method="get">
<input type="checkbox" name="maths" value="on"> Maths
<input type="checkbox" name="physics" value="on"> Physics
<input type="submit" value="Select Subject" />
</form>
```

The result of this code is the following form

☐ Maths ☐ Physics Select Subject

Following is the list of important checkbox attributes:

- **type:** Indicates that you want to create a checkbox.
- **name:** Name of the control.
- **value:** The value that will be used if the checkbox is selected. More than one checkbox should share the same name only if you want to allow users to select several items from the same list.
- **checked:** Indicates that when the page loads, the checkbox should be selected.

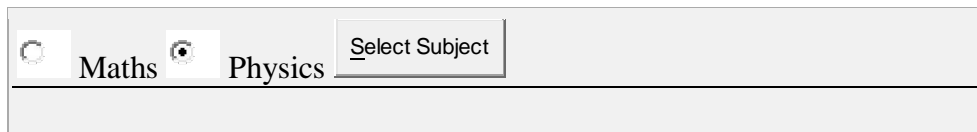
HTML Forms – Radio box Control:

Radio Buttons are used when only one option is required to be selected. They are created using `<input>` tag as shown below:

Here is example HTML code for a form with two radio button:

```
<form action="/cgi-bin/radiobutton.cgi" method="post">
<input type="radio" name="subject" value="maths" /> Maths
<input type="radio" name="subject" value="physics" /> Physics
<input type="submit" value="Select Subject" />
</form>
```

The result of this code is the following form

A screenshot of a web form. It contains two radio buttons. The first radio button is selected and is followed by the text 'Maths'. The second radio button is not selected and is followed by the text 'Physics'. To the right of these is a submit button labeled 'Select Subject'.

Following is the list of important radiobox attributes:

- **type:** Indicates that you want to create a radiobox.
- **name:** Name of the control.
- **value:** Used to indicate the value that will be sent to the server if this option is selected.
- **checked:** Indicates that this option should be selected by default when the page loads.

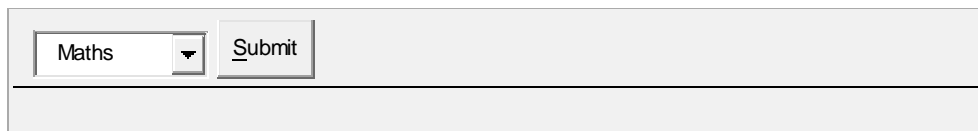
HTML Forms - Select box Control:

Drop down Box is used when we have many options available to be selected but only one or two will be selected.

Here is example HTML code for a form with one drop down box

```
<form action="/cgi-bin/dropdown.cgi" method="post">
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
<input type="submit" value="Submit" />
</form>
```

The result of this code is the following form

A screenshot of a web form. It contains a dropdown menu with 'Maths' selected. To the right of the dropdown is a submit button labeled 'Submit'.

Following is the list of important attributes of <select>:

- **name:** This is the name for the control.
- **size:** This can be used to present a scrolling list box.
- **multiple:** If set to "multiple" then allows a user to select multiple items from the menu.

Following is the list of important attributes of <option>:

- **value:** The value that is sent to the server if this option is selected.

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

- **selected:** Specifies that this option should be the initially selected value when the page loads.
- **label:** An alternative way of labeling options.

HTML Forms - File Select Boxes:

If you want to allow a user to upload a file to your web site from his computer, you will need to use a file upload box, also known as a file select box. This is also created using the `<input>` element.

Here is example HTML code for a form with one file select box

```
<form action="/cgi-bin/hello_get.cgi" method="post"
      name="fileupload" enctype="multipart/form-data">
<input type="file" name="fileupload" accept="image/*" />
</form>
```

The result of this code is the following form

HTML Forms - Hidden Controls:

If you will want to pass information between pages without the user seeing it. Hidden form controls remain part of any form, but the user cannot see them in the Web browser. They should not be used for any sensitive information you do not want the user to see because the user could see this data if she looked in the source of the page.

Following hidden form is being used to keep current page number. When a user will click next page then the value of hidden form will be sent to the back-end application and it will decide which page has be displayed next.

```
<form action="/cgi-bin/hello_get.cgi"
      method="get" name="pages">
<p>This is page 10</p>
<input type="hidden" name="page number" value="10" />
<input type="submit" value="Next Page" />
</form>
```

This will produce following result:

This is page 10

Next Page

HTML Forms - Submit and Reset Button:

These are special buttons which can be created using `<input>` When submit button is clicked then Forms data is submitted to the back-end application. When reset button is clicked then all the forms control are reset to default state.

You already have seen submit button above, we will give one reset example here:

```
<form action="/cgi-bin/hello_get.cgi" method="get">
First name:
<input type="text" name="first_name" />
```

```
<br>
Last name:
<input type="text" name="last_name" />
<input type="submit" value="Submit" />
<input type="reset" value="Reset" />
</form>
```

This will produce following result. Type something and click reset button.



HTML Embed Multimedia - Movie, Music

You can add music or video into your web page. The easiest way to add video or sound to your web site is to include the special HTML tag called `<embed>`. This tag causes the browser itself to include controls for the multimedia automatically. You do not need to have any ActiveX, Java VM, VBscript or JavaScript to support this `<embed>` tag.

it's also a good idea to include the `<noembed>` tag to support browsers which don't recognize the `<embed>` tag. You could, for example, use `<embed>` to display a movie of your choice, and `<noembed>` to display a single JPG image.

Here is a simple example to play embed a midi file:

```
<embed src="/html/yourfile.mid" width="100%" height="60" >
<noembed></noembed>
</embed>
```

You can put any media file in src attribute. You can try it yourself by giving various files.

Attributes:

Following is the list of important attributes for `<embed>` element.

- **align** - Determines how to align the object. It takes either *center*, *left* or *right*.
- **autostart** - Indicates if the media should start automatically. Netscape default is true, Internet Explorer is false.
- **loop** - Specifies if the sound should be played continuously (set loop to true), a certain number of times (a positive value) or not at all (false). This is supported by Netscape only.
- **playcount** - Specifies the number of times to play the sound. This is alternat option for *loop* if you are usiong IE.
- **hidden** - Defines if the object shows on the page. A false value means no and true means yes.
- **height** - Height of the object in pixels or en.
- **width** - Width of the object in pixels or en.
- **pluginspage** - Specifies the URL to get the plugin software.

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

- **name** - A name used to reference the object.
- **src** - URL of the object to be embedded. This can be any recognizable by the user's browser. It could be .mid, .wav, .mp3, .avi and so on).
- **volume** - Controls volume of the sound. Can be from 0 (off) to 100 (full volume). This attribute is supported by Netscape only.

HTML - Video Media Types

Flash movies (.swf), AVI's (.avi), and MOV's (.mov) file types are supported by the embed tag.

- .swf files - are the file types created by Macromedia's Flash program.
- .wmv files - are Microsoft's Window's Media Video file types.
- .mov files - are Apple's Quick Time Movie format.
- .mpeg files - are movie files created by the Moving Pictures Expert Group.

Here is a simple example to play a flash file.

```
<embed src="/html/yourfile.swf" width="100%" height="250" >  
<noembed></noembed>  
</embed>
```

This will produce following result. Select a picture and paint it using virtual bursh.

Background Audio - The <bgsound> Element:

You can use the <bgsound> tag to play a soundtrack in the background. This tag is for Internet Explorer documents only. Other browsers ignore the tag. It downloads and plays an audio file when the host document is first downloaded by the user and displayed. The background sound file also will replay whenever the user refreshes the browser display.

This tag is having only two attributes *loop* and *src*. Both these attributes have same meaning as explained above.

Here is a simple example to play a small midi file:

```
<bgsound src="/html/yourfile.mid" >  
<noembed></noembed>  
</bgsound>
```

This will produce blank screen. This tag does not display any component and remains hidden. Currently, Internet Explorer can handle three different sound format files: wav, the native format for PCs; au, the native format for most Unix workstations; and MIDI, a universal music-encoding scheme.

DHTML

DHTML is basically a combination of **CSS style sheet, Document Object Model (DOM) and java Script** with the result that you can make your web pages more interactive. It can be as simple as an “onmouseover” effect like color changes on our menu or you can use DHTML to do more complex things such as the slide in menu on the left or have a scrolled to draw attention to certain part of your web pages.

Information Sources for Dynamic HTML: - In java Script,
The statement, document.write(); is used to write output to a webpage.

Ex:-

```
<html>
<body>
<script type="text/JavaScript">
Document.write(date());
</script>
</body>
</html>
```

Advantages of DHTML

1. DHTML makes documents dynamic. Dynamic documents:
 - Allow the designer to control how the HTML displays Web pages' content.
 - React and change with the actions of the visitor.
 - Can exactly position any element in the window, and change that position after the document has loaded.
 - Can hide and show content as needed.
2. DHTML allows any HTML element (any object on the screen that can be controlled independently using JavaScript) in Internet Explorer to be manipulated at any time, turning plain HTML into dynamic HTML.
3. With DHTML, changes occur entirely on the client-side
4. Using DHTML gives the author more control over how the page is formatted and how content is positioned on the page.

Components of DHTML

Dynamic HTML includes the following components:

- Conventional HTML
- Scripts – Small programs designed to manipulate Web pages.
- Document Object Model (DOM) – The road map through which you can locate any element in an HTML document and use a scripting language, such as JavaScript, to change the element's properties.

- Absolute Positioning – The elements on the page are placed in a fixed location, as opposed to relative positioning, in which an element's location is relative to particular elements on the page.
- Multimedia filters – Multimedia features that create visual effects for text, images, and other objects, without imposing long download times on the user.

The Document Object Model (DOM)

The DOM is like a roadmap of your Web page. You describe a path that leads from the HTML document down to the various elements on the page. The DOM for an image called button1 would be: document.images.button1.

The aim of DOM is to allow programmers to dynamically update the content, structure and style of Documents.

Document Object Model was created with these four goals in mind:

- To delineate a hierarchical structure representing all parts of a Web document.
- To allow the modification of that structure through adding and removing content.
- To provide a way to monitor and manipulate the characteristics of content on the page.
- To provide information about how the items on a page interrelate with the user and each other.

Components of the DOM

1. **Objects** – The object is the basic unit of the DOM. Every element on the page is part of the DOM. Text and images are examples of objects. A table would be a parent object; its cells would be children of the table. A form named “MyForm” could be referred to as: <FORM NAME=”MyForm”>. After the author names an object, it can be referred to easily in a JavaScript.
2. **Properties** – Properties are adjectives that describe parts of the Web page. Examples could include height, width, color, and size. If a cell border is two pixels wide, its width property would be referenced as: WIDTH=5.
3. **Events** – An event is an action or occurrence of a Web page. When an event occurs on a page, the item that received the event notifies the DOM that the event has occurred. This is called firing the event.

Some examples of events:

- **onMouseDown** – Fired when the user presses the mouse button.
- **onMouseOver** – Fired when the user positions the mouse pointer over an object.
- **onMouseOut** - Fired when the user moves the mouse pointer outside the boundaries of an object.
- **onKeyPress** – Fired when the user presses a key.
- **onFocus** – Fired when the object receives the focus.
- **onClick** – Fired when the mouse button is clicked over an area.
- **Focus-** The element gets the focus.
- **Resize-** The user resizes the window.
- **Submit-** The user submits the form by clicking the submit button or hitting Enter in a text field.
- **Key press-** A key is pressed while the element has fo-cus.

4. Methods – They describe the actions an object can take. For example, the open method directs the window to open a new browser window. Other examples are focus, Run, and reload.
5. Collections – Collections are lists of items that are associated with a particular object. For example, the collection name images is a list of all image objects.
6. Event bubbling – (supported only in Internet Explorer) – When an object fires an event, it also notifies its parent object that the event has occurred. The event continues to travel, or “bubble,” up the hierarchy until it reaches the top or the bubbling action is cancelled.

Examples 1:- the following example show and hidden text.

```
<html>
<body>
<p id="p">My Name is Sunil Kumar</p>
<input type="button" value="Hide text"
onClick="document.getElementById('p').style.visibility='hidden'"/>
<input type="button" value="Show text"
onClick="document.getElementById('p').style.visibility='visible'"/>
</body>
</html>
```

Examples 2:- The following example changes the src attribute of an img element

```
<html>
<body>

<script type="text/javascript">document.getElementById("abc"). src="file:///E:/suneel
bagerpur/sunil ji card/2.jpg";</script>
</body>
</html>
```

Examples 3:- The following example changes the content of an h1 element:

```
<html>
<body>
<h1 id="header">Old Header</h1>
<script type="text/javascript">document.getElementById("header").innerHTML="New
Header";</script>
</body>
</html>
```

Example 4:- Same information print on a page

```
<!DOCTYPE html>
<html>
<head>
<script>
function sameInfo()
{
for (i=0; i<document.myForm1.option.length; i++)
{
document.myForm2.option[i].value=document.myForm1.option[i].value;
}
}
}
```

```
</script>
</head>
<body>
<form name="myForm1">
First name: <input type="text" name="option"><br />
Last name: <input type="text" name="option"><br />
Address: <input type="text" name="option"><br />
E-mail: <input type="text" name="option"><br />
<br />
<input type="button" onclick="sameInfo()" value="Same information below"><br />
</form>
<form name="myForm2">
First name: <input type="text" name="option"><br />
Last name: <input type="text" name="option"><br />
Address: <input type="text" name="option"><br />
E-mail: <input type="text" name="option"><br />
</form>
</body>
</html>
```

Example:-Onblur

```
<!DOCTYPE html>
<html>
<head>
<script>
function message()
{
alert("This alert box was triggered by the onblur event handler");
}
</script>
</head>
<body>
<p>The onblur event occurs when an element loses focus. Try to click or write in the input
field below, then click elsewhere in the document so the input field loses focus.</p>
<form>
Enter your name: <input type="text" onblur="message()" size="20">
</form>
</body>
</html>
```

Example: - align an image

```
<! DOCTYPE html>
<html>
<head>
<script>
function alignImg()
{
document.getElementById("compman").align="right";
}

```

```
</script>
</head>
<body>

<p>Some text. Some text. Some text. Some text.</p>
<input type="button" onclick="alignImg()" value="Align Image" />
</body>
</html>
```

Example:- Onkeypress

```
<!DOCTYPE html>
<html>
<head>
<script>
function message()
{
alert("This alert box was triggered when you pressed a button on your keyboard");
}
</script>
</head>
<body onkeypress="message()">
<p>The onkeypress event is triggered when the user presses a button on the keyboard.</p>
<p>To trigger the onkeypress event, make sure that this window has focus.</p>
</body>
</html>
```

Example: - change position

```
<!DOCTYPE html>
<html>
<head>
<script>
function moveleft()
{
document.getElementById('header').style.position="absolute";
document.getElementById('header').style.left="0";
}
function moveback()
{
document.getElementById('header').style.position="relative";
}
</script>
</head>
<body>
<h1 id="header"
onmouseover="moveleft()"
onmouseout="moveback()">
Mouse over this text</h1>
</body>
```

Example: - Drop down menu

```
<!DOCTYPE html>
<html>
<head>
<style>
body{ font-family:arial;}
table{ font-size:80%;background:black}
a{ color:black;text-decoration:none;font:bold}
a:hover{ color:#606060}
td.menu{ background:lightblue}
table.menu
{
font-size:100%;
position:absolute;
visibility:hidden;
}
</style>
<script>
function showmenu(elmnt)
{
document.getElementById(elmnt).style.visibility="visible";
}
function hidemenu(elmnt)
{
document.getElementById(elmnt).style.visibility="hidden";
}
</script>
</head>
<body>
<h3>Drop down menu</h3>
<table width="100%">
<tr bgcolor="#FF8080">
<td onmouseover="showmenu('tutorials')" onmouseout="hidemenu('tutorials')">
<a href="/default.asp">Tutorials</a><br />
<table class="menu" id="tutorials" width="120">
<tr><td class="menu"><a href="/html/default.asp">HTML</a></td></tr>
<tr><td class="menu"><a href="/css/default.asp">CSS</a></td></tr>
<tr><td class="menu"><a href="/xml/default.asp">XML</a></td></tr>
<tr><td class="menu"><a href="/xsl/default.asp">XSL</a></td></tr>
</table>
</td>
<td onmouseover="showmenu('scripting')" onmouseout="hidemenu('scripting')">
<a href="/default.asp">Scripting</a><br />
<table class="menu" id="scripting" width="120">
<tr><td class="menu"><a href="/js/default.asp">JavaScript</a></td></tr>
<tr><td class="menu"><a href="/vbscript/default.asp">VBScript</a></td></tr>
<tr><td class="menu"><a href="default.asp">DHTML</a></td></tr>
<tr><td class="menu"><a href="/asp/default.asp">ASP</a></td></tr>
<tr><td class="menu"><a href="/ado/default.asp">ADO</a></td></tr>
```

```
</table>
</td>
<td onmouseover="showmenu('validation')" onmouseout="hidemenu('validation')">
  <a href="/site/site_validate.asp">Validation</a><br />
  <table class="menu" id="validation" width="120">
    <tr><td class="menu"><a href="/web/web_validate.asp">Validate HTML</a></td></tr>
    <tr><td class="menu"><a href="/web/web_validate.asp">Validate XHTML</a></td></tr>
    <tr><td class="menu"><a href="/web/web_validate.asp">Validate CSS</a></td></tr>
    <tr><td class="menu"><a href="/web/web_validate.asp">Validate XML</a></td></tr>
    <tr><td class="menu"><a href="/web/web_validate.asp">Validate WML</a></td></tr>
  </table>
</td>
</tr>
</table>
<p>Mouse over these options to see the drop down menus</p>
</body>
</html>
```

Cascading Style Sheets

Cascading Style Sheets is a rich specification that gives the Web page author powerful visual control over such HTML elements as location, size, color, background, images, etc. Cascading Style Sheets is supported in HTML 4.0. Cascading Style Sheets Allow you to define how HTML tags should display their content. Let you position HTML elements anywhere in the window, as well as control the visibility of those elements.

With CSS, you can specify a number of style properties for a given HTML element. Each property has a name and a value, separated by a colon (:). Each property declaration is separated by a semi-colon (;).

```
<p style="color:red;font-size:24px;">Using Style Sheet Rules</p>
```

This will produce following result:

```
Using Style Sheet Rules
```

There are three ways of using a style sheet in an HTML document:

- 1) **Internal Style Sheet:-** An internal style sheet can be used if one single document has a unique style. Internal styles are defined in the <head> section of an HTML page, by using the <style> tag, like this:

```
<head>
<style type="text/css">
body {background-color:yellow;}
p {color:blue;}
</style>
</head>
```

- 2) **External Style Sheet:-** An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the <head> section:

Example

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

3) Inline Style Sheet:

You can apply style sheet rules directly to any HTML element. This should be done only when you are interested to make a particular change in any HTML element only. To use inline styles you use the style attribute in the relevant tag. Below is an example:

```
<p style="color:red;font-size:24px;">Using Style Sheet Rules</p>
```

CSS rules: - define what the HTML should look like and how it should behave in the browser window.

- **Selectors** – Alpha-numeric characteristics that identify the rule
- **Properties** – What is being defined
- **Values** – Assigned to a property by defining its nature. A value can be a keyword, such as “yes” or “no,” a number, or a percentage.

Style – A style is a group of properties that define how an HTML element will appear in a document. CSS lets you make changes to an entire document. CSS operates by collecting all the different properties such as bold, italic, font size, etc., that you want to apply to similar types of text, then giving these groups of properties a common name.

What can we do with CSS?

Cascading Style Sheets allows us to manipulate the text of our Web pages in many ways. Some of them are:

1. Control font sizes, styles, colors, etc.
2. Control text, by adjusting it in various ways, such as:
 - Kerning – The amount of space between the letters in a word.
 - Word spacing.
 - Leading – Line height
 - Justifying the text – Setting the horizontal alignment (left, right, center) of the text
 - Aligning the text vertically.

**DEPARTMENT OF COMPUTER APPLICATIONS,
IFTM UNIVERSITY MORADABAD**

- Indenting paragraphs
 - Controlling text case
 - Decorating the text – You can embellish the text with different looks: underline, overline, line-through, or supported by Internet Explorer.
 - Controlling the amount of white space on a line
3. Creating lists
 4. Controlling background colors and graphics
 5. Controlling margins and borders
 6. Controlling the position of elements on a page
 7. Controlling visibility or invisibility
 8. Absolute positioning
 9. Relative positioning
 10. Static positioning
 11. Setting margins etc.

Example:-

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
body
{
background-color:#d0e4fe;
}
h1
{
color:orange;
text-align:center;
}
p
{
font-family:"Times New Roman";
font-size:20px;
}
</style>
</head>
<body>
<h1>CSS example!</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Example:- Id selector

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
#para1
```

```
{
text-align:center;
color:red;
}
</style>
</head>
<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
</body>
</html>
```

Example:- class sector

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
.center
{
text-align:center;
}
</style>
</head>
<body>
```

Example :- Font, Color and Size

```
<!DOCTYPE html>
<html>
<body>
<h1 style="font-family:verdana;">A heading</h1>
<p style="font-family:arial;color:red;font-size:20px;">A paragraph.</p>
</body>
</html>
```

Example :-Text Alignment

```
<!DOCTYPE html>
<html>
<body>
<h1 style="text-align:center;">Center-aligned heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

HTML Scripts

A *script* is a small piece of program that can add interactivity to your website. For example, a script could generate a pop-up alert box message, or provide a dropdown menu. This script could be Javascript or VBScript.

You can write your Event Handlers using any of the scripting language and then you can trigger those functions using HTML attributes.

There are two ways of using a style sheet in an HTML document:

1.External Script:

If you have to use single script functionality among many HTML pages then it is a good idea to keep that function in a single script file and then include this file in all the HTML pages. You can include a style sheet file into HTML document using `<script>` element. Below is an example:

```
<head>
<script src="yourfile.js" type="text/javascript" />
</head>
```

2.Internal Script:

You can write your script code directly into your HTML document. Usually we keep script code in header of the document using `<script>` tag, otherwise there is no restriction and you can put your source code anywhere in the document. You can specify whether to make a script run automatically (as soon as the page loads), or after the user has done something (like click on a link). Below is an example this would write a *Hello Javascript!* message as soon as the page loads.:

```
<head>
<title>Internal Script</title>
</head>
<body>
<script type="text/javascript">
    document.write("Hello Javascript!")
</script>
</body>
```

This will produce following result:

```
Advertisements

Hello Javascript!
```

It is very easy to write an event handler. Following example explains how to write an event handler. Let's write one simple function *myAlert* in the header of the document. We will call this function when any user will bring mouse over a paragraph written in the example.

```
<head>
<title>Event Handler Example t</title>
<script type="text/javascript">
function myAlert()
{
    alert("I am an event handler....");
    return;
}
```

```
</script>
</head>
<body>
<span onmouseover="myAlert();">
  Bring your mouse here to see an alert
</span>
</body>
```

Now this will produce following result. Bring your mouse over this line and see the result:

Advertisements
Bring your mouse here to see an alert

The <noscript> Element:

You can also provide alternative info for users whose browsers don't support scripts and for users who have disabled scripts. You do this using the `<noscript>` tag.

JavaScript Example:

```
<script type="text/javascript">
<!--
document.write("Hello Javascript!");
//-->
</script>
<noscript>Your browser does not support Javascript!</noscript>
```

VBScript Example:

```
<script type="text/vbscript">
<!--
document.write("Hello VBScript!")
'-->
</script>
<noscript>Your browser does not support VBScript!</noscript>
```

Default Scripting Language

You can specify a default scripting language for all your *script* tags to use. This saves you from having to specify the language everytime you use a script tag within the page. Below is the

Example: `<meta http-equiv="Content-Script-Type" content="text/JavaScript">`

Example:-

```
<!DOCTYPE html>
<html>
<body>
<h1>My Web Page</h1>
<p id="myPar">I am a paragraph.</p>
<div id="myDiv">I am a div.</div>
<p>
<button type="button" onclick="myFunction()">Try it</button>
</p>
<script>
function myFunction()
{
document.getElementById("myPar").innerHTML="Hello Dolly";
document.getElementById("myDiv").innerHTML="How are you?";
}
</script>
<p>When you click on "Try it", the two elements will change.</p>
</body>
</html>
```

Example:-

```
<!DOCTYPE html>
<html>
<body>
<h1 id="myH1"></h1>
<p id="myP"></p>
<script>
// Write to a heading:
document.getElementById("myH1").innerHTML="Welcome to my Homepage";
// Write to a paragraph:
document.getElementById("myP").innerHTML="This is my first paragraph.";
</script>
<p><strong>Note:</strong> The comments are not executed.</p>
</body>
</html>
```

Example:-

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
alert("Hello! I am an alert box!");
}
</script>
</head>
```

```
<body>
<input type="button" onclick="myFunction()" value="Show alert box" />
</body>
</html>
```

Example:-

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to demonstrate line-breaks in a popup box.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
alert("Hello\nHow are you?");
}
</script>
</body>
</html>
```

Example:- Call a function

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
alert("Hello World!");
}
</script>
</head>

<body>
```

```
<button onclick="myFunction()">Try it</button>
```

```
<p>By clicking the button above, a function will be called. The function will alert a
message.</p>
```

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