

## HTML COMMON TAGS:

### LIST:

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

1. **<ul>** - An unordered list. This will list items using bullets
2. **<ol>** - A ordered list. This will use different schemes of numbers to list your items
3. **<dl>** - A definition list. This is arranging 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 `<ul>` tag. Each item in the list is marked with a bullet. The bullet itself comes in three flavors: squares, discs, and circles. The default bullet displayed by most web browsers is the traditional full disc.

#### Example:

```
<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

## UNIT-I

---

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">
```

<code>&lt;ul type="square"&gt;</code>	<code>&lt;ul type="disc"&gt;</code>	<code>&lt;ul type="circle"&gt;</code>
<ul style="list-style-type: none"><li>▪ Hindi</li><li>▪ English</li><li>▪ Maths</li><li>▪ Physics</li></ul>	<ul style="list-style-type: none"><li>• Hindi</li><li>• English</li><li>• Maths</li><li>• Physics</li></ul>	<ul style="list-style-type: none"><li>○ Hindi</li><li>○ English</li><li>○ Maths</li><li>○ Physics</li></ul>

### 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 `<li>`

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

#### Example:

```
<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>
```

# UNIT-I

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>&lt;ol type="I"&gt;</code>	<code>&lt;ol type="i"&gt;</code>	<code>&lt;ol type="a"&gt;</code>	<code>&lt;ol type="A"&gt;</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>
```

## UNIT-I

---

```
<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>
```

# UNIT-I

---

This will produce following result:

## HTML

This stands for Hyper Text Markup Language

## HTTP

This stands for Hyper Text Transfer Protocol

## TABLE:

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>
```

This will produce following result:

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

In the above example border is an attribute of <table> and it will put border across all the cells. If you do not need a border then you can use border="0".

# UNIT-I

---

## 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.

### Example:

```
<table border="1" cellpadding="5" cellspacing="5">
<tr>
```

## UNIT-I

---

```
<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:

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.

**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>
```

## UNIT-I

---

`</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 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.

**Example using bgcolor:**

```
<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		



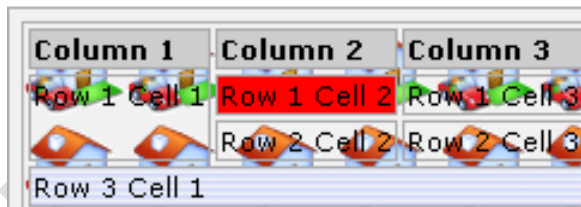
## UNIT-I

---

**Example using background:**

```
<table border="1" background="/images/home.gif">
<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" background="/images/pattern1.gif">
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.

**Example:**

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

## UNIT-I

---

```
<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

### Using Table Caption:

The *caption* tags will serve as a title or explanation and show up at the top of the table. This tag is deprecated in newer version of HTML/XHTML.

```
<table border="1">
<caption>This is the caption</caption>
<tr>
<td>row 1, column 1</td><td>row 1, columnn 2</td>
</tr>
</table>
```

This will produce following result:

This is the caption	
row 1, column 1	row 1, column 2

### Using a Header, Body, and Footer:

Tables can be divided into three portions: a header, a body, and a foot. The head and foot are rather similar to headers and footers in a word-processed document that remain the same for every page, while the body is the main content of the table.

The three elements for separating the head, body, and foot of a table are:

- **<thead>** - to create a separate table header.
- **<tbody>** - to indicate the main body of the table.
- **<tfoot>** - to create a separate table footer.

## UNIT-I

---

A table may contain several <tbody> elements to indicate different *pages* or groups of data. But it is notable that <thead> and <tfoot> tags should appear before <tbody>

```
<table border="1" width="100%">
  <thead>
  <tr>
  <td colspan="4">This is the head of the table</td>
  </tr>
  </thead>
  <tfoot>
  <tr>
  <td colspan="4">This is the foot of the table</td>
  </tr>
  </tfoot>
  <tbody>
  <tr>
  <td>Cell 1</td>
  <td>Cell 2</td>
  <td>Cell 3</td>
  <td>Cell 4</td>
  </tr>
  <tr>
  ...more rows here containing four cells...
  </tr>
  </tbody>
  <tbody>
  <tr>
  <td>Cell 1</td>
  <td>Cell 2</td>
  <td>Cell 3</td>
  <td>Cell 4</td>
  </tr>
  <tr>
```

## UNIT-I

---

*...more rows here containing four cells...*

`</tr>`

`</tbody>`

`</table>`

This will produce following result:

This is the head of the table			
Cell 1	Cell 2	Cell 3	Cell 4
...more rows here containing four cells...			
Cell 1	Cell 2	Cell 3	Cell 4
...more rows here containing four cells...			
This is the foot of the table			

### 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>
```

## UNIT-I

---

```
<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:

<b>Name</b>	<b>Salary</b>	<ul style="list-style-type: none"><li>• This is another cell</li><li>• Using list inside this cell</li></ul>
Ramesh Raman	5000	
Shabbir Hussein	7000	
Row 2, Column 1	Row 2, Column 2	

### IMAGE:

Images are very important to beautify as well as to depict many concepts on your web page.

#### Insert Image - The <img> Element:

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

```

```

#### Image Attributes:

Following are most frequently used attributes for <img> 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.

# UNIT-I

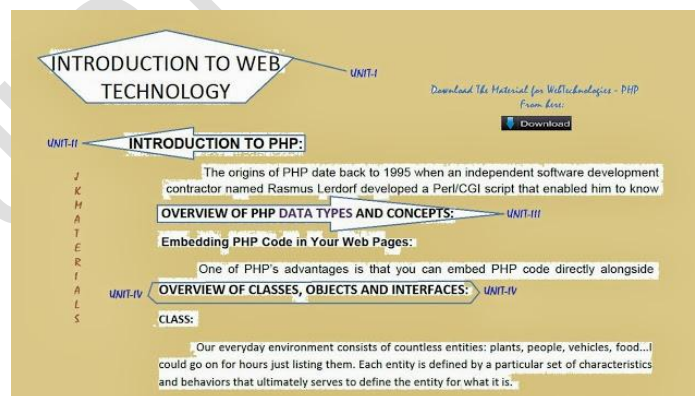
- **alt:** this is an alternate text which will be displayed if image is missing.
- **align:** this sets horizontal alignment of the image and takes value either *left*, *right* or *center*.
- **valign:** this sets vertical alignment of the image and takes value either *top*, *bottom* or *center*.
- **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 <img> 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.

## A Simple Example:

```

```

This will produce following result:



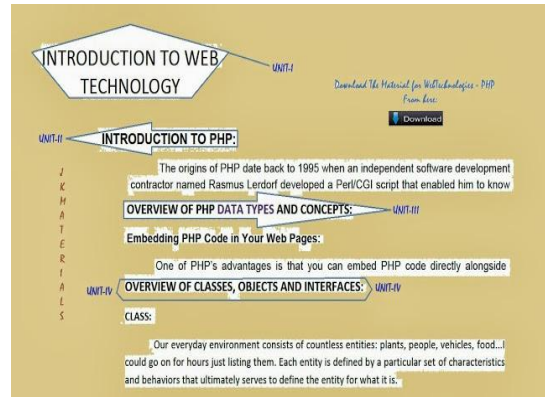
## Image Attributes - width, height, title, border and align:

Now let us try to set some more attributes:

```

```

This will produce following result:



**Wrapping text around images:**

**Example 1:**

```
<p>This is the first paragraph that appears above the paragraph with the image!</p>
```

```
<p>
```

The image will appear along the right hand side of the paragraph. As you can see this is very nice for adding a little eye candy that relates to the specified paragraph.</p>

<p>The left and right image-alignment values tell the browser to place an image against the left or right margin, respectively, of the current text flow. The browser then renders subsequent document content in the remaining portion of the flow adjacent to the image. The net result is that the document content following the image gets wrapped around the image. </p>

This will produce following result:

This is the first paragraph that appears above the paragraph with the image!

The image will appear along the right hand side of the paragraph. As you can see this is very nice for adding a little eye candy that relates to the specified paragraph.



## UNIT-I

---

The left and right image-alignment values tell the browser to place an image against the left or right margin, respectively, of the current text flow. The browser then renders subsequent document content in the remaining portion of the flow adjacent to the image. The net result is that the document content following the image gets wrapped around the image.

### FORM:

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.

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, textarea 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. Possible values are like:
  - **application/x-www-form-urlencoded** - This is the standard method most forms use. It converts spaces to the plus sign and non-alphanumeric characters into the hexadecimal code for that character in ASCII text.
  - **multipart/form-data** - This allows the data to be sent in parts, with each consecutive part corresponding to a form control, in the order they appear in the form. Each part can have an optional content-type header of its own indicating the type of data for that form control.



## UNIT-I

---

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

# UNIT-I

---

This will produce following result:



A screenshot of a web form. It contains two text input fields. The first is labeled "First name:" and the second is labeled "Last name:". Below these fields is a button labeled "submit".

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:



A screenshot of a web form. It contains two text input fields. The first is labeled "Login :" and the second is labeled "Password :". Below these fields is a button labeled "submit".

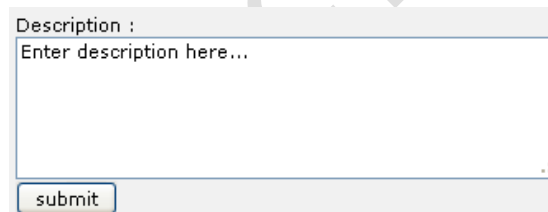
## Multiple-Line Text Input Controls:

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:



The screenshot shows a web browser window with a form. The form has a label "Description :" followed by a multi-line text input field containing the placeholder text "Enter description here...". Below the text field is a "submit" button. The browser window has a standard title bar and a scroll bar on the right side.

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.

## UNIT-I

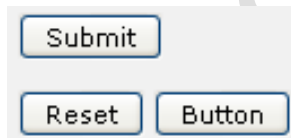
---

- **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.

**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:



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 web server.

### 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.

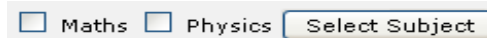
Here is example HTML code for a form with two checkboxes

```
<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>
```

## UNIT-I

---

The result of this code is the following form:



A screenshot of a web form. It contains two checkboxes: one for 'Maths' and one for 'Physics'. To the right of these checkboxes is a button labeled '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.

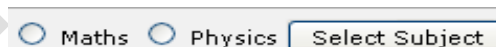
### HTML Forms - Radio box Control:

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: one for 'Maths' and one for 'Physics'. To the right of these radio buttons is a 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.

# UNIT-I

---

## 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:



The image shows a rendered HTML form. It consists of a dropdown menu with the text 'Maths' and a small downward arrow, and a 'Submit' button to its right. The entire form is enclosed in a light gray border.

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

## 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.

## UNIT-I

---

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.

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.

### 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>
```

## UNIT-I

---

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

### The <frameset> Element Attributes:

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

- **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, use `cols="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, use `cols="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.



## UNIT-I

---

- **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, let's 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).
- **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"`.

## UNIT-I

---

- **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"`

### 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>
```

The *target* attribute can also take the attribute values listed in the table that follows.

## UNIT-I

Value	Description
_self	Loads the page into the current frame.
_blank	Loads a page into a new browser window. Opening a new window.
_parent	Loads the page into the parent window, which in the case of a single frameset is the main browser window.
_top	Loads the page into the browser window, replacing any current frames..

### CASCADING STYLE SHEETS:

Style sheets describe how documents are presented on screens, in print, or perhaps how they are pronounced. W3C has actively promoted the use of style sheets on the Web since the Consortium was founded in 1994.

Cascading Style Sheets (CSS) is a style sheet mechanism that has been specifically developed to meet the needs of Web designers and users.

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:

### External Style Sheet:

If you have to give same look and feel to too many pages then it is a good idea to keep all the style sheet rules in a single style sheet file and include this file in all the HTML pages. You can include a style sheet file into HTML document using <link> element. Below is an example:

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

# UNIT-I

---

## Internal Style Sheet:

If you want to apply Style Sheet rules to a single document only then you can include those rules into that document only. Below is an example:

```
<head>
<style type="text/css">
  body{background-color: pink;}
  p{color:blue; 20px;font-size:24px;}
</style>
</head>
```

## 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>
```

## INTRODUCTION TO JAVASCRIPT:

JavaScript is a scripting language produced by Netscape for use within HTML Web pages. JavaScript is loosely based on Java and it is built into all the major modern browsers. JavaScript started life as LiveScript, but Netscape changed the name, possibly because of the excitement being generated by Java to JavaScript. JavaScript made its first appearance in Netscape 2.0 in 1995 with a name LiveScript.

JavaScript is a lightweight, interpreted programming language with object - oriented capabilities that allows you to build interactivity into otherwise static HTML pages. The general - purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers

Client - side JavaScript: Client - side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need no longer be static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

# UNIT-I

---

The JavaScript client - side mechanism features many advantages over traditional CGI server - side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid they would be submitted to the Web Server. JavaScript can be used to trap user - initiated events such as button clicks, link navigation, and other actions that the user explicitly or implicitly initiates.

JavaScript is a programming language that allows users to add interactive content to web pages. The browser reads the code and executes its instructions. JavaScript commands are included in the HTML code for a web page and are enclosed in <script> tags. The browser knows to run the JavaScript program because it's enclosed in these tags. It only works with HTML elements and can enhance the interactivity of a web page by creating HTML dynamically, validating form fields and performing basic calculations.

JavaScript is essentially an object-based event-driven language. An object is selected, triggering an event and a piece of JavaScript code is executed. For example, when a user clicks a button on a web page, the button object is selected, triggering a "click" event which may activate certain instructions.

For JavaScript purposes, objects are defined as computer entities that can be referenced in code. The major web objects are document, elements, form, frame, image, link, window, history and navigator. Each object consists of properties, methods and events.

A property is an attribute or characteristic of an object, and can be used to modify an object. Each object has its own specific properties. Some properties exist for several different objects, while other properties only exist for certain objects.

JavaScript is:

- JavaScript is a lightweight, interpreted programming language
- Designed for creating network - centric applications
- Complementary to and integrated with Java
- Complementary to and integrated with HTML
- Open and cross - platform

**Advantages of JavaScript:** The merits of using JavaScript are:

**Less server interaction:** You can validate user input before sending the page off to the server. This saves server traffic, which means fewer loads on your server.

## UNIT-I

---

**Immediate feedback to the visitors:** They don't have to wait for a page reload to see if they have forgotten to enter something.

**Increased interactivity:** You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

**Richer interfaces:** You can use JavaScript to include such items as drag – and – drop components and sliders to give a Rich Interface to your site visitors.

### **Limitations with JavaScript:**

We cannot treat JavaScript as a fully fledged programming language. It lacks the following important features:

- Client – side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessing capabilities.
- Once again, JavaScript is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

### **JavaScript Development Tools:**

**Microsoft FrontPage:** Microsoft has developed a popular HTML editor called FrontPage. FrontPage also provides web developers with a number of JavaScript tools to assist in the creation of an interactive web site.

**Macromedia Dreamweaver MX:** Macromedia Dreamweaver MX is a very popular HTML and JavaScript editor in the professional web development crowd. It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.

**Macromedia Home Site 5:** This provided a well - liked HTML and JavaScript editor, which will manage their personal web site just fine.

A JavaScript consists of JavaScript statements that are placed within the <script>... </script> HTML tags in a webpage. You can place the <script> tag containing your JavaScript anywhere within you web page but it is preferred way to keep it within the <head> tags.

## UNIT-I

---

The `<script>` tag alerts the browser program to begin interpreting all the text between these tags as a script. So simple syntax of your JavaScript will be as follows Syntax:

```
<script ...>  
JavaScript code  
</script>
```

The script tag takes two important attributes:

### **language:**

This attribute specifies what scripting language you are using. Typically, its value will be JavaScript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.

### **type:**

This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/JavaScript".

So your JavaScript segment will look like:

```
<script language="javascript" type="text/javascript">  
JavaScript code  
</script>
```

Your First JavaScript code:

Let us write our class example to print out "Hello World".

```
<html>  
<body>  
<script language="javascript" type="text/javascript">  
<!--  
document.write("Hello World!");  
// -->  
</script>  
</body>  
</html>
```

# UNIT-I

---

## OBJECTS IN JAVASCRIPT:

Almost everything in JavaScript can be an Object: Strings, Functions, Arrays, Dates.... Objects are just data, with properties and methods.

### Properties and Methods:

Properties are values associated with objects.

Methods are actions that objects can perform.

For example consider a Real Life Object like A Car:

Properties	Methods
car.name = Fiat	car.start()
car.model = 500	car.drive()
car.weight = 850kg	car.break()
car.color = white	

The properties of the car include name, model, weight, color, etc. All cars have these properties, but the property values differ from car to car.

The methods of the car could be start(), drive(), brake(), etc. All cars have these methods, but they are performed at different times.

*In JavaScript, objects are data (variables), with properties and methods. You create a JavaScript String object when you declare a string variable like this:*

```
var txt = new String("Hello World");
```

String objects have built-in properties and methods:

Object	Property	Method
"Hello World"	txt.length	txt.indexOf("World")

The string object above has a length property of 11, and the indexOf("World") method will return 6. In object oriented languages, properties and methods are often called object members.

**Creating JavaScript Objects:** Everything in JavaScript can be treated as objects, like Strings, Dates, Arrays, Functions, etc... You can also create your own objects. This example creates an object called "person", and adds four properties to it:



## UNIT-I

---

```
<!DOCTYPE html>  
<html>  
<body>  
<script>  
var person=new Object();  
person.firstname="John";  
person.lastname="Doe";  
person.age=50;  
person.eyecolor="blue";  
document.write(person.firstname + " is " + person.age + " years old.");  
</script>  
</body>  
</html>
```

This shows the following result:

John is 50 years old.

### Accessing Object Properties:

The syntax for accessing the property of an object is:

```
objectName.propertyName
```

This example uses the length property of the String object to find the length of a string:

```
var message="Hello World!";  
var x=message.length;
```

The value of x, after execution of the code above will be:

12

### Accessing Object Methods:

You can call a method with the following syntax:

```
objectName.methodName()
```

This example uses the toUpperCase() method of the String object, to convert a text to uppercase:

## UNIT-I

---

```
var message="Hello world!";  
var x=message.toUpperCase();
```

The value of x, after execution of the code above will be:

HELLO WORLD!

### **DYNAMIC HTML WITH JAVASCRIPT:**

Dynamic HTML is the combination of CSS, DOM, scripting. DHTML really is just JavaScript. The Document Object Model or DOM suggests that we think about a document as a collection of objects.

In case of the web documents tags are same as objects. Every <p> tag makes an HTML paragraph object, every <form> tag a form object

Web authors today face significant challenges when making their Web pages interactive. The static nature of HTML pages limits their creative choices, and interactive components can be difficult to build and reuse. In addition, using proprietary extensions means authoring browser specific Web pages.

Microsoft® Dynamic HTML technology helps to remove these barriers for content providers and offers users more engaging and interactive Web pages. Dynamic HTML provides authors with enhanced creative control so they can manipulate any page element at any time.

Dynamic HTML is also the easiest way to make Web pages interactive, using open, standards-based technologies. Microsoft is working with the World Wide Web Consortium (W3C) to help ensure interoperability and support for users on multiple systems with different browsers.

Dynamic HTML builds upon existing HTML standards to expand the possibilities of Web page design, presentation, and interaction. Ultimately, mastering DHTML will allow you to build Web based applications, rather than mere portraits of data. Because DHTML is essentially an "added value" technology, you should be rather familiar with basic Web page design using traditional HTML specifications. Experience with JavaScript programming is also necessary to employ the potential of DHTML.

Making simple updates, such as changing the color of text after a Web page loads, traditionally has meant reloading the page. These limitations have slowed the user experience and have impeded interactivity on the Web.

## UNIT-I

---

Microsoft's Dynamic HTML takes interactivity to the next level. Pages authored with Dynamic HTML come alive, with every element in the page being truly dynamic. Whether the page has loaded already, content providers can change any element of the page -- text or graphics -- without a round-trip to the server. This increased control and flexibility result in more compelling sites. For users, the Web experience becomes more responsive.

*Key features of Dynamic HTML include these:*

**Document Object Model (DOM).** Dynamic HTML provides a comprehensive object model for HTML. This model exposes all page elements as objects. Changing their attributes or applying methods to them at any time can easily manipulate these objects.

Dynamic HTML also provides full support for keyboard and mouse events on all page elements. Support for the Document Object Model enables the following:

**Dynamic content:** Text or graphics can be added, deleted, or modified on the fly. For example, a Web page can display an updated headline, without refreshing the page. The text surrounding the headline will reflow automatically

**Dynamic Styles:** Internet Explorer 4.0 fully supports Cascading Style Sheets (CSS). As such, any CSS attribute, including color and font, can be updated without a server roundtrip. For instance, text can change color or size when a mouse pointer passes over it. Multimedia filters and transition effects can be applied to HTML elements simply by adding the filter CSS attribute.

**Absolute positioning:** CSS positioning coordinates for existing page content can be updated at any time to create animated effects, without reloading the page.

**Data Binding:** Data-driven application front ends can be built that present, manipulate (e.g., sort, filter), and update data on the client without numerous round-trips to the server.

**Scriptlets:** A scriptlet is a Web page, authored with Dynamic HTML, which content providers can use as a component in their Web applications. With Scriptlets, content providers can author content once, and then easily reuse the content in other Web pages or applications.

Adding dynamic behavior to Web pages formerly required writing complex applets or controls, and incorporating them in Web pages using scripts. Although these components perform useful tasks, many authors found them hard to develop compared with scripts or HTML.

## UNIT-I

---

Microsoft Dynamic HTML was designed so that Web builders can use the scripting languages they know today -- such as JavaScript and the Microsoft Visual Basic® programming system, Visual Basic Scripting Edition (VBScript) -- to make their Web pages interactive.

Developers can also write full-featured Web applications with controls and applets that use Dynamic HTML. Web builders can easily reuse Dynamic HTML-based content through support for Scriptlets in Microsoft Internet Explorer 4.0, using just HTML and script.

Dynamic HTML will be featured in all versions of Microsoft Internet Explorer 4.0, including versions for the Windows® operating system and for the Macintosh and UNIX platforms. Microsoft's implementation of Dynamic HTML will also be available free of charge as a component for third-party use.

JKDIRECTORY