**INTRODUCTION ABOUT HTML 5.0**

## Html 1.0 (1989-1994):

* The First version of HTML that supports inline images and text controls.HTML 1.0 was limited in terms of styling and presentation of content.

EX: Specifying fonts, changing page background.

* Support of these came in HTML 2.0. But HTML 2.0 is missing some of the Netscape/Microsoft extensions and did not support tables (or) align attributes.

* HTML 3.2 was the next official version integrating support for tables, image, heading and other element align attributes. It is also missing some of the Microsoft extensions such as frames , applet.
* Support for these came in HTML 4.0.
* HTML 4.0 supports extra features (i.e) CSS, extra table, form and javascript.
* HTML 4.0 did not support audio and video files. For supporting these HTML version 5 was introduced.

**II. Basic Syntax of HTML5.0 and the Differences b/w Html4.0**

**&Html 5.0**

|  |  |
| --- | --- |
| **Html** | **Html5** |
| Doctype declaration in Html is too longer <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"> | DOCTYPE declaration in Html5 is very simple "<!DOCTYPE html> |
| character encoding in Html is also longer  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"> | character encoding (charset) declaration is also very simple <meta charset="UTF-8"> |
| Audio and Video are not part of HTML4 | Audio and Videos are integral part of HTML5 e.g. <audio> and <video> tags. |
| Vector Graphics is possible with the help of technologies such as VML, Silverlight, Flash etc | Vector graphics is integral part of HTML5 e.g. SVG(Scalable Vector Graphics) and canvas. |
| It is almost impossible to get true GeoLocation of user browsing any website especially if it comes to mobile devices. | JS GeoLocation API in HTML5 helps identify location of user browsing any website (provided user allows it) |
| Html5 use cookies. | It provides local storage in place of cookies. |
| Not possible to draw shapes like circle, rectangle, triangle. | Using Html5 you can draw shapes like circle, rectangle, triangle. |
| Does not allow JavaScript to run in browser. JS runs in same thread as browser interface. | Allows JavaScript to run in background. This is possible due to JS Web worker API in HTML5 |
| Works with all old browsers | Supported by all new browser. |

## HTML <!DOCTYPE> Declaration

The DOCTYPE statement indicates the particular version of HTML being used in the document. It is used to check the document for validity. DOCTYPE declaration is not an HTML tag. It is only a declaration and always appears at the first line of your HTML documents.

**Definition of HTML:**

HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page.

The markup tells the Web browser how to display a Web page's words and images for the user. Each individual markup code is referred to as an element (but many people also refer to it as a tag).

HiperText: Specially designed text for Webbrowser.

Markup: With meaningful tags or elements

Language: In Simple English Language.

**Basic structure of an HTML document**

An HTML document has two main parts:

1. *head*. The head [element](http://www.scriptingmaster.com/html/HTML-elements.asp) contains title and meta data of a web document.
2. *body*. The body element contains the information that you want to display on a web page.

To make your web pages compatible with HTML 4, you need to add a document type declaration (DTD) before the HTML element. Many web authoring software add DTD and basic tags automatically when you create a new web page.

In a web page, the first tag (specifically, <html>) indicates the markup language that is being used for the document. The [<head> tag](http://www.scriptingmaster.com/html/HTML-meta-tags.asp) contains information about the web page. Lastly, the content appears in the <body> tag. The following illustration provides a summary.

|  |
| --- |
| **Summary of Basic HTML Tags** |

**HTML Tags**

Tags are instructions that are embedded directly into the text of a HTML document. Each HTML tag describes that the browser should do something instead of simply displaying the text. In HTML, the tags begin with (<) and end with (>)  
HTML tags can be of two types. They are

1. Paired Tags  
2. Unpaired Tags

**Paired Tags:**

A tag is said to be a paired tag if the text is placed between a tag and its companion tag. In paired tags, the first tag is referred to as *Opening Tag* and the second tag is referred to as *Closing Tag.*

**Example:**

<i>This text is in italics. </i>

**Note:** Here <i> is called opening tag. and </i> is called closing tag.

**Unpaired Tags:**

An unpaired tag does not have a companion tag. Unpaired tags are also known as *Singular* or *Stand-Alone* Tags.  
**Example**

<br> , <hr> etc. These tags does not require any companion tag.

## HTML Formatting Elements

HTML also defines special **elements** for defining text with a special **meaning**.

Formatting elements were designed to display special types of text:

* <b> - Bold text
* <strong> - Important text
* <i> - Italic text
* <em> - Emphasized text
* <mark> - Marked text
* <small> - Small text
* <del> - Deleted text
* <ins> - Inserted text
* <sub> - Subscript text
* <sup> - Superscript text

## HTML <b> and <strong> Elements

The HTML **<b>** element defines **bold** text, without any extra importance.

### Example:

<b>This text is bold</b>

The HTML **<strong>** element defines **strong** text, with added semantic "strong" importance.

### Example:

<strong>This text is strong</strong>

## HTML <i> and <em> Elements

The HTML **<i>** element defines italic text, without any extra importance.

### Example:

<i>This text is italic</i>

The HTML **<em>** element defines emphasized text, with added semantic importance.

### Example:

<em>This text is emphasized</em>

**Note:** Browsers display <strong> as <b>, and <em> as <i>. However, there is a difference in the meaning of these tags: <b> and <i> defines bold and italic text, but <strong> and <em> means that the text is "important".

## HTML <small> Element

The HTML **<small>** element defines smaller text:

## Example:

<h2>HTML <small>Small</small> Formatting</h2>

## HTML <del> Element

The HTML **<del>** element defines  (removed) text.

**Example:**

<p>My favorite color is <del>blue</del> red.</p>

## HTML <ins> Element

The HTML **<ins>** element defines inserted (added) text.

**Example:**

<p>My favorite <ins>color</ins> is red.</p>

## HTML <sub> Element

The HTML **<sub>** element defines subscripted text.

**Example:**

<p>This is <sub>subscripted</sub> text.</p>

## HTML <sup> Element

The HTML **<sup>** element defines superscripted text.

**Example:**

<p>This is <sup>superscripted</sup> text.</p>

# HTML Images

Images can improve the design and the appearance of a web page.

### Example:

<img src="pulpitrock.jpg" alt="Mountain View">

## HTML Images Syntax

In HTML, images are defined with the **<img>** tag.

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The src attribute specifies the URL (web address) of the image:

<img src="*url*">

## The alt Attribute

The alt attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

The value of the alt attribute should describe the image:

### Example

<img src="img\_chania.jpg" alt="Flowers in Chania">

## Image Size - Width and Height

You can use the **style** attribute to specify the width and height of an image.

### Example

<img src="img\_girl.jpg" alt="Girl in a jacket" style="width:500px;height:600px;">

Alternatively, you can use the **width** and **height** attributes:

### Example

<img src="img\_girl.jpg" alt="Girl in a jacket" width="500" height="600">

The width and height attributes always defines the width and height of the image in pixels.

**Note:** Always specify the width and height of an image. If width and height are not specified, the page might flicker while the image loads.

## Width and Height, or Style?

Both the width, height, and style attributes are valid in HTML5.

However, we suggest using the style attribute. It prevents styles sheets from changing the size of images:

<!DOCTYPE html>  
<html>  
<head>  
<style>  
img {   
    width:100%;   
}  
</style>  
</head>  
<body>  
  
<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">  
<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">  
  
</body>  
</html>

# HTML Lists

HTML offers three ways for specifying lists of information. All lists must contain one or more list elements. They are −

* **<ul>** − An unordered list. This will list items using plain bullets.
* **<ol>** − An ordered list. This will use different schemes of numbers to list your items.
* **<dl>** − A definition list. This arranges 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.

This list is created by using HTML **<ul>** tag. Each item in the list is marked with a bullet.

### Example

<!DOCTYPE html>

<html>

<head>

<title>HTML Unordered List</title>

</head>

<body>

<ul>

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ul>

</body>

</html>

## The type Attribute

You can use **type** attribute for <ul> tag to specify the type of bullet you like. By default, it is a disc.

Following are the possible options −

<ul type = "square">

<ul type = "disc">

<ul type = "circle">

### Example

Following is an example where we used <ul type = "square">

<!DOCTYPE html>

<html>

<head>

<title>HTML Unordered List</title>

</head>

<body>

<ul type = "square">

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ul>

</body>

</html>

This will produce the following result −

* Beetroot
* Ginger
* Potato
* Radish

## HTML Ordered Lists

* If you are required to put your items in a numbered list instead of bulleted, then HTML ordered list will be used.
* This list is created by using **<ol>** tag.
* The numbering starts at one and is incremented by one for each successive ordered list element tagged with <li>.

### Example

<!DOCTYPE html>

<html>

<head>

<title>HTML Ordered List</title>

</head>

<body>

<ol>

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ol>

</body>

</html>

This will produce the following result –

1. Beetroot
2. Ginger
3. Potato
4. Radish

## The type Attribute

You can use **type** attribute for <ol> tag to specify the type of numbering you like. By default, it is a number.

Following are the possible options –

<ol type = "1"> - Default-Case Numerals.

<ol type = "I"> - Upper-Case Numerals.

<ol type = "i"> - Lower-Case Numerals.

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

<ol type = "a"> - Lower-Case Letters.

### Example

Following is an example where we used <ol type = "1">

[Live Demo](http://tpcg.io/1IgyrU)

<!DOCTYPE html>

<html>

<head>

<title>HTML Ordered List</title>

</head>

<body>

<ol type = "1">

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ol>

</body>

</html>

This will produce the following result −

1. Beetroot
2. Ginger
3. Potato
4. Radish

## The start Attribute

You can use **start** attribute for <ol> tag to specify the starting point of numbering you need.

Following are the possible options −

<ol type = "1" start = "4"> - Numerals starts with 4.

<ol type = "I" start = "4"> - Numerals starts with IV.

<ol type = "i" start = "4"> - Numerals starts with iv.

<ol type = "a" start = "4"> - Letters starts with d.

<ol type = "A" start = "4"> - Letters starts with D.

### Example

Following is an example where we used <ol type = "i" start = "4" >

<!DOCTYPE html>

<html>

<head>

<title>HTML Ordered List</title>

</head>

<body>

<ol type = "i" start = "4">

<li>Beetroot</li>

<li>Ginger</li>

<li>Potato</li>

<li>Radish</li>

</ol>

</body>

</html>

This will produce the following result −

1. Beetroot
2. Ginger
3. Potato
4. Radish

## HTML Definition Lists

* HTML and XHTML supports a list style which is called **definition lists** where entries are listed like in a dictionary or encyclopedia.
* 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 four tags.
* <dl> − Defines the start of the list
* <dt> − A term
* <dd> − Term definition
* </dl> − Defines the end of the list

### Example

<!DOCTYPE html>

<html>

<head>

<title>HTML Definition List</title>

</head>

<body>

<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>

</body>

</html>

This will produce the following result −

**HTML**

This stands for Hyper Text Markup Language

**HTTP**

This stands for Hyper Text Transfer Protocol

## HTML Links – Hyperlinks

A webpage can contain various links that take you directly to other pages and even specific parts of a given page. These links are known as hyperlinks.

Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images. Thus you can create hyperlinks using text or images available on a webpage.

## Linking Documents

* A link is specified using HTML tag <a>.
* This tag is called **anchor tag** and anything between the opening <a> tag and the closing </a> 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 <a> tag.

<a href = "Document URL" ... attributes-list>Link Text</a>

### Example

Let's try following example which links http://www.tutorialspoint.com at your page –

<!DOCTYPE html>

<html>

<head>

<title>Hyperlink Example</title>

</head>

<body>

<p>Click following link</p>

<a href = "https://www.google.com" target = "\_self">Google Page</a>

</body>

</html>

**Href Attribute**

* The href attribute specifies the destination address (https://www.google.com) of the link.
* The link text is the visible part .

## The target Attribute

## This attribute is used to specify the location where linked document is opened. Following are the possible options −

|  |  |
| --- | --- |
| **Sr.No** | **Option & Description** |
| 1 | **\_blank**  Opens the linked document in a new window or tab. |
| 2 | **\_self**  Opens the linked document in the same frame. |
| 3 | **\_parent**  Opens the linked document in the parent frame. |
| 4 | **\_top**  Opens the linked document in the full body of the window. |
| 5 | **Targetframe**  Opens the linked document in a named *targetframe*. |

**Table Tag:**

* The HTML allow web authors to arrange data like text, images, links, other tables, etc.
* To arrange the data into rows and columns we can use the “Table Tag”.
* The HTML tables are created using the **<table>** tag.
* The table tag consist 3 tags.

(1) <tr> tag

(2) <td> tag

(3) <th> tag

  (1)**<tr> tag**: It is used to create table rows.

(2)**<td> tag**: It is used to create data cell or to insert the data into

the table.

**Note:** The elements under <td> are regular and left aligned by default.

  (3)**<th> tag:** It is used to define the header(or) column of the table.

**Note:**

(i)By default, table headings are bold and centered.

(ii) The <td> elements are the data containers of the table.

(iii)They can contain all sorts of HTML elements; text, images, lists,

other tables, etc.

**Example:**

<!DOCTYPE html>

<html>

<head>

<title> Table Tag</title>

<h1> Employee Details</h1>

</head>

<body>

<table style="width:70%">

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

</tr>

<tr>

<td>Jill</td>

<td>Smith</td>

<td>50</td>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

</tr>

<tr>

<td>John</td>

<td>Doe</td>

<td>80</td>

</tr>

</table>

</body>

</html>

## Adding a Border to theTable

* ->If we do not specify a border for the table, it will be displayed without borders.
* A border can be set using the CSS **border** property:

i.e

<style>

table, th, td

  {  
    border: 1px solid black;  
 }

</style>

**Example:**

<!DOCTYPE html>

<html>

<head>

<title>Defining the Table</title>

<h1>Employee Details</h1>

<style>

table, th, td

{

border: 1px solid red;

}

</style>

</head>

<body>

<table>

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

</tr>

<tr>

<td>Jill</td>

<td>Smith</td>

<td>50</td>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

</tr>

<tr>

<td>John</td>

<td>Doe</td>

<td>80</td>

</tr>

</table>

</body>

</html>

## Collapsed Borders to the Table:

If you want the borders to collapse into one border, add the “CSS **border-collapse”** property:

Example:

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

**border-collapse: collapse;**

}

</style>

</head>

<body>

<table style="width:100%">

<tr>

<th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

</tr>

<tr>

<td>Jill</td>

<td>Smith</td>

<td>50</td>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

</tr>

<tr>

<td>John</td>

<td>Doe</td>

<td>80</td>

</tr>

</table>

</body>

</html>

## HTML Table - Left-align Headings

->By default, table headings are bold and centered.

->To left-align the table headings, use the CSS **text-align** property:

**Syntax:**

th {  
    text-align: left;  
 }

**Cells that Span Many Columns**

->To make a cell span more than one column, use the **colspan** attribute:

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

border-collapse: collapse;

}

th, td {

padding: 5px;

text-align: left;

}

</style>

</head>

<body>

<h2>Cell that spans two columns:</h2>

<table style="width:100%">

<tr>

<th>Name</th>

<th colspan="2">Telephone</th>

</tr>

<tr>

<td>Bill Gates</td>

<td>55577854</td>

<td>55577855</td>

</tr>

</table>

</body>

</html>

## Cells that Span Many Rows

To make a cell span more than one row, use the **rowspan** attribute:

### Example:

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

border-collapse: collapse;

}

th, td {

padding: 5px;

text-align: left;

}

</style>

</head>

<body>

<h2>Cell that spans two rows:</h2>

<table style="width:100%">

<tr>

<th>Name:</th>

<td>Bill Gates</td>

</tr>

<tr>

<th rowspan="2">Telephone:</th>

<td>55577854</td>

</tr>

<tr>

<td>55577855</td>

</tr>

</table>

</body>

</html>

## Adding a Caption to the Table

->To add a caption to a table, we can use the **<caption>** tag:

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

border-collapse: collapse;

}

th, td {

padding: 5px;

text-align: left;

}

</style>

</head>

<body>

<table style="width:100%">

<caption>Monthly savings</caption>

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

<tr>

<td>February</td>

<td>$50</td>

</tr>

</table>

</body>

</html>

**Note:** The <caption> tag must be inserted immediately after the <table> tag.

# Frames

HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document.

A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

## Creating Frames

To use frames on a page we use <frameset> tag instead of <body> tag. The <frameset> tag defines, how to divide the window into frames. The **rows** attribute of <frameset> tag defines horizontal frames and **cols** attribute defines vertical frames. Each frame is indicated by <frame> tag and it defines which HTML document shall open into the frame.

**Note** − The <frame> tag deprecated in HTML5. Do not use this element.

### Example

Following is the example to create three horizontal frames −

<!DOCTYPE html>

<html>

<head>

<title>HTML Frames</title>

</head>

<frameset rows = "10%,80%,10%">

<frame name = "top" src = "/html/top\_frame.htm" />

<frame name = "main" src = "/html/main\_frame.htm" />

<frame name = "bottom" src = "/html/bottom\_frame.htm" />

<noframes>

<body>Your browser does not support frames.</body>

</noframes>

</frameset>

</html>

### Example

Let's put the above example as follows, here we replaced rows attribute by cols and changed their width. This will create all the three frames vertically –

<!DOCTYPE html>

html>

<head>

<title>HTML Frames</title>

</head>

<frameset cols = "25%,50%,25%">

<frame name = "left" src = "/html/top\_frame.htm" />

<frame name = "center" src = "/html/main\_frame.htm" />

<frame name = "right" src = "/html/bottom\_frame.htm" />

<noframes>

<body>Your browser does not support frames.</body>

</noframes>

</frameset>

</html>

## The <frameset> Tag Attributes

Following are important attributes of the <frameset> tag −

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **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 the 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. |
| 2 | **Rows**  This attribute works just like the cols attribute and takes 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. |
| 3 | **Border**  This attribute specifies the width of the border of each frame in pixels. For example, border = "5". A value of zero means no border. |
| 4 | **Frameborder**  This attribute 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. |
| 5 | **Framespacing**  This attribute 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. |

## The <frame> Tag Attributes

Following are the important attributes of <frame> tag −

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **Src**  This attribute is used to give the file name that should be loaded 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. |
| 2 | **Name**  This 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 an another frame, in which case the second frame needs a name to identify itself as the target of the link. |
| 3 | **Frameborder**  This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no). |
| 4 | **Marginwidth**  This attribute 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". |
| 5 | **Marginheight**  This attribute 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". |
| 6 | **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". |
| 7 | **Scrolling**  This attribute 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. |

## Browser Support for Frames

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.

So 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 <frameset> element then it should understand what is inside the <body> element which is contained in a <noframes> element.

You can put some nice message for your user having old browsers. For example, *Sorry!! your browser does not support frames.* as shown in the above example.

## 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 main pages into a separate frame.

Let's see following example where a test.htm file has following code −

<!DOCTYPE html>

<html>

<head>

<title>HTML Target Frames</title>

</head>

<frameset cols = "200, \*">

<frame src = "/html/menu.htm" name = "menu\_page" />

<frame src = "/html/main.htm" name = "main\_page" />

<noframes>

<body>Your browser does not support frames.</body>

</noframes>

</frameset>

</html>

Here, we have created two columns to fill with two frames. The first frame is 200 pixels wide and will contain the navigation menu bar implemented by **menu.htm** file. The second column fills in remaining space and will contain the main part of the page and it is implemented by **main.htm** file. For all the three links available in menu bar, we have mentioned target frame as **main\_page**, so whenever you click any of the links in menu bar, available link will open in main page.

Following is the content of menu.htm file

<!DOCTYPE html>

<html>

<body bgcolor = "#4a7d49">

<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>

</body>

</html>

Following is the content of main.htm file −

<!DOCTYPE html>

<html>

<body bgcolor = "#b5dcb3">

<h3>This is main page and content from any link will be displayed here.</h3>

<p>So now click any link and see the result.</p>

</body>

</html>

When we load **test.htm** file, it produces following result −

Now you can try to click links available in the left panel and see the result.

The *targetattribute* can also take one of the following values −

|  |  |
| --- | --- |
| **Sr.No** | **Option & Description** |
| 1 | **\_self**  Loads the page into the current frame. |
| 2 | **\_blank**  Loads a page into a new browser window. Opening a new window. |
| 3 | **\_parent**  Loads the page into the parent window, which in the case of a single frameset is the main browser window. |
| 4 | **\_top**  Loads the page into the browser window, replacing any current frames. |
| 5 | **Targetframe**  Loads the page into a named targetframe. |

## Disadvantages of Frames:

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages −

* Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
* Sometimes 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 that do not support frame technology.

**HTML Forms**

An HTML form is a section of document that contains interactive controls that enable a user to submit information to a web server.

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called controls like inputbox, checkboxes, radio-buttons, submit buttons, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for processing.

->The **“<form>”** tag is used to create an HTML form.

-> Here's a simple example of a login form:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Example of HTML Form Controls</title>

</head>

<body>

<!-- Browser may generate "Not secure" warning due to presence of password field in insecure form. We've created this form just for demo purpose. -->

<form>

<fieldset>

<legend>Sign In</legend>

<label for="user-name">Username:</label>

<input type="text" name="username" id="user-name">

<label for="user-pwd">Password:</label>

<input type="password" name="user-password" id="user-pwd">

</fieldset>

<fieldset>

<legend>Gender</legend>

<input type="radio" name="sex" id="male">

<label for="male">Male</label>

<input type="radio" name="sex" id="female">

<label for="female">Female</label>

</fieldset>

<fieldset>

<legend>Hobbies</legend>

<input type="checkbox" name="sports" id="soccer">

<label for="soccer">Soccer</label>

<input type="checkbox" name="sports" id="cricket">

<label for="cricket">Cricket</label>

<input type="checkbox" name="sports" id="cricket">

<label for="baseball">Baseball</label>

</fieldset>

<fieldset>

<legend>Address</legend>

<textarea rows="3" cols="30"></textarea>

</fieldset>

<fieldset>

<legend>Upload file</legend>

<label for="file-select">Upload:</label>

<input type="file" name="upload" id="file-select">

</fieldset>

<fieldset>

<legend>Select Your City</legend>

<label for="city">City:</label>

<select name="city" id="city">

<option value="sydney">Sydney</option>

<option value="melbourne">Melbourne</option>

<option value="cromwell">Cromwell</option>

</select>

</fieldset>

<fieldset>

<legend>Action</legend>

<input type="submit" value="Submit">

<input type="reset" value="Reset">

</fieldset>

</form>

</body>

</html>

# <fieldset> Tag

The <fieldset> tag is used to group related elements in a form.

The <fieldset> tag draws a box around the related elements.

[**<legend>**](https://www.w3schools.com/tags/tag_legend.asp)**tag**

# The [<legend>](https://www.w3schools.com/tags/tag_legend.asp) tag defines a caption for the <fieldset> element.

## <select> Tag:

## <select> element define each list item.

## Example:

## <form>

## <label for="city">City:</label>

## <select name="city" id="city">

## <option value="sydney">Sydney</option>

## <option value="melbourne">Melbourne</option>

## <option value="cromwell">Cromwell</option>

## </select>

## </form>

## Submit and Reset Buttons

## A submit button is used to send the form data to a web server. When submit button is clicked the form data is sent to the file specified in the form's action attribute to process the submitted data.

## A reset button resets all the forms control to default values.

## Example:

## <form action="action.php" method="post" id="users">

## <label for="first-name">First Name:</label>

## <input type="text" name="first-name" id="first-name">

## <input type="submit" value="Submit">

## <input type="reset" value="Reset">

## </form>

## Most frequently used form attributes are:

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Name | The name of the form. |
| Action | URL of the program that processes the information submitted via form. |
| Method | The HTTP method that the browser uses to submit the form. Possible values are get and post. |
| Target | A name or keyword indicating the target page where the result of the script will be displayed. The reserved keywords are \_blank, \_self, \_parent and \_top. |

## HTML5 introduces several new input types for forms.

## New Input Types in HTML5

HTML5 introduces several new [<input>](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php) types like email, date, time, color, range, etc. to improve the user experience and to make the forms more interactive.

However, if a browser failed to recognize these new input types, it will treat them like a normal text box.

**They are:**

* [color](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#color-input)
* [date](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#date-input)
* [datetime](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#datetime-input)
* [datetime-local](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#datetime-local-input)
* [email](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#email-input)
* [month](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#month-input)
* [number](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#number-input)
* [range](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#range-input)
* [search](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#search-input)
* [tel](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#tel-input)
* [time](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#time-input)
* [url](https://www.tutorialrepublic.com/html-tutorial/html5-new-input-types.php#url-input)

## Input Type Color

The color input type allows the user to select a color from a drop-down color picker and returns the hex value for that color.

**Example:**

<form>  
  Select your favorite color:  
  <input type="color" name="favcolor">  
</form>

## Input Type Date

The date input type allows the user to select a date from a drop-down calendar.

**Example:**

<form>

# <label>

# Select Date: <input type="date" name="mydate">

# </label>

# </form>

# Input Type Datetime

# The datetime input type allows the user to select a date and time along with time zone.

# Example:

# <form>

# <label>

# Date & Time: <input type="datetime" name="mydatetime">

# </label>

# </form>

# Input Type Datetime-local

# The datetime-local input type allows the user to select a local date and time. The local date and time doesn't provide timezone information.

# Example:

# <form>

# <label>

# Local Date & Time: <input type="datetime-local" name="mylocaldatetime">

# </label>

# </form>

# Input Type Email

# The email input type allows the user to enter e-mail address. It is very similar to a standard text input type, but if it used in combination with the required attribute, the browser may look for patterns to ensure a valid e-mail address should be entered.

# Example:

# <form>

# <label>

# Email Address: <input type="email" name="myemail" required>

# </label>

# </form>

# Note:The validation for the input type="email" is supported by all major

# browsers like Mozilla Firefox, Google Chrome, Apple Safari, Internet

# Explorer 10+ and Opera.

# Input Type Month

# The month input type allows the user to select a month and year from a drop-down calendar.

# Example:

# <form>

# <label>

# Select Month: <input type="month" name="mymonth">

# </label>

# </form>

# Input Type Number

# The number input type can be used for entering a numerical value. You can also restrict the user to enter only acceptable values using the additional attributes min, max, and step.

# Example:

# <form>

# <label>

# Select Number: <input type="number" value="1" min="1" max="10" step="0.5" name="mynumber">

# </label>

# </form>

# Input Type Range

# The range input type can be used for entering a numerical value within a specified range. It works very similar to number input, but it offer a simpler control for entering a number.

# Example:

# <form>

# <label>

# Select Number: <input type="range" value="1" min="1" max="10" step="0.5" name="mynumber">

# </label>

# </form>

# Input Type Search

# The search input type can be used for creating search fields.

# Example:

# <form>

# <label>

# Search Website: <input type="search" name="mysearch">

# </label>

# </form>

# Input Type Tel

# ->The tel input type can be used for entering a telephone number.

# Example:

# <form>

# <label>

# Telephone Number: <input type="tel" name="mytelephone" required>

# </label>

# </form>

# Input Type Time

# ->The time input type can be used for entering a time.

# Example:

# <form>

# <label>

# Select Time: <input type="time" name="mytime">

# </label>

# </form>

# Input Type URL

# ->The url input type can be used for entering web addresses i.e. URL's.

# Example:

# <form>

# <label>

# Website URL: <input type="url" name="mywebsite" required>

# </label>

# </form>

# Input Type Week

# ->The week input type allows the user to select a week and year from a drop-down calendar.

# Example:

# <form>

# <label>

# Select Week: <input type="week" name="myweek">

# </label>

# </form>

# <section> Tag

The <section> tag defines sections in a document, such as chapters, headers, footers, or any other sections of the document.

**Example:**

<!DOCTYPE html>

<html>

<body>

<section>

<h1>WWF</h1>

<p>The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation, research and restoration of the environment, formerly named the World Wildlife Fund. WWF was founded in 1961.</p>

</section>

<section>

<h1>WWF's Panda symbol</h1>

<p>The Panda has become the symbol of WWF. The well-known panda logo of WWF originated from a panda named Chi Chi that was transferred from the Beijing Zoo to the London Zoo in the same year of the establishment of WWF.</p>

</section>

<p><strong>Note:</strong> The section tag is not supported in Internet Explorer 8 and earlier versions.</p>

</body>

</html>

# <article> Tag

->The <article> tag specifies independent, self-contained content.

->An article should make sense on its own and it should be possible to distribute

it independently from the rest of the site.

->Potential sources for the <article> element:

* Forum post
* Blog post
* News story
* Comment

**Example:**

**<body>**

**<article>**

**<h1>The article title</h1>**

**<p>This is the contents of the article element.</p>**

**</article>**

**</body>**

# HTML Input Attributes

## 1)The value Attribute:

The **value** attribute specifies the initial value for an input field:

**Example:**

form action="">  
First name:<br>  
<input type="text" name="firstname" value="John">  
</form>

## 2)The readonly Attribute

The **readonly** attribute specifies that the input field is read only (cannot be changed):

**Example:**

<form action="">  
First name:<br>  
<input type="text" name="firstname" value="John" readonly>  
</form>

## 3)The disabled Attribute

* The **disabled** attribute specifies that the input field is disabled.
* A disabled input field is unusable and un-clickable, and its value will not be sent when submitting the form:

**Example:**

<form action="">  
 First name:<br>  
<input type="text" name="firstname" value="John" disabled>  
</form>

## 4)The size Attribute

The **size** attribute specifies the size (in characters) for the input field:

**Example:**

<form action="">  
First name:<br>  
<input type="text" name="firstname" value="John" size="40">  
</form>

## 5)The maxlength Attribute

The **maxlength** attribute specifies the maximum allowed length for the input field:

**Example:**

<form action="">  
First name:<br>  
<input type="text" name="firstname" maxlength="10">  
</form>

**Note:**

1. With a maxlength attribute, the input field will not accept more than the allowed number of characters.
2. The maxlength attribute does not provide any feedback. If you want to alert the user, you must write JavaScript code.

## HTML5 Attributes

HTML5 added the following attributes for <input>:

* autocomplete
* autofocus
* form
* formaction
* formenctype
* formmethod
* formnovalidate
* formtarget
* height and width
* list
* min and max
* multiple
* pattern (regexp)
* placeholder
* required
* step

**CHAPTER-II**

**CSS**

**Introduction:**

To apply the style and lay outs to the web pages — for example, to alter the font, colour, size and spacing of your content, split it into multiple columns, or add animations and other decorative features.For that purpose we can use “Cascading Style Sheets(CSS)”.

## CSS Versions

Cascading Style Sheets, level 1 (CSS1) was came out of W3C as a recommendation in December 1996. This version describes the CSS language as well as a simple visual formatting model for all the HTML tags.

CSS2 was became a W3C recommendation in May 1998 and builds on CSS1.This version adds support for media-specific style sheets e.g. printers and aural devices, downloadable fonts, element positioning and tables.

CSS3 was became a W3C recommendation in June 1999 and builds on older versions CSS. It has divided into documentations is called as Modules and here each module having new extension features defined in CSS2.

## Advantages of CSS

* **CSS saves time** − You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
* **Pages load faster** − If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
* **Easy maintenance** − To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
* **Superior styles to HTML** − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
* **Multiple Device Compatibility** − Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
* **Global web standards** − Now HTML attributes are being deprecated and it is being recommended to use CSS. So it is a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.
* **Offline Browsing** − CSS can store web applications locally with the help of an offline catche. Using of this, we can view offline websites. The cache also ensures faster loading and better overall performance of the website.
* **Platform Independence** −The Script offer consistent platform independence and can support latest browsers as well.

**Styling HTML with CSS**

* CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
* CSS **saves a lot of work**. It can control the layout of multiple web pages all at once.

->CSS can be added to HTML elements in 3 ways:

1. **Inline CSS** -

**Def:** Using the <style> attribute with in HTML elements or HTMLtags

is called “Inline CSS”.

* An Inline CSS is used to apply a unique style to a single HTML element.
* This example sets the text color of the <h1> element to blue:

**Example:**

<!DOCTYPE html>

<html>

<body>

<h1 style="color:blue;">This is a Blue Heading</h1>

</body>

</html>

1. **Internal** CSS-

**Def**: Using a <style> element in the <head> section is called “Internal

CSS”.

* An Internal CSS is used to define a style for “a single HTML page”.
* An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

**Example:**

<!DOCTYPE html>  
<html>  
<head>  
 <style>  
 body {background-color: powderblue;}  
 h1   {color: blue;}  
 p  {color: red;}  
 </style>  
 </head>  
<body>  
 <h1>This is a heading</h1>  
 <p>This is a paragraph.</p>  
</body>  
</html>

1. **External** CSS- Using a <style> element in an external CSS file is called

“External CSS”.

* An external style sheet is used to define the style for “many HTML pages”.
* **With an external style sheet, you can change the look of an entire web site, by changing one file.**

The most common way to add CSS, is to keep the styles in separate CSS files. However, here we will use inline and internal styling, because this is easier to demonstrate, and easier for you to try it yourself.

**Example:**

To use an external style sheet, add a link to it in the <head> section of the HTML page:

<!DOCTYPE html>  
 <html>  
 <head>  
   <link rel="Stylesheet" href="Styles.css">  
 </head>  
<body>  
 <h1>This is a heading</h1>  
 <p>This is a paragraph.</p>  
 </body>  
</html>

* An external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is how the **"Styles.css"** looks:

body {  
    background-color: red;  
}  
h1 {  
    color: blue;  
}  
p {  
    color: red;  
}

# Style Specification Formats:

* Format depends on the level of the style sheet
* [Inline:](http://cs.loc.edu/~chu/ITEC305/ch3/inline_ex.html)
  + Style sheet appears as the value of the style attribute
  + General form:

Style=”Property1:Value1;

Property2:Value2; ………

Propertyn:Valuen;

aluen;

* [Document-level (or) Internal-level:](http://cs.loc.edu/~chu/ITEC305/ch3/document_level.html)
  + Style sheet appears as a list of rules that are the content of a <style> tag
  + The <style> tag must include the type attribute, set to "text/css"
  + General form:

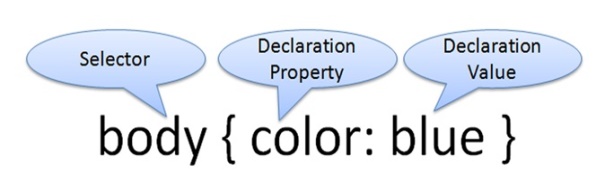
<style type=”text/css”>

Rule list

</style>

* [External style sheets](http://cs.loc.edu/~chu/ITEC305/ch3/external.html)
  + Form is a list of style rules, as in the content of a <style> tag for document-level style sheets
  + General Form:  
    selector {list of property/values}
    - Each property/value pair has the form:  
             property: value
    - Pairs are separated by semicolons, just as in the value of a <style> tag
  + Comments in the rule list must have a different form - use C comments (/\*...\*/)

**Example:**



## CSS Selectors

A CSS selector is the part of a CSS rule set that actually selects the content to which we want to apply the style. Let’s look at all the different kinds of selectors available, with a brief description of each.

## (i)Universal Selector

The “**universal selector”** works like a wild card character, selecting all elements on a page. Every HTML page is built on HTML tags. Each set of tags represents an element on the page.

-> CSS example, which uses the universal selector:

<style>

\* {

color: green;

font-size: 20px;

line-height: 25px;

}

</style>

**Note:** 1)The three lines of code inside the curly braces (color, font-size, and line-

height) will apply to all elements on the HTML page.

2) As seen here, the universal selector is declared using an “asterisk(\*)”.

3) You can also use the universal selector in combination with other selectors.

## (ii) Element Type Selector

Also referred to simply as a “type selector,” this selector must match one or more HTML elements of the same name.

**Example: A** selector of <ul> would match all HTML unordered lists,

or <ul>elements.

(i.e) The following example uses an element type selector to match all <ul> elements:

ul {

list-style: none;

border: solid 1px #fff;

}

To put this in some context, here’s a section of HTML to which we’ll apply the above CSS:

<ul>

<li>Fish</li>

<li>Apples</li>

<li>Cheese</li>

</ul>

<div class="example">

<p>Example paragraph text.</p>

</div>

<ul>

<li>Water</li>

<li>Juice</li>

<li>Maple Syrup</li>

</ul>

There are three main elements making up this part of the page:

Two <ul> elements and a <div>. The CSS will apply only to the two <ul> elements, and not to the <div>. Were we to change the element type selector to use <div> instead of <ul>, then the styles would apply to the <div> and not to the two <ul> elements.

Also note that the styles will not apply to the elements inside the <ul> or <div>elements. That being said, some of the styles may be inherited by those inner elements.

## (iii)ID Selector

An ID selector is declared using **“a hash, or pound symbol (**#**)”** preceding a string of characters. The string of characters is defined by the developer.

This selector matches any HTML element that has an ID attribute with the same value as that of the selector, but minus the hash symbol.

Here’s an example:

#container {

   width: 960px;

   margin: 0 auto;

}

This CSS uses an ID selector to match an HTML element such as:

<div id="container"></div>

In this case, the fact that this is a <div> element doesn’t matter—it could be any kind of HTML element. As long as it has an ID attribute with a value of container, the styles will apply.

An ID element on a web page should be unique. That is, there should only be a single element on any given page with an ID of container.

## (iv)Class Selector

The class selector is the most useful of all CSS selectors. **It’s declared with a dot preceding a string of one or more characters.**

The class selector also matches all elements on the page that have their class attribute set to the same value as the class, minus the dot.

Take the following rule set:

.box {

   padding: 20px;

   margin: 10px;

   width: 240px;

}

These styles will apply to the following HTML element:

<div class="box"></div>

The same styles will also apply to any other HTML elements that have a class attribute with a value of box. Having multiple elements on a single page with the same class attribute is beneficial, because it allows you to reuse styles, and avoid needless repetition.

Another reason the class selector is a valuable is that HTML allows multiple classes to be added to a single element. This is done by separating the classes in the HTML class attribute using spaces. Here’s an example:

<div class=”box box-more box-extended”></div>

## (v)Descendant Combinator

The descendant selector or,the descendant combinator combine two or more selectors. so we could be more specific in selection method.

For example:

#container .box {

   float: left;

   padding-bottom: 15px;

}

This declaration block will apply to all elements that have a class of box that are inside an element with an ID of container. It’s worth noting that the .box element doesn’t have to be an immediate child: there could be another element wrapping .box, and the styles would still apply.

Look at the following HTML:

<div id="container">

<div class="box"></div>

<div class="box-2"></div>

</div>

<div class="box"></div>

If we apply the CSS in the previous example to this section of HTML, the only element that’ll be affected by those styles is the first <div> element that has a class of box.

The <div> element that has a class of box-2 won’t be affected by the styles. Similarly, the second <div> element with a class of box won’t be affected because it’s not inside an element with an ID of container.

## (vi)Child Combinator

A selector that uses the child combinator is similar to a selector that uses a descendant combinator, except it only targets immediate child elements:

#container > .box {

   float: left;

   padding-bottom: 15px;

}

This is the same code from the descendant combinator example, but instead of a space character, we’re using the greater-than symbol (or right angle bracket.)

In this example, the selector will match all elements that have a class of box and that are immediate children of the #container element. That means, unlike the descendant combinator, there can’t be another element wrapping .box—it has to be a direct child element.

Here’s an HTML example:

<div id="container">

<div class="box"></div>

<div>

<div class="box"></div>

</div>

</div>

In this example, the CSS from the previous code example will apply only to the first <div> element that has a class of box. As you can see, the second <div> element with a class of box is inside another <div> element. As a result, the styles will not apply to that element, even though it too has a class of box.

Again, selectors that use this combinator can be somewhat restricting, but they can come in handy—for example, when styling nested lists.

## (vii)General Sibling Combinator

A selector that uses a general sibling combinator matches elements based on sibling relationships. That is to say, the selected elements are beside each other in the HTML.

h2 ~ p {

   margin-bottom: 20px;

}

->This selector is declared using the tilde character (~).

In this example, all paragraph elements (<p>) will be styled with the specified rules, but only if they are siblings of <h2> elements. There could be other elements in between the <h2> and <p>, and the styles would still apply.

Let’s apply the CSS from above to the following HTML:

<h2>Title</h2>

<p>Paragraph example.</p>

<p>Paragraph example.</p>

<p>Paragraph example.</p>

<div class="box">

<p>Paragraph example.</p>

</div>

In this example, the styles will apply only to the first three paragraph elements. The last paragraph element is not a sibling of the <h2> element because it sits inside the <div> element.

## (viii)Adjacent Sibling Combinator

A selector that uses the adjacent sibling combinator uses the plus symbol (+), and is almost the same as the general sibling selector.

The difference is that the targeted element must be an immediate sibling, not just a general sibling.

**Example**:

p + p {

   text-indent: 1.5em;

   margin-bottom: 0;

}

This example will apply the specified styles only to paragraph elements that immediately follow other paragraph elements. This means the first paragraph element on a page would not receive these styles. Also, if another element appeared between two paragraphs, the second paragraph of the two wouldn’t have the styles applied.

So, if we apply this selector to the following HTML:

<h2>Title</h2>

<p>Paragraph example.</p>

<p>Paragraph example.</p>

<p>Paragraph example.</p>

<div class="box">

<p>Paragraph example.</p>

<p>Paragraph example.</p>

</div>

…the styles will apply only to the second, third, and fifth paragraphs in this section of HTML.

## (ix) Attribute Selector

The attribute selector targets elements based on the presence and/or value of HTML attributes, and is declared using square brackets:

input[type="text"] {

   background-color: #444;

   width: 200px;

}

There should not be a space before the opening square bracket unless you intend to use it along with a descendant combinator.

**Example:**

The above CSS would match the following element:

<input type="text">

* But it wouldn’t match to the following code:

<input type="submit">

The attribute selector can also be declared using just the attribute itself, with no value, like this:

input[type] {

   background-color: #444;

   width: 200px;

}

This will match all input elements with an attribute of type, regardless of the value.

## (x)Pseudo-class

A pseudo-class uses **“a colon”** character to identify a pseudo-state that an element in Html page.

**Example:**

a:hover {

   color: red;

}

In this case, the pseudo-class portion of the selector is the :hover(Select and style a link when you mouse over it) part. Here we’ve attached this pseudo-class to all anchor elements ( elements). This means that when the user hovers their mouse over an element, the color property for that element will change to red.

This type of pseudo-class is a dynamic pseudo-class, because it occurs only in response to user interaction—in this case, the mouse moving over the targeted element.

**FONT PROPERTIES IN CSS3:**

The font property in CSS is a shorthand property that combines all the following sub-properties in a single declaration.

body {

font: normal small-caps 16px 1.4 Georgia;

}

/\* is the same as:

body {

font-family: Georgia;

line-height: 1.4;

font-weight: normal;

font-variant: small-caps;

font-size: 16px;

}

\*/

There are Six Font sub-properties. They are:

* [**font-style**](https://css-tricks.com/almanac/properties/f/font-style): It makes the text appear italicised or oblique.

|  |  |
| --- | --- |
| **Syntax:** | font-style: <value>; |

**Possible Values:**

* + - * normal
      * italic
      * oblique
      * inherit
* [**font-variant**](https://css-tricks.com/almanac/properties/f/font-variant): It changes target text to small caps.

|  |  |
| --- | --- |
| **Syntax:** | font-variant: <value>; |
| **Possible Values:** |  |

* + - * normal
      * small-caps
      * inherit
* [**font-weight**](https://css-tricks.com/almanac/properties/f/font-weight): It sets the weight or the thickness of the font.

|  |  |
| --- | --- |
| **Syntax:**  **Possible Values:** | font-weight: <value>; |

* + - * normal
      * bold
      * bolder
      * lighter
      * 100, 200, 300, 400, 500, 600, 700, 800, 900
      * inherit
* [**font-size**](https://css-tricks.com/almanac/properties/f/font-size)**:** It sets the height of the font.

|  |  |
| --- | --- |
| **Syntax:** | font-size: <value>; |

**Possible Values:**

* + - * xx-small
      * x-small
      * small
      * medium
      * large
      * x-large
      * xx-large
      * smaller, larger
      * percentage
      * inherit
* [**line-height**](https://css-tricks.com/almanac/properties/l/line-height)**:** It defines the amount of space above and below inline elements.

**Syntax:** line-height: <value>;

**Possible Values:**

* + - * normal
      * number (font-size multiplier)
      * percentage
* [**font-family**](https://css-tricks.com/almanac/properties/f/font-family)**:** It definies the font that is applied to the element.

**Syntax:** font-family: [[<family-name> | <generic-family>]]\* [<family-

name> | <generic-family>];

**<generic-family> names:**

* + - * sans-serif
      * serif
      * monospace
      * cursive
      * fantasy

**<font-family> names:**

* + - * caption
      * icon
      * menu
      * message-box
      * small-caption
      * status-bar
      * string

**Example:**

p.normal {

font-style: normal;

}

p.italic {

font-style: italic;

font-size: 40px;

font-weight: bold;

line-height: 90%;

font-family: Copperplate Gothic;

}

p.oblique {

font-style: oblique;

font-variant:small-caps;

}

</style>

</head>

<body>

<p class="normal">This is a paragraph in normal style.</p>

<p class="italic">This is a paragraph in italic style.</p>

<p class="oblique">This is a paragraph in oblique style.</p>

</body>

</html>

**CSS List Properties**

In CSS,there are two main types of lists:

* unordered lists (<ul>) - the list items are marked with bullets
* ordered lists (<ol>) - the list items are marked with numbers or letters

**The CSS list properties allow you to:**

* Set different list item markers for ordered lists
* Set different list item markers for unordered lists
* Set an image as the list item marker
* Add background colors to lists and list items

**1,2)Different List Item Markers for Ordered and Unordered**

**Lists:**

We can use  “**list-style-type”** property to specify the type of list item marker.

Example:

<!DOCTYPE html>

<html>

<head>

<style>

ul.a {

list-style-type: circle;

}

ul.b {

list-style-type: square;

}

ol.c {

list-style-type: upper-roman;

}

ol.d {

list-style-type: lower-alpha;

}

</style>

</head>

<body>

<p>Example of unordered lists:</p>

<ul class="a">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<ul class="b">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<p>Example of ordered lists:</p>

<ol class="c">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ol class="d">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

</body>

</html>

**3)** **Set an image as the list item marker:**

We can use the **“**list-style-image” property to specify an image as the list item marker.

Example:

<!DOCTYPE html>

<html>

<head>

<style>

ul {

list-style-image: url('Imagename');

}

</style>

</head>

<body>

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

</body>

</html>

## Position The List Item Markers:

We can use **“**list-style-position” property to specify whether the list-item markers should appear inside or outside the content flow.

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

ul.a {list-style-position:inside;}

ul.b {list-style-position:outside;}

</style>

</head>

<body>

<p>The following list has list-style-position: inside:</p>

<ul class="a">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<p>The following list has list-style-position: outside:</p>

<ul class="b">

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

<p>"list-style-position: outside" is the default setting.</p>

</body>

</html>

## 4)Styling List With Colors:

We can also style lists with colors, to make them look a little more interesting.

Anything added to the <ol> or <ul> tag, affects the entire list, while properties added to the <li> tag will affect the individual list items.

Example:

<!DOCTYPE html>

<html>

<head>

<style>

ol {

background: green;

padding: 20px;

}

ul {

background: yellow;

padding: 20px;

}

ol li {

background: red;

padding: 5px;

margin-left: 35px;

}

ul li {

background: pink;

margin: 5px;

}

</style>

</head>

<body>

<h1>Styling Lists With Colors:</h1>

<ol>

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ol>

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Coca Cola</li>

</ul>

</body>

</html>

**Color Properties in CSS:**

CSS color properties allows us to color the “**Background and Foreground Color”** on a Web Page. We can set CSS color on text, backgrounds, borders, and other parts of elements in a document.

## I.CSS Background Color

### (i)Set CSS body background color:

You can define the background color of a webpage by specifying its body **“background-color”** property.

**Example:**

body {background-color:#C0C0C0;}

### (ii)Set CSS Paragraph background color :

**Example:**

a. body

{

background-color:#C0C0C0;

}

b. p {background-color:#FFFFFF;}

### (iii)Set CSS div back color :

body {background-color:#C0C0C0;}

p {background-color:#FFFFFF;}

div {background-color:#00FFFF;}

## 2.CSS Foreground Color

### (i) Change CSS text color :

When we want to change the color of a text (foreground color) in an HTML

document the term color is used to specify the CSS property.

**Example:**

body { color: #800000 }

### (ii) Change the Font Color with CSS :

When you set foreground color , actually the font color will change.

**Example:**

h1 { color: green; }

The above code changes the font color inside the h1 tag.

### 3.Border Color

**(i)Set CSS border-color:**

The **“border-color”** property specifies the border color for each side of the box.

**Example:**

P{

border-width: 2px;

border-color:red;

border-style: solid;

or

border:2px solid red;

}

You can specify border color to each side specifically.

**Example:**

<html>

<head>

<style type="text/css">

p{

border-width: 8px;

border-style: solid;

border-top-color: red;

border-right-color: green;

border-bottom-color: purple;

border-left-color: blue;

}

</style>

</head>

<body>

<p>Border color define to each side</p>

</body>

</html>

## Color Keywords

The first and easiest way to specify a color is using one of the 17 predefined color **keywords** specified in CSS2.1.

|  |  |  |
| --- | --- | --- |
| **Color** | **Keyword** | **Hex Value** |
|  | Black | #000000 |
|  | Gray | #808080 |
|  | Silver | #c0c0c0 |
|  | White | #ffffff |
|  | Maroon | #800000 |
|  | Red | #ff0000 |
|  | Purple | #800080 |
|  | Fuchsia | #ff00ff |
|  | Green | #008000 |

|  |  |  |
| --- | --- | --- |
| **Color** | **Keyword** | **Hex Value** |
|  | Lime | #00ff00 |
|  | Olive | #808000 |
|  | Yellow | #ffff00 |
|  | Navy | #000080 |
|  | Blue | #0000ff |
|  | Teal | #008080 |
|  | Aqua | #0000ff |
|  | Orange | #ffa500 |

**Allignment of Text:**

* The **“**text-align” property in CSS is used for aligning the inner content of a Html element.

Example:

p {

text-align: center;

}

* These are the traditional values for text-align:
* left - The default value.Text aligns from the left side.
* right – Content or Text aligns from the right side.
* center - Content centers between the left and right edges. White space on

the left and right sides of each line should be equal.

* justify - Content spaces out such that as many blocks fit onto one line as

possible and the first word on that line is along the left edge and the

last word is along the right edge.

* inherit - The value will be whatever the parent element's is.

## Background Images

The **“**background-image” property specifies an image to use as the background of an element.

* By default, the image is repeated so it covers the entire element.

->The background image for a page can be set like this:

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-image: url("paper.gif");

}

</style>

</head>

<body>

<h1>Hello World!</h1>

<p>This page has an image as the background!</p>

</body>

</html>

# Div and Span Tags

**Div tags and span tags** are very common **HTML** elements that are growing in popularity due to their flexibility.

HTML div tags are more specific for organizational tasks like setting up the layout of your page, which are preferred over tables because of their fluid like nature. -

Span tags are used more to permit customization of text and are often used inside other HTML elements to customize a certain piece of content from the rest.

The internal layout of this page is created with div elements.

## HTML Div Element

## The div element is a block level element, much like the paragraph tag.

## Example

## <div> I am a div!</div> <div> Me too!</div>

## HTML Span Element

## The HTML span element is an inline element, which means that it can be used inside a block element and not create a new line.

## If we want to highlight some text inside a paragraph, wrap that text in a span tag.

## Span tags are often used to incorporate a specific CSS style to differentiate certain parts of content.

## Example

## <p> Something here is <span style="color:#900;"> special</span> , but which one?</p>

## Example:

<html>

## <head>

## <title>Span and Div Tags</title>

## <style>

## div

## {

## background-color:yellow;

## border:2px solid red;

## }

## p

## {

## color:green;

## }

## </style>

## </head>

## <body>

## <div>

## <h1> Hi,Good Morning</h2>

## <p> Welcome<span style="color:red;text-decoration:underline"> to</span> Paragraph</p>

## </div>

## </body>

## </html>

**Important Questions From UNIT-I**

1. a) Explain the request and response phase of HTTP (8m)

b) Explain how the levels of style sheets in CSS work (4m)

2. List out various selector forms in CSS and explain it with example(12m)

3. a) Define: IP address and Domain Name. Give suitable examples(4m)

b) Explain in detail about the operations of web server. How web browser

communicates to the web server?(8m)

4. a) Explain: “image tag” and “Hyperlink” in HTML(4m)

b) Describe Font families in CSS with examples(8m)

5. a) Explain the function of Web sever with suitable examples(6m)

b) Create a web page for entering customer orders using DHTML(6m)

6. a) What is a style sheet? Explain the basic features available in a style sheet(6m)

b) Distinguish between frames and forms with suitable examples(6m)

7. a) Give the relevant explanation of URL, HTTP, MIME(6m)

b)Generate an HTML document that would take student name,roll no.and address(6m) 8.a) Create a document using DHTML that aligns the image on the page, to left, centre

and to right using the links in the same document(6m)

b) What is the need for CSS? Explain its significance(6m)

9.a) Give the functionality and purpose of HTTP. List the notion of transactions and the

primary characteristics(6m)

b) Write an HTML document to provide a form that collects names and telephone

numbers(6 m)

10.a) Create a document using DHTML that changes the alignment of the heading to left,

right and centered using the links given in the same document(6m)

b) What are the advantages and disadvantages of using style sheets(6m)

11.Write a note about WWW. Explain caching technique used in WWW. Describe web

browser and web servers with example(12 m)

12.Write down the limitations of using entirely “pure” CSS only.(12m)

13.What is CSS ? Determine the advantages & disadvantages of CSS. Explain with

example what are the 4 ways to put CSS and HTML together (12m)

14. Explain how DHTML work. Write down the advantages & disadvantages of using

DHTML(12m).