

## **UNIT-I: HTML & CSS**

**HTML:** Basic Syntax, Standard HTML Document Structure, Basic Text Markup, Images, Hypertext Links, Lists, Tables, Forms, HTML5

**CSS:** Levels of Style Sheets, Style Specification Formats, Selector Forms, the Box Model, Conflict Resolution

---

HTML stands for Hypertext Markup Language. It is the most widely used language to create web pages. HTML is a markup language used to simply –mark-up a text document with tags that tell a web browser how to structure it to display.

HTML is defined with the use of the Standard Generalized Markup Language (SGML), which is an International Standards Organization (ISO) standard notation for describing text-formatting languages. The addition of style sheets to HTML in the late 1990s advanced its capabilities to specify presentation details of the content in HTML document.

### **BASIC SYNTAX**

- HTML is a descriptive language that helps convert ordinary text into hypertext by adding special code called tags or elements. The fundamental syntactic units of HTML are tags.
- Tags tell the web browser how to display the content in the HTML document.
- The syntax of a tag is the tag's name surrounded by angle brackets (< and >). Most tags appear in pairs: an opening tag and a closing tag. The name of a closing tag is the name of its corresponding opening tag with a slash attached to the beginning.
  - For example, if the tag's name is p, the corresponding closing tag is named /p, whatever appears between a tag and its closing tag is the content of the tag.
- A browser display of an HTML document shows the content of all of the document's tags. Not all tags can have content. The opening tag and its closing tag together specify a container for the content they enclose. The container and its content together are called an element.
  - For example, consider the following element:  
<p> this is simple stuff. </p>
- Attributes, which are used to describe the tag, can appear between an opening tag's name and its right angle bracket. They are specified in the form of name – value pairs; the attribute name is followed by an equal sign (=) and the attribute's value. Attribute values must be delimited by double quotes.
- Comments in programs increase the readability of those programs.
  - They can appear in the following form:  
<!-- Anything except two *adjacent dashes* -->
  - Browsers ignore comments—they are for people only. Comments can be spread over as many lines as are needed.
  - For example, you could have the following comment:

```
<!-- PetesHome.html
      This document describes the home document of
      Pete's Pickles -->
```

## **STANDARD HTML DOCUMENT STRUCTURE**

A HTML Document is mainly divided into two parts:

- **HEAD:** This contains the information about the HTML document. For Example, Title of the page, Meta Data, external link files etc.
- **BODY:** This contains everything you want to display on the Web Page.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>
```

Title of the document is written here

```
    </title>
```

```
</head>
```

```
<body>
```

```
    -----
```

```
    -----
```

```
</body>
```

```
</html>
```

- The <!DOCTYPE html> tells the document is a HTML Document and its version.
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the HTML page
- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.

And the HTML documents have the file extension name **.html** or **.htm**.

### **Example:**

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Page Title</title>
```

```
</head>
```

```
<body>
```

```
    <h1>My First Heading</h1>
```

```
    <p>My first paragraph.</p>
```

```
</body>
```

```
</html>
```

**HTML Head Section:**

The HTML <head> element is a container for the following elements: <title>, <style>, <meta>, <link>, <script> and <base>.

- The <title> element defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.
- The <style> element is used to define style information for a single HTML page
- The <link> element defines the relationship between the current document and an external resource. The <link> tag is most often used to link to external style sheets.
- The <meta> element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings.
- The <script> element is used to define client-side Java Scripts.

**HTML Body Section:**

The actual contents of the web page are placed in the body section. It include elements such as tables, paragraphs, lists, images, hyperlinks, headings, forms, etc. The <body> tag consists of the following attributes:

- **bbgcolor:** specifies the background of the web page
- **link:** specifies the color of the unvisited link color
- **alink:** specifies the color of active link
- **vlink:** specifies the color of the visited link
- **text:** specifies the text color
- **background:** specifies the URL of an image which is to be set as background of the body

## **BASIC TEXT MARKUP TAGS**

Markup means how the text content of a HTML document is being formatted with the use of HTML tags. Formatting describes layout and presentation details of the content in the document.

**PARAGRAPHS**

Text is normally organized into paragraphs in the body of the document. The <p> tag in HTML defines a paragraph. These have both opening and closing tag. So anything mentioned within <p> and </p> is treated as a paragraph.

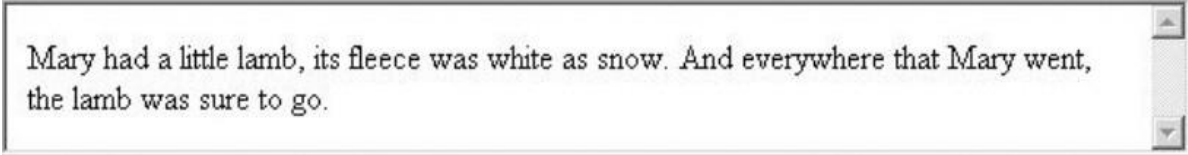
**Usage:**

<p> lines of content </p>

**Example:** Consider the below markup

```
<p>
  Mary had
a
  little lamb, its fleece was white as snow. And
  everywhere that
  Mary went, the lamb
  was sure to go.
</p>
```

Then the web page content is displayed as follows:



Mary had a little lamb, its fleece was white as snow. And everywhere that Mary went,  
the lamb was sure to go.

Any Line breaks embedded in text are ignored by the web browser.


## LINE BREAKS

Sometimes text requires a line break without the preceding blank line. For this purpose break tag `<br/>` is used. The break tag differs syntactically, as it have no content and therefore has no closing tag.

### Example:

```
<p>
Mary had a little lamb, <br />
  its fleece was white as snow.
</p>
```

Then the web page content is displayed as follows:



Mary had a little lamb,  
its fleece was white as snow.


## PRESERVING WHITE SPACE

Sometimes it is desirable to preserve the white space in text—that is, to prevent the browser from eliminating multiple spaces and ignoring embedded line breaks. This can be specified with the `<pre>` tag.

### Example:

```
<p><pre>
Mary
  had a
    little
      lamb
</pre>
</p>
```

Then the web page content is displayed as follows:



```
Mary
  had a
    little
      lamb
```

## HEADINGS

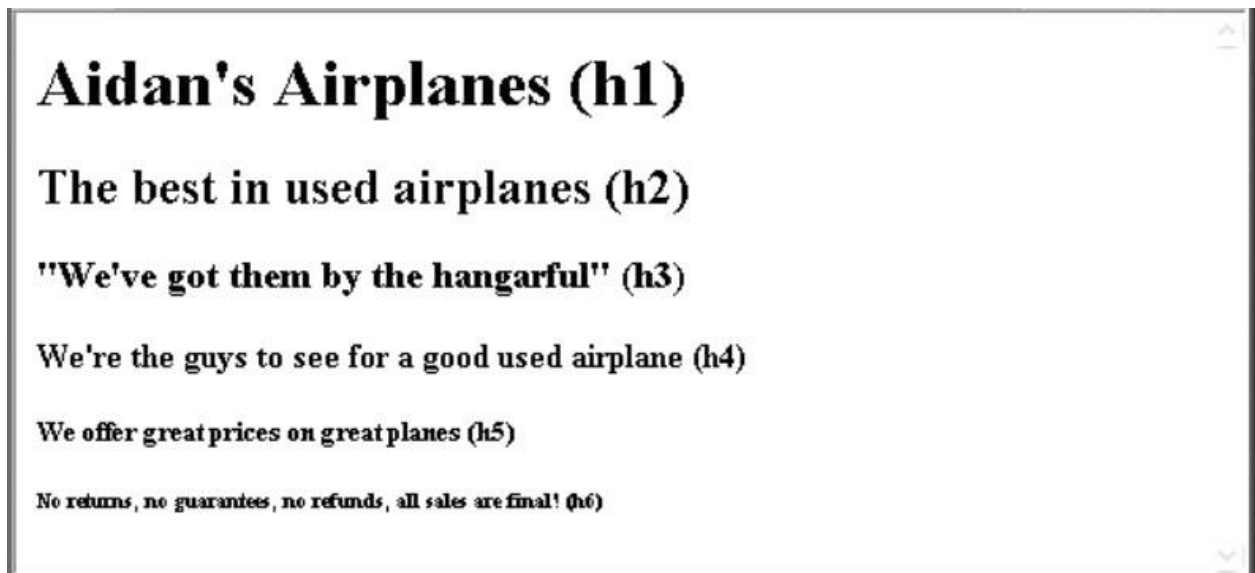
Text is often separated into sections in documents by beginning each section with a heading. Larger sections sometimes have headings that appear more prominent than headings for sections nested inside them. In HTML, there are six levels of headings, specified by the tags `<h1>`, `<h2>`,

<h3>, <h4>, <h5>, and <h6>, where <h1> specifies the highest-level heading, <h2> is below to the level of <h1>, <h3> is below to the level of <h2> and so on, <h6> is the lowest level of heading.

**Example:**

```
<html>
<head> <title> Headings </title>
</head>
<body>
<h1> Aidan's Airplanes (h1) </h1>
<h2> The best in used airplanes (h2) </h2>
<h3> "We've got them by the hangarful" (h3) </h3>
<h4> We're the guys to see for a good used airplane (h4) </h4>
<h5> We offer great prices on great planes (h5) </h5>
<h6> No returns, no guarantees, no refunds, all sales are final! (h6) </h6>
</body>
</html>
```

Below figure shows how the browser display the headings



## **TEXT FORMATTING**

**Formatting** is a process of formatting text for better look and feel. HTML provides us ability to format text without using CSS. There are many formatting tags in HTML. These tags are used to make text bold, italicized, underlined, superscript, subscript, etc.

### **BOLD TEXT**

HTML<b> and <strong> formatting elements to bold text

- The HTML <b> element is a physical tag which displays text in bold font, without any logical importance. If you write anything within <b>.....</b> element, is shown in bold letters.

Example:

```
<p> <b>Write Your First Paragraph in bold text.</b></p>
```

Output:

**Write Your First Paragraph in bold text.**

- The HTML `<strong>` tag is a logical tag, which displays the content in bold font and informs the browser about its logical importance. If you write anything between `<strong>??????</strong>`, is shown important text.

Example:

```
<p><strong>This is an important content</strong>, and this is normal content</p>
```

Output:

**This is an important content, and this is normal content**

## ITALIC TEXT

HTML `<i>` and `<em>` formatting elements to apply italic style

- The HTML `<i>` element is physical element, which display the text content in italic font, without any added importance. If you write anything within `<i>.....</i>` element, is shown in italic letters.

Example:

```
<p> <i>Write Your First Paragraph in italic text.</i></p>
```

Output:

*Write Your First Paragraph in italic text*

- The HTML `<em>` tag is a logical element, which will display the textual content in italic font, with added semantics importance.

Example:

```
<p><em>This is an important content</em>, which displayed in italic font.</p>
```

Output:

*This is an important content, which displayed in italic font.*

## UNDERLINED TEXT

If you write anything within `<u>.....</u>` element, is shown in underlined text.

Example:

```
<p> <u>Write Your First Paragraph in underlined text.</u></p>
```

Output:

Write Your First Paragraph in underlined text.

## STRIKE TEXT

Anything written within `<strike>.....</strike>` element is displayed with strikethrough. It is a thin line which crosses the statement.

Example:

```
<p> <strike>Write Your First Paragraph with strikethrough</strike>.</p>
```

Output:

~~Write Your First Paragraph with strikethrough.~~

### MONOSPACED FONT

If you want that each letter has the same width then you should write the content within `<tt>.....</tt>` element.

Example:

```
<p>Hello <tt>Write Your First Paragraph in monospace font.</tt></p>
```

Output:

Hello Write Your First Paragraph in monospace font.

### SUPERSCRIP TEXT

If you put the content within `<sup>.....</sup>` element, is shown in superscript; means it is displayed half a character's height above the other characters.

Example:

```
<p>Hello <sup>Write Your First Paragraph in superscript.</sup></p>
```

Output:

Hello <sup>Write Your First Paragraph in superscript.</sup>

### SUBSCRIPT TEXT

If you put the content within `<sub>.....</sub>` element, is shown in subscript; means it is displayed half a character's height below the other characters.

Example:

```
<p>Hello <sub>Write Your First Paragraph in subscript.</sub></p>
```

Output:

Hello <sub>Write Your First Paragraph in subscript.</sub>

### HORIZONTAL RULES

The `<hr>` tag in HTML stands for horizontal rule and is used to insert a horizontal rule or a thematic break in an HTML page to divide or separate document sections. The `<hr>` tag is an empty tag and it does not require an end tag.

**Attributes:**

- **Align** is used to specify the alignment of the horizontal rule
- **Size** is used to specify the thickness (in terms of pixels) of the horizontal rule
- **Width** is used to specify the width of the horizontal ruler

**Example:**

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML hr tag</title>
  </head>
  <body>
    <p>There is a horizontal rule below this paragraph.</p>
    <hr>
    <p>This is a horizontal rule above this paragraph.</p>
  </body>
</html>
```

**Outcome:**

There is a horizontal rule below this paragraph.

---

This is a horizontal rule above this paragraph.

**FONT STYLE AND SIZE**

HTML's **<font>** tag can be used to add style, size, and color to the text of web pages. HTML's **<basefont>** tag is used to set all of your text to the same size, face, and color.

This tag can be used in HTML 4 and it cannot be used new versions of HTML.

**Attributes:**

- **Size** is the attribute used to specify the size of the font with a range of 1(smallest) to 7 (largest). Default size is 3. Size of the font can also be specified relative to the current size.
- **Face** is the attribute used to specify the font face
- **Color** is the attribute used to specify the font color

**Example:**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Font Size, Face and Color</title>
  </head>
  <body>
    <font face = "Times New Roman" size = "5" color="red">Times New Roman</font><br />
    <font face = "Verdana" size = "+2" color="#006633">Verdana</font><br />
    <font face = "Comic sans MS" size = " 5" color="#654789">Comic Sans MS</font><br />
  </body>
</html>
```

**Outcome:**



Times New Roman  
 Verdana  
 Comic Sans MS

### <BASEFONT> ELEMENT

The <basefont> element is supposed to set a default font size, color, and typeface for any parts of the document. One can use the <font> tag to override the <basefont> settings. The <basefont> tag also takes color, size and face attributes

### CHARACTER ENTITIES

HTML provides a collection of special characters that are sometimes needed in a document but cannot be typed as themselves – for example, >, <, and &. These special characters are defined as *entities*, which are codes for the characters. An entity in a document is replaced by its associated character by the browser.

Following table lists these special characters:

Character	Entity	Meaning
&	&amp;	Ampersand
<	&lt;	Is less than
>	&gt;	Is greater than
"	&quot;	Double quote
'	&apos;	Single quote (apostrophe)
$\frac{1}{4}$	&frac14;	One-quarter
$\frac{1}{2}$	&frac12;	One-half
$\frac{3}{4}$	&frac34;	Three-quarters
°	&deg;	Degree
(space)	&nbsp;	Nonbreaking space

## HTML Elements

An HTML element is defined by a start tag, some content, and an end tag.

---

<tagname>Content goes here...</tagname>

Examples of some HTML elements:

<h1>My First Heading</h1>

<p>My first paragraph.</p>

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
<p>	My first paragraph.	</p>
 	None	none

**Note:** Some HTML elements have no content (like the <br> element). These elements are called empty elements. Empty elements do not have an end tag!

### Nested HTML Elements

HTML elements can be nested (this means that elements can contain other elements).

All HTML documents consist of nested HTML elements.

The following example contains four HTML elements (<html>, <body>, <h1> and <p>):

#### Example

```
<!DOCTYPE html>
<html>
<body>

<p>My first <h1>paragraph</h1>.</p>

</body>
</html>
```

#### Empty HTML Elements

HTML elements with no content are called empty elements.

The <br> tag defines a line break, and is an empty element without a closing tag:

## HTML Attributes

- All HTML elements can have **attributes**
- Attributes provide **additional information** about elements
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

### The href Attribute

The <a> tag defines a hyperlink. The href attribute specifies the URL of the page the link goes to:

#### Example

```
<a href="https://www.w3schools.com">Visit W3Schools</a>
```

### The src Attribute

The `<img>` tag is used to embed an image in an HTML page. The `src` attribute specifies the path to the image to be displayed:

#### Example

```

```

#### The width and height Attributes

The `<img>` tag should also contain the `width` and `height` attributes, which specifies the width and height of the image (in pixels):

#### Example

```

```

#### The alt Attribute

The required `alt` attribute for the `<img>` tag specifies an alternate text for an image, if the image for some reason cannot be displayed. This can be due to slow connection, or an error in the `src` attribute, or if the user uses a screen reader.

#### Example

```

```

See what happens if we try to display an image that does not exist:

```

```

## HTML Headings

HTML headings are titles or subtitles that you want to display on a webpage.

HTML headings are defined with the `<h1>` to `<h6>` tags.

`<h1>` defines the most important heading. `<h6>` defines the least important heading.

```
<!DOCTYPE html>
<html>
<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
</body>
</html>
```

#### Output:

**Heading 1****Heading 2****Heading 3****Heading 4****Heading 5****Heading 6**

## **IMAGES**

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

The HTML `<img>` tag is used to embed an image in a web page. The `<img>` tag has two required attributes:

- `src` - Specifies the path (URL) to the image
- `alt` - Specifies an alternate text for the image

Images are not inserted into a web page; images are just linked to web pages. When a web page loads; then the browser gets the image from the location specified in `src` attribute and inserts it into the page. If the browser cannot find the image, then it will display the text message specified to the `alt` attribute.

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

**Usage:**

```

```

**Example:**

```

```

- Another two attributes – `height` and `width` used to specify the height and width of the image to be displayed. The height and width attribute values are specified in terms of pixels.

**Example:**

```

```

Some more attributes are:

- **`border`** - specifies the border thickness of image in terms of pixels
- **`hspace`** – specified the horizontal margin of the image
- **`vspace`** – specifies the vertical margin of the image
- **`align`** – sets the alignment of the image, possible values are left, right and center

### **IMAGE AS LINK**

To use an image as a link, put the `<img>` tag inside the anchor tag, `<a>`

**Example**

```
<a href="default.asp">
```

```

</a>
```

**ABSOLUTE PATH AND RELATIVE PATH:**

To insert an image into web page, the path of the image file is given in **src** attribute.

Path of the file can be specified in two ways:

1. **Absolute Path** *Absolute path* is the full address of the image file to access in the Internet.

Example:

```

```

2. **Relative Path** *Relative path* is the address of the image file in relation to the current web page loaded in web browser.

- Below is the example shows the path of the file present in the same folder of the current web page file.

```

```

- Below is the example shows the path of the file present in a folder above the folder of the current web page file.

```

```

## HTML LISTS

HTML lists allow web developers to group a set of related items in lists. HTML offers three ways for specifying lists of information. All lists must contain one or more list elements.

Lists are of three types:

1. Unordered list
2. Ordered List
3. Definition list

**HTML UNORDERED LIST:** 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>
```

**Outcome**

- Beetroot
- Ginger
- Potato
- Radish

**TYPE Attribute**

The **type** attribute is used 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">
```

**HTML ORDERED LISTS**

HTML ordered list lists items as a numbered list instead of bulleted. 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>
```



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

**TYPE Attribute**

The **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.

**START Attribute**

The **start** attribute for <ol> tag is used 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.

**HTML DESCRIPTION LIST OR DEFINITION LIST**

Description list is also a list style which is supported by HTML. It is also known as definition list where entries are listed like a dictionary or encyclopedia. The definition list is very appropriate when you want to present glossary, list of terms or other name-value list.

The HTML definition list contains following three tags:

1. **<dl> tag** defines the start of the definition list
2. **<dt> tag** defines an individual term.
3. **<dd> tag** defines the term definition (description).

**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>
```

**Outcome:****HTML**

This stands for Hyper Text Markup Language

**HTTP**

This stands for Hyper Text Transfer Protocol

## **HTML TABLES**

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

- In HTML a table is created 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.
- Table heading can be defined using <th> element. Normally you will put your top row as table heading; <th> element is used for this purpose instead of <td>.

**Example:**

```
<table border="1">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td>Raghu</td>
<td>15000</td>
</tr>
<tr>
<td>Raju</td>
<td>17000</td>
</tr>
</table>
```

This will produce the following result:

Name	Salary
Raghu	15000
Raju	17000

**Attributes**

The HTML <table> tag has the following additional attributes –



Attribute	Value	Description
<b>Align</b>	right left center justify	Visual alignment.
<b>Bgcolor</b>	rgb(x,x,x) #hexcode colorname	Specifies the background color of the table.
<b>Border</b>	Pixels	Specifies the border width. Value of "0" means no border.
<b>cellpadding</b>	pixels or %	Specifies the space between the cell borders and their contents.
<b>cellspacing</b>	pixels or %	Specifies the space between cells.
<b>Width</b>	pixels or %	Specifies the width of the table.
<b>background</b>	File path	Specifies the path of the image file to set the image as background of the table
<b>Colspan</b>	No of columns	To merge cells in two or more columns into a single cell
<b>Rowspan</b>	No of rows	To merge cells in two or more rows into a single cell

**Example:**

```

<html>
<body>
<table border="5" align="center" bgcolor="yellow" cellpadding=20px cellspacing=10px>
<thead="student data">
<tr>
<th>Name</th>
<th>Salary</th>
</tr>
<tr>
<td colspan=3>Raghu</td>
<td>15000</td>
</tr>
<tr>
<td>Raju</td>
<td>17000</td>
</tr>
</table>
</body>
</html>

```

**Output:**

Name	Salary	
Raghu	15000	
Raju	17000	

**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.
- ❖ The **<thead>** element is used in conjunction with the **<tbody>** and **<tfoot>** elements to specify each part of a table (header, body, footer).
- ❖ Browsers can use these elements to enable scrolling of the table body independently of the header and footer. Also, when printing a large table that spans multiple pages, these elements can enable the table header and footer to be printed at the top and bottom of each page.

```
<table>
  <thead>
    <tr>
      <th>Month</th>
      <th>Savings</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>January</td>
      <td>$100</td>
    </tr>
    <tr>
      <td>February</td>
      <td>$80</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>Sum</td>
      <td>$180</td>
    </tr>
  </tfoot>
</table>
```

## HTML Multimedia

Multimedia comes in many different formats. It can be almost anything you can hear or see, like images, music, sound, videos, records, films, animations, and more.

Web pages often contain multimedia elements of different types and formats.

### **Browser Support**

The first web browsers had support for text only, limited to a single font in a single color.

Later came browsers with support for colors, fonts, images, and multimedia!

### **Multimedia Formats**

Multimedia elements (like audio or video) are stored in media files.

The most common way to discover the type of a file, is to look at the file extension.

Multimedia files have formats and different extensions like: .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

### Common Video Formats

Only MP4, WebM, and Ogg video are supported by the HTML standard.

#### Example:

```
<!DOCTYPE html>
<html>
<body>
<video width="400" controls>
  <source src="D:\D Drive\z60 Photos\WhatsApp Video\VID-20180515-WA0001.mp4"
  type="video/mp4">
```

Your browser does not support HTML video.

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

### Common Audio Formats

MP3 is the best format for compressed recorded music. The term MP3 has become synonymous with digital music.

If your website is about recorded music, MP3 is the choice.

**Note:** Only MP3, WAV, and Ogg audio are supported by the HTML standard.

```
<!DOCTYPE html>
<html>
<body>

<audio controls>
  <source src="D:\D Drive\ILAYARAJA Vol.2\016manchukuresa.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

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

### HTML Plug-ins

Plug-ins are computer programs that extend the standard functionality of the browser.

the <object> Element

The `<object>` element is supported by all browsers.

The `<object>` element defines an embedded object within an HTML document.

It was designed to embed plug-ins (like Java applets, PDF readers, and Flash Players) in web pages, but can also be used to include HTML in HTML:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<object width="100%" height="500px" data="C:\Users\Public\Pictures\Sample  
Pictures\Chrysanthemum.jpg"></object>
```

```
</body>
```

```
</html>
```

## **HTML HYPERLINKS**

HTML links are hyperlinks. You can click on a link and jump to another document. When you move the mouse over a link, the mouse arrow will turn into a little hand.

Web pages can contain hyperlinks that take you directly to other pages and even specific parts of a given page. Hyperlinks allow visitors to navigate between Web sites or between the web pages of one single web site by clicking on words, phrases, and images. Hyperlinks can be created using text or images.

### **Linking Documents - The `<a>` Element:**

A link is specified using the `<a>` element. This element is called anchor tag as well. Anything between the opening `<a>` tag and the closing `</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 this tag:

```
<a href="Document URL" /> Link Phrase </a>
```

The **href** attribute is used to define the address of the document to be linked.

- Links can include images in their content, in which case the browser displays the image together with the link:

```
<a href = "c210data.html" >
```

```
<img src = "small-airplane.jpg" alt = "An image of a small airplane" />
```

```
</a>
```

An image itself can be an effective link (the content of the anchor element). The content of an anchor element for such a link is just the image element.

### **Example:**

*[first.html](#)*

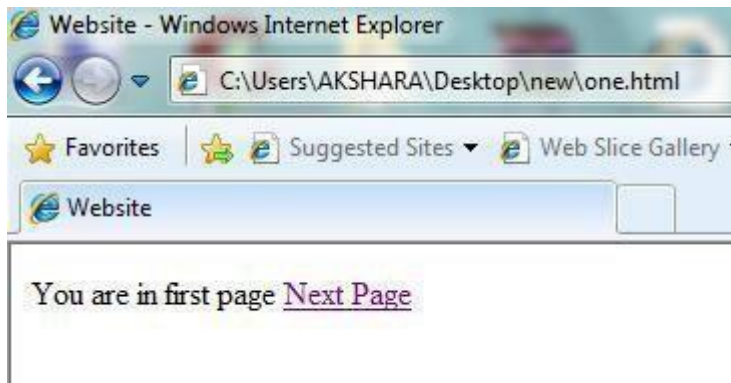
```
<html>
```

```
<head><title> Navigation </title></head>
```

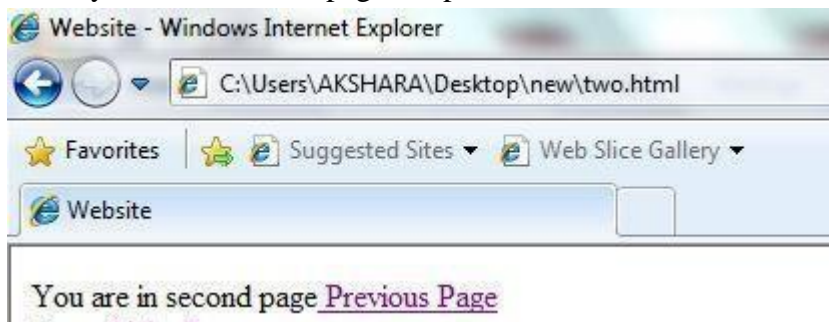
```
<body>
<p align="left">You are in first page<a href="second.html">Next Page</a></p>
</body>
</html>
```

### *second.html*

```
<html>
<head><title> Navigation </title></head>
<body>
<p align="left">You are in second page<a href="first.html">Previous Page</a></p>
</body>
</html>
```



When you click on Next page, it opens



## Targets within Documents

If the target of a link is not at the beginning of a document, it must be some element within the document, in which case there must be some means of specifying it. The target element can include an id attribute, which can then be used to identify it in **href** attribute.

- Consider the following example:

```
<h2 id = "avionics"> Avionics </h2>
```

The value of an id attribute must be unique within the document.

- If the target is in the same document as the link, the target is specified in the **href** attribute value by preceding the id value with a pound sign (#), as in the following example:

`<a href = "#avionics"> What about avionics? </a>`

- When the target is a part or fragment of another document, the name of the part is specified at the end of the URL, separated by a pound sign (#), as in this example:

`<a href = "AIDAN1.html#avionics"> Avionics </a>`

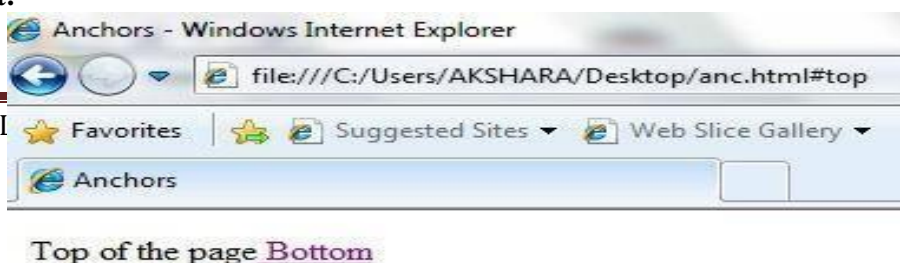
**<!—write an example to illustrate the usage of anchor tag - - >**

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

- href:** specifies the URL of the target of a hyperlink. Its value is any valid document URL, absolute or relative, including a fragment identifier or a JavaScript code fragment.
- target:** specify where to display the contents of a selected hyperlink.
  - If set to "\_blank" then a new window will be opened to display the loaded page.
  - If set to "\_top" or "\_parent" then same window will be used to display the loaded document
  - If set to "\_self" then loads the new page in current window.
  - By default it is "\_self".
- name & id:** attributes places a label within a document. When that label is used in a link to that document, it is the equivalent of telling the browser to goto that label.
- event:** attributes like onClick, onMouseOver etc. are used to trigger any Java script code.
- title:** attribute lets you specify a title for the document to which you are linking. The value of the attribute is any string, enclosed in quotation marks. The browser might use it when displaying the link, perhaps flashing the title when the mouse passes over the link.
- accesskey:** This attribute provides a keyboard shortcut that can be used to activate a link. For example, you could make the T key an access key so that when the user presses either the Alt or Ctrl key on his keyboard (depending on his operating system) along with the T key, the link gets activated.

```
<html>
  <head><title> Anchors </title></head>
  <body>
    <p align="left">Top of the page<a name="top" href="#bottom">Bottom</a></p>
    <br/><br/><br/><br/><br/><br/><br/><br/><br/>
    <p align="left">Bottom of the page<aname="bottom" href="#top"> Back to Top
  </a></p>
</body>
</html>
```

**Output:**



- ☐ When you click on bottom

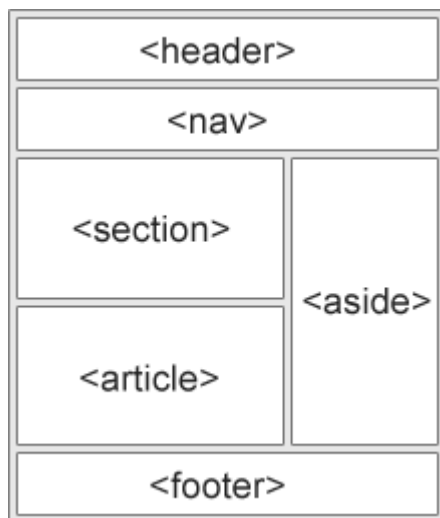


## HTML Layout

Websites often display content in multiple columns (like a magazine or a newspaper).

### HTML Layout Elements

HTML has several semantic elements that define the different parts of a web page:



- **<header>** - Defines a header for a document or a section
- **<nav>** - Defines a set of navigation links
- **<section>** - Defines a section in a document
- **<article>** - Defines an independent, self-contained content
- **<aside>** - Defines content aside from the content (like a sidebar)
- **<footer>** - Defines a footer for a document or a section
- **<details>** - Defines additional details that the user can open and close on demand
- **<summary>** - Defines a heading for the **<details>** element

### Example:

```
<html>
```

```
<body>
```

```
<header>
```

```
<h1>City Gallery</h1>
```

```
</header>
```

```
<nav>
```

```
London<br>
```

```
Paris<br>
```

```
Tokyo
```

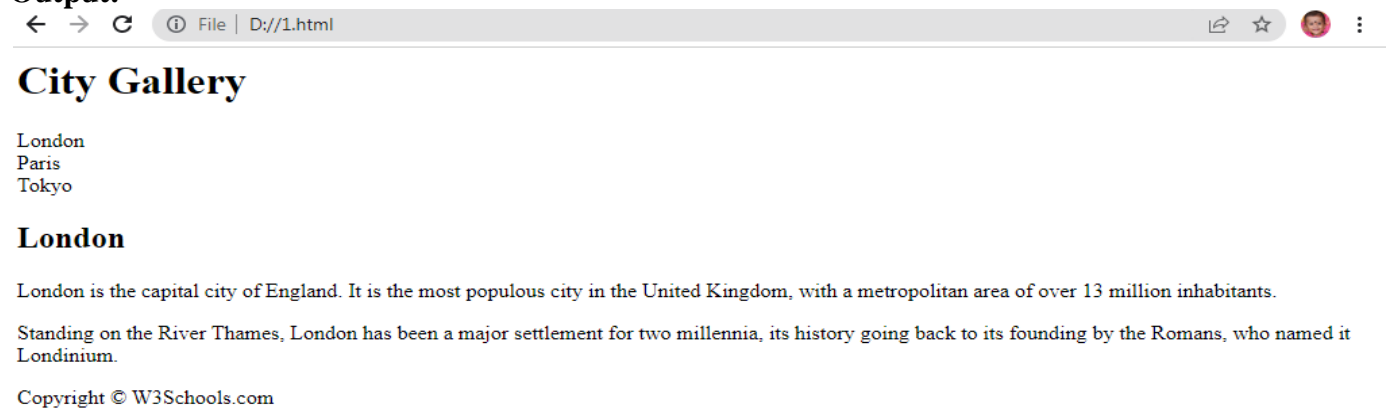
```
</nav>
```

```
<section>
```

```
<h1>London</h1>
<p>London is the capital city of England. It is the most populous city in the United Kingdom,
with a metropolitan area of over 13 million inhabitants.</p>
<p>Standing on the River Thames, London has been a major settlement for two millennia,
its history going back to its founding by the Romans, who named it Londinium.</p>
</section>

<footer>
Copyright © W3Schools.com
</footer>

</body>
</html>
```

**Output:**

## Frames

Frames provide an interface which makes your web site easy to navigate. Frames divide the browser window into horizontal or vertical sections. Each section can be used to display a separate HTML page.

Each frame within a webpage has the following properties:

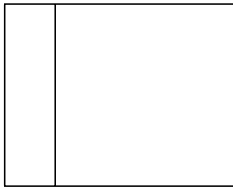
- ❖ It has unique name
- ❖ It displays an HTML document independent of other frames
- ❖ Its size can be dynamically changed according to the size of the content in HTML page

**Creating frames:**

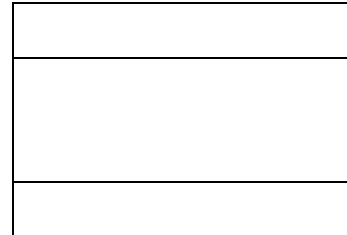
When we talk about frames actually we are referring to frameset, which is a special type of web page. Simply frameset is nothing but collection of frames. Web page that contains frame element is called framed page. Framed page begins with `<frameset>` tag and ends with `</frameset>`. Each individual frame is identified through `<frame>` tag. Creation of framed page is very simple. You can nest the framesets. First you decide how you want to divide your webpage and accordingly define frame elements.



Consider the following diagrams, first page divides into two columns and the second form divides into three rows.



***Two columns frameset***



***Three rows frameset***

### **The <FRAMESET> . . . </FRAMESET> Tags**

The <FRAMESET> . . . </FRAMESET> tags are the container tags for all other tags that are used to create frames. These tags are used to specify the size and number of the frames to be created in an HTML Page. The frames within these tags are jointly referred to as *aframeset*.

Syntax:

```
<FRAMESET cols="n,n" rows="n,n">
  <FRAME> Tags
</FRAMESET>
```

### **Attributes**

**COLS:** The COLS attribute is used to specify the number of vertical frames within HTML Page.

**ROWS:** The ROWS attribute is used to specify the number of horizontal frames within HTML Page. To specify the values of the attributes, use a comma-separated list of values. The values can be of the following two types:

- Value: This numeric value is used to specify number of pixels that a frame will occupy in the browser window.
- Value%: This value is used to specify percentage of space that each frame will occupy in the browser window. The total of all the values specified in the comma-separated list is 100%.

In order to divide into two columns we can use the following syntax

```
<FRAMESET COLS="25%,75%">
  <frame> tags
</FRAMESET>
```

In the second diagram we have three rows so by using rows parameter of frameset, we can divide logically the window into three rows.

```
<FRAMESET ROWS="20%,70%,10%">
  <frame> tags
</FRAMESET>
```

According to above code, first row occupies 20% of the window, third row occupies 10% of the window, second row occupies 70% of the window.

### **<FRAME> Tag:**

You use the <FRAME> Tag to create frame with in *frameset*. You can use the following attributes

to specify the name of the frame and the HTML page that you need to display in the frame.

- ❖ **SRC**: SRC attribute to specify the HTML page that you want to display in a frame.
- ❖ **NAME**: NAME attribute to specify the name of the frame. This *NAME* is used by the anchor element to display the linked HTML Pages with in frame.
- ❖ **SCROLLING**: attribute used to add scrollbars to a frame. The SCROLLING attribute takes three value: YES, NO, AUTO.
  - The value *YES* specifies that a scrollbar should always be visible on the frame
  - The value *NO* specifies that the scrollbar should never be visible on the frame
  - The value *AUTO* specifies the browser to automatically display or remove scrollbars from a frame
- ❖ **FRAMEBORDER**: attribute to specify whether a frame should have a border. The value 0(zero) specifies no border. If you specify any other numeric value, the border is displayed.
- ❖ **NORESIZE**: By default, You can resize a frame by dragging its borders. You can restrict this by specifying the NORESIZE attribute.

**Syntax:**

```
<FRAME SRC = "URL" NAME = " myframe" SCROLLING = "yes | no |  
auto"FRAMEBORDER = "0|1" [NORESIZE]/>
```

**Nested Framesets:**

Some times it is required to divide your window into rows and columns, and then there is requirement of nested framesets. *Frameset with in another frameset is known as nested frameset.*

**Example:**

**home.html**

```
<frameset rows="20%,*">  
  <frame name="fr1" src="top.html">  
  <frameset cols="25%,*">  
    <frame name="fr2"src="dept.html">  
    <frame name="fr3"src="desc.html">  
  </frameset>  
</frameset>
```

**top.html**

```
<html>  
  <body text="magenta">  
    <center>  
      <br/><h1> VLITS::VADLAMUDI</h1>  
    </center>  
  </body></html>
```

**dept.html**

```
<html>  
  <body text="red">
```

```
<center>
<h1>
  <br/>CSE<br/>
  <br/>EEE<br/>
  <br/>ECE<br/>
</h1>
</body></html>
```

**desc.html**

```
<html>
  <body text="maroon">
    <center>
      <br/><br/>
      <h1>Description of College</h1>
    </center>
  </body></html>
```

## **HTML FORMS**

The most common way for a user to communicate information from a Web browser to the server is through a form. HTML provides tags to generate the commonly used objects on a screen form to collect data from user, just like a paper form. These objects are called controls or widgets.

There are controls for single-line and multiple-line text collection, checkboxes, radio buttons, and menus, among others. All control tags are inline tags. Most controls are used to gather information from the user in the form of either text or button selections. Each control can have a value, usually given through user input. Every form requires a Submit button. When the user clicks the Submit button, the form data is encoded and sent to the Web server for processing.

### **The <form> Tag**

The <form> tag is used to create a form in a webpage. All of the controls of a form appear as the content of a <form> tag.

- The <form> tag has an attribute – **action**, used to specify the URL of the application on the Web server that is to be called when the user clicks the *Submit* button.
- The <form> tag has another attribute – **method**, used to pass the form data to the server.

There are two techniques for passing the data to the server: **get** and **post**

- **GET is used to request data from a specified resource.**

When get method is used, the browser attaches the query string to the URL of the HTTP request, so the form data is transmitted to the server together with the URL.

- **POST is used to send data to a server to create/update a resource.**

When the post method is used, the query string is passed by some other method to the form - processing program. There is no length limitation for the query string with the post method, so it is the better choice. The data sent to the server with POST is stored in the request body of the HTTP request

### The <input> TAG

Many of the commonly used controls are specified with the inline tag <input> for text-box, passwords, checkboxes, radio buttons, and the action buttons *Reset*, *Submit*, and *plain*.

The one attribute of <input> that is required for all of the controls is **type**, which specifies the particular kind of control.

Type	Description
Button	A push button with no default behavior displaying the value of the value attribute, empty by default.
Checkbox	A check box allowing single values to be selected or deselected.
File	A control that lets the user select a file. Use the <b>accept</b> attribute to define the types of files that the control can select.
Hidden	A control that is not displayed but whose value is submitted to the server.
Password	A single-line text field whose value is obscured.
Radio	A radio button, allowing a single value to be selected out of multiple choices with the same name value.
Reset	A button that resets the contents of the form to default values. Not recommended.
Submit	A button that submits the form data to the web server.
Text	A single-line text field used to input textual content from the user

### TEXT CONTROL:

A text control, which we usually refer to as a text box, creates a horizontal box into which the user can type text. Text boxes are often used to gather information from the user, such as the user's name and address.

#### Attributes

**Type** attribute takes the value – **text**, to create a text box

**Name** attribute takes any valid identifier name so that it identifies uniquely while processing the form data either at the client side or server side.

**Value** attribute takes the value that is entered in the text box by the user

**Size** attribute specifies the length of the text box in terms of number of characters

**Maxlength** attribute specifies the maximum number of characters accepted by the browser

**Example:** <form action = "">

```
<p>
  <input type = "text" name = "Name" size = "25" maxlength = "25" />
</p>
</form>
```

### PASSWORD CONTROL:

If the contents of a text box should not be displayed when they are entered by the user, a password control can be used as follows:

**<input type = "password" name = "myPassword" size = "10" maxlength = "10" />**

In this case, regardless of what characters are typed into the password control, only bullets or asterisks are displayed by the browser

### RADIO BUTTONS

<Input> elements of type **radio** are generally used in radio groups, which is a collection of radio buttons describing a set of related options. Only one radio button in a given group can be selected at the same time. Radio buttons are typically rendered as small circles, which are filled or highlighted when selected.

#### Attributes

- **value** attribute contains the radio button's value, which is hidden to the user
- A radio group is defined by giving each of radio buttons in the group the same name – using the attribute **name**. Once a radio group is established, selecting any radio button in that group automatically deselects any currently-selected radio button in the same group.
- **Checked** is a boolean attribute indicating whether or not this radio button is the default-selected item in the group

Example:

```
<head>
  <title> Radio </title>
</head>
<body>
<p> Age Category </p>
<form action = "">
<p>
<label><input type = "radio" name = "age" value = "under20" checked = "checked" /> 0-19 </label>
<label><input type = "radio" name = "age" value = "20-35" /> 20-35 </label>
<label><input type = "radio" name = "age" value = "36-50" /> 36-50 </label>
<label><input type = "radio" name = "age" value = "over50" /> Over 50 </label>
</p>
</form>
</body>
```

&lt;/html&gt;

**Outcome:**

Age Category

☒ 0-19 ☐ 20-35 ☐ 36-50 ☐ Over 50

**CHECK BOXES**

<Input> elements of type checkbox are rendered by default as boxes that are checked (ticked) when activated. Generally this is appeared like a square but it may have rounded corners. A checkbox allows you to select single value or multiple values.

**Attributes:*****Name***

This attribute takes any valid identifier name as its value so that it is identified uniquely while the form is processing on the client side or server side.

***Value***

It is the value of the checkbox when submitting the form, if the checkbox is checked.

**Checked**

A Boolean attribute that indicates checkbox is checked by default, when the page is loaded.

**Indeterminate**

A Boolean attribute indicates that the value of the checkbox is indeterminate rather than true or false

**Example:**

```
<html>
<head> <title> Checkboxes </title></head>
<body>
<p>Grocery Checklist</p>
<form action = "">
<p>
<label> <input type = "checkbox" name = "groceries" value = "milk" checked = "checked" /> Milk
</label>
<label> <input type = "checkbox" name = "groceries" value = "bread" /> Bread </label>
<label> <input type = "checkbox" name = "groceries" value = "eggs" /> Eggs </label>
</p>
</form>
</body>
</html>
```

**Outcome:**A screenshot of a web browser window displaying a form titled "Grocery Checklist". Below the title, there are three checkboxes: "Milk" (which is checked, indicated by a small square with a checkmark), "Bread" (unchecked), and "Eggs" (unchecked). The browser window has a standard address bar and scrollbar.**The <select> TAG**

<select> tag is used to create a menu of items which are large in number so that the user can select the desired one among the available items. It allows the user to select either one item or multiple items.

- The attribute **multiple** whose value set to **multiple** can allows the user to select multiple options among the choice; otherwise only one single option is allowed to select.
- The **size** attribute, specifying the number of menu items that are to be displayed for the user, can be included in the <select> tag.
  - If no size attribute is specified, the value 1 is used. Then just one menu item is displayed, with a downward scroll arrow. If the scroll arrow is clicked, the menu is displayed as a pop-up menu.
  - If the size attribute is set to a number larger than 1, the menu is usually displayed as a scrolled list.
- Each of the items in a menu is specified with an **<option>** tag, nested in the select element. The content of an <option> tag is the value of the menu item. The <option> tag can include the **selected** attribute, which specifies that the item is pre-selected.

- **Example:**

```
<html>
<head> <title> Menu </title></head>
<body>
<p> Grocery Menu - milk, bread, eggs, cheese</p>
<form action = "">
<p>
With size =1 (default)
<select name = "groceries">
<option> milk </option>
<option> bread </option>
<option> eggs </option>
<option> cheese </option>
</select>
</p>
</form>
</body>
</html>
```

**Outcome:**

**THE <textarea> TAG**

In some situations, a multiline text area is needed. The <textarea> tag is used to create such a control. The text typed here is not limited in length, and there is implicit scrolling when needed, both vertically and horizontally. To specify the size of the text area, **rows** and **cols** attributes should be used

**Example:**

```
<body>
<p>
Please provide your employment aspirations
</p>
<form action = "handler">
<p>
<textarea name = "aspirations" rows = "3" cols = "40">
(Be brief and concise)
</textarea>
</p>
</form>
</body>
```

**Outcome:**




**BUTTONS:**

Buttons are most commonly used to submit, clear, reset a form data and also used to trigger client side scripts. We can create buttons in three ways as shown below:

**Using an <input> element with a type attribute whose value is submit, reset, or button**

In this method, the type attribute can take the following values:

- **submit**, which creates a button which automatically submits a form

```
<input type="submit" name="sub" value="Submit">
```

- **reset**, which creates a button that automatically resets form controls to their initial values

```
<input type="reset" name="res" value="Reset">
```

- **button**, which creates a button that is used to trigger a client side script

```
<input type="button" name="but" value="Submit">Example:
```

```
<form action = "">
```

```
<p>
```

```
<input type = "submit" value = "Submit Form" />
```

```
<input type = "reset" value = "Reset Form" />
```

```
</p>
```

```
</form>
```

**Outcome:**

## Get and Post Method

There are many differences between the Get and Post request. Let's see these differences:

GET	POST
1) In case of Get request, only <b>limited amount of data</b> can be sent because data is sent in header.	In case of post request, <b>large amount of data</b> can be sent because data is sent in body.
2) Get request is <b>not secured</b> because data is exposed in URL bar.	Post request is <b>secured</b> because data is not exposed in URL bar.
3) Get request <b>can be bookmarked</b> .	Post request <b>cannot be bookmarked</b> .
4) Get request is <b>idempotent</b> . It means	Post request is <b>non-idempotent</b> .

second request will be ignored until response of first request is delivered	
5) Get request is <b>more efficient</b> and used more than Post.	Post request is <b>less efficient</b> and used less than get.

## GET and POST

Two common methods for the request-response between a server and client are:

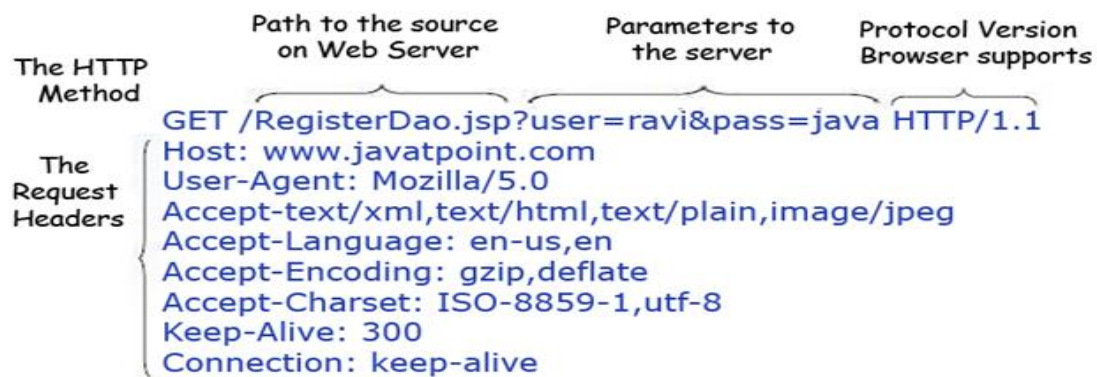
- **GET**- It requests the data from a specified resource
- **POST**- It submits the processed data to a specified resource

## Anatomy of Get Request

The query string (name/value pairs) is sent inside the URL of a GET request:

**GET/RegisterDao.jsp?name1=value1&name2=value2**

As we know that data is sent in request header in case of get request. It is the default request type. Let's see what information is sent to the server.



Some other features of GET requests are:

- It remains in the browser history
- It can be bookmarked
- It can be cached
- It have length restrictions
- It should never be used when dealing with sensitive data
- It should only be used for retrieving the data

## Anatomy of Post Request

The query string (name/value pairs) is sent in HTTP message body for a POST request:

**POST/RegisterDao.jsp HTTP/1.1**

**Host: www.javatpoint.com**

**name1=value1&name2=value2**

As we know, in case of post request original data is sent in message body. Let's see how information is passed to the server in case of post request.



Some other features of POST requests are:

- This requests cannot be bookmarked
- This requests have no restrictions on length of data
- This requests are never cached
- This requests do not retain in the browser history

## INTRODUCTION TO HTML 5

HTML 5 is the fifth revision of the HTML standard. It offers new features that provide not only rich media support but also enhance support for creating web applications that can interact with users, their local data, and servers more easily and effectively than was previously possible. It has improved the markup available for documents and has introduced application programming interfaces (API) and Document Object Model (DOM).

### Features:

1. It has introduced new multimedia features which supports audio and video controls by using `<audio>` and `<video>` tags.
2. There are new graphics elements including vector graphics and tags.
3. Drag and drop- the user can grab an object and drag it.
4. Geo-location services- it helps to locate the geographical location of a client.
5. Web storage facility which provides web application methods to store data on web browser.
6. Uses SQL database to store data offline.
7. Allows drawing various shapes like triangle, rectangle, circle, etc.
8. Capable of handling incorrect syntax.
9. Easy doctype declaration i.e. `<!doctype html>`
10. Easy character encoding i.e. `<meta charset="utf-8">`

### The *audio* element

The `<audio>` tag is used to embed sound content in a document, such as music or other audio streams. The `<audio>` tag contains one or more `<source>` tags with different audio sources. The browser will choose the first source it supports. The text between the `<audio>` and `</audio>` tags will only be displayed in browsers that do not support the `<audio>` element. There are three supported audio formats in HTML: MP3, WAV, and OGG.

#### Usage:

```
<audio controls= -controls||>
    <source src="horse.ogg" type="audio/ogg">
    <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio tag.
</audio>
```

The only commonly used attribute of the audio element is *controls*, which we always set to `-controls||`. When this attribute is used, it creates a display of start/stop button, a clock, a slider of the progress of the play, that total time of the audio file and a slider for volume control.

Attribute	Value	Description
Autoplay	autoplay	Specifies that the audio will start playing as soon as it is ready
Controls	controls	Specifies that audio controls should be displayed (such as a play/pause button etc)
Loop	loop	Specifies that the audio will start over again, every time it is finished
muted	muted	Specifies that the audio output should be muted

### The *video* element

The `<video>` tag is used to embed video content in a document, such as a movie clip or other video streams. The `<video>` tag contains one or more `<source>` tags with different video sources. The browser will choose the first source it supports. The text between the `<video>` and `</video>` tags will only be displayed in browsers that do not support the `<video>` element. There are three supported video formats in HTML: MP4, WebM, and OGG.

**Usage:**

```
<video width="320" height="240" controls = -controls||>  
  <source src="movie.mp4" type="video/mp4">  
  <source src="movie.ogg" type="video/ogg">  
  Your browser does not support the video tag.  
</video>
```

Attribute	Value	Description
autoplay	autoplay	Specifies that the video will start playing as soon as it is ready
controls	controls	Specifies that video controls should be displayed (such as a play/pause button, volume control, a slider to show the progress of play, total time of the video file).
height	pixels	Sets the height of the video player
loop	loop	Specifies that the video will start over again, every time it is finished
width	pixels	Sets the width of the video player

### The *time* element

The `<time>` tag is used to display the human-readable date/time. It can also be used to encode dates and times in a machine-readable form. The main advantage for users is that they can offer to add birthday reminders or scheduled events in their calendars' and search engines can produce smarter search results.

#### Usage:

```
<time attribute>
```

```
Time.....
```

```
</time>
```

This tag contains single attribute *datetime* which is used to define the date/time in a machine-readable form of the `<time>` element.

#### Example:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>The time element</h1>
```

```
<p>Open from 10:00 to <time>21:00</time> every weekday.</p>
```

```
<p>I have a date on <time datetime="2008-02-14 20:00">Valentines day </time>.</p>
```

```
</p>
```

```
<p><b> Note: </b>
```

```
The time element does not render as anything special in any of the major browsers.</p>
```

```
</p>
```

```
</body>
```

```
</html>
```

### The *article* element

The `<article>` element represents a section of content that forms an independent part of a document, such as a blog post, article, or other self-contained unit of information that may be linked to or included in some other content body.

#### Example:

```
<article>
```

```
<h1>Introduction to HTML</h1>
```

```
<p>HTML is a markup language that is used for creating web pages.</p>
```

```
</article>
```

**The *section* element**

The <section> element defines a section of a document, such as header, footer etc.

**Example:**

```
<section>
    <h1>Welcome to Our Website</h1>
    <p>Welcome and thank you for taking the time to visit our website.</p>
</section>
```

**CASCADING STYLE SHEETS**

The cascading style sheet is markup language used in web document for presentation purpose. The purpose of CSS is to separate web content from web presentation.

**Advantages:**

1. By combining CSS with HTML document, considerable amount of flexibility into the content submission can be achieved.
2. Separating out the style from actual contents help in managing large scale complex sites. So, CSS facilitates publication of contents in multiple presentation formats.
3. If CSS is used effectively then global style sheet can be applied to a web document. This helps maintaining consistency in web document.
4. If a small change needs to be done in the style of web content, then CSS makes it more convenient.

A Cascading Style Sheet is a collection of CSS style rules.

Each style rule in the rule list has two parts:

1. Selector, which indicates the element or elements affected by rule
2. A list of property – value pairs

*Usage:*

**selector { property : value ; property : value ; ..... }**

**h1 { font – family : arial } →**This CSS rule applies to all <h1> elements

**LEVELS OF CASCADING STYLE SHEETS**

There are three levels of style sheets, from lowest to highest in order:

1. *Inline Style Sheet*
2. *Embedded Style Sheet/Document Level Style Sheet*
3. *External Style Sheet*

Inline style sheets are applied to single HTML Element. Document Level Style sheets apply to the whole body of the document. External Style sheets can apply to bodies of any number of

documents. Inline style sheets have precedence over document level, which have precedence over external style sheets.

**Inline Style Sheet:**

Internal styles are styles that are written directly in the HTML tag on the document. The normal rules of CSS apply inside the style attribute. Each CSS statement must be separated with a semicolon ";" and colons appear between the CSS property and its value.

For example,

```
<p style="background: blue; color: white;">
```

A new background and font color with inline CSS

```
</p>
```

**Advantages of Inline Styles**

1. Inline styles have the highest precedence. That means they are going to be applied no matter what. The only styles that have higher precedence than inline style.
2. Inline styles are easy and quick to add.

**Disadvantages of Inline Styles**

1. Inline styles must be applied to every element you want them on.
2. It's impossible to style pseudo-elements and -classes with inline styles.

**Example:**

```
<html>
```

```
<head>
```

```
<title>Inline cascading style sheets</title>
```

```
</head>
```

```
<body>
```

```
<p> This is simple text</p>
```

```
<p style="font-size:30pt; font-family:script">This text is different</p>
```

```
<p style="font-size:50pt; color:#ff0000">This text is colored.</p>
```

```
</body>
```

```
</html>
```

**Outcome:**



**Embedded Style Sheet:**

Embedded Style Sheets refer to embed style sheet information into an HTML document using the style element. Embedding the style sheet information within `<style>...</style>` tags which appears only in the head section of your document.

**Advantages:**

The Embedded style sheet helps to decide the layout of the webpage.

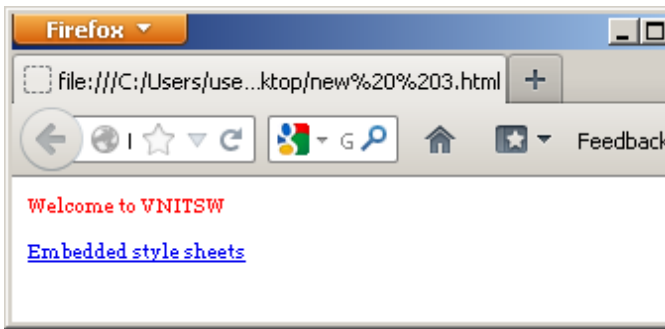
It is Useful when we want to apply the **unique style sheet** for the webpage.

**Disadvantages:**

When we want to apply the style to more than one documents at a time then this style sheet is of no use.

**Example:**

```
<html>
<head>
  <style type="text/css">
    p { font-family: georgia; font-size: x-small;color:red}
  </style>
</head>
<body>
  <p>Welcome to VNITSW</p>
  <p>
    <a href=" C:\Users\user21\Desktop\vignan.jpg" target="_blank">Embedded style sheets</a>
  </p>
</body>
</html>
```



### External Style Sheet:

In those times when we need to apply particular style to more than one web document, in such cases external style sheets can be used. The central idea of style sheet is defined in .css file. The style defined in .css file will be applied to all the web pages by using LINK tag.

External Style Sheet is specified using the LINK element within the HEAD element block to specify the URL location of the External Style Sheet. URL values may be relative or absolute.

#### Usage:

```
<link rel="stylesheet" type="text/css" href="[Style sheet URL]">
```

#### Example:

Below is the style sheet file (ex1.css):

```
body {background-color:tan;}
h1 {color:maroon;font-size:20pt;}
hr {color:navy;}
p {font-size:11pt;margin-left:15px;}
```

Below is the html file (.html)

```
<html>
<head>
  <link rel= -stylesheet type= -text/css href= -ex1.css//>
</head>
<body>
  <h1>This is header1</h1><br>
  <p>this is paragraph</p><br>
</body>
</html>
```

#### Outcome:



## STYLE SPECIFICATION FORMATS

The format of style specification depends on the level of style sheet.

- ❖ Inline style specifications appear as values of the *style* attributes of a tag, the general form of which is as follows:

*style = -property\_1:value\_1; property\_2:value\_2; .....property\_n:value\_n;||*

- ❖ Document style specifications appear as the content of a *style* element within the head section of the document. The general form of the content of a *style* elements is as follows:

<style type= -text/css||>

Rule list

</style>

The **type attribute** of the <style> tag tells the browser the type of the style specification for CSS.

- Each style rule in the rule list has two parts:
  1. Selector, which indicates the element or elements affected by rule
  2. A list of property – value pairs
- Following is the form of style rule:

**selector**

```
{
    property_1:value_1;
    property_2:value_2;
    .....
    property_n:value_n;
}
```

- ❖ An external style sheet consists of a list of style rules of the same form as in document style

sheets, but the <style> tag is not included. All the style information is placed in separate file, created with the extension .css and this can be used for multiple html documents. This .css file is linked to the html document by including <link> tag in the head section of the html, which can be written as follows:

```
<link rel="stylesheet" type="text/css" href="[Style sheet URL]">
```

## **SELECTOR FORMS**

A selector specifies the elements to which the style information applies. The selector can have a variety of forms.

### **1. Simple Selector Forms**

The simplest selector form is a single element name, such as h1, h2, p, etc...

In this case the property values in the rule apply to all the occurrences of the named element.

Example:

```
h1{
```

```
font-family:Arial;
```

```
font-size:20px;
```

```
}
```

```
<html>
```

```
<head>
```

```
<title>type selector</title>
```

```
<style type="text/css">
```

```
    h1, h2, p {
```

```
        font-family:Arial;
```

```
        font-size:20px;
```

```
    }
```

```
</style>
```

```
<body>
```

```
    <h1>Vignan Nirula</h1>
```

```
    <h2>Vignan Nirula</h2>
```

```
    <p>Vignan Nirula</p>
```

```
</body>
```

```
</html>
```

### **2. Class Selectors**

Class selectors are used to allow different occurrences of the same tag to use different style

specifications. A style class is defined in a style element by giving the style class a name, which is attached to the tag's name with a period.

**Example:**

```
<html>
<head>
<title>class selector</title>
<style type="text/css">
    h1.redtext
    {
    font-family:monotype corsiva;
    color:red;
    font-size:14pt;
    }
    h1.bluetext
    {
    font-family:times new roman;
    color:blue;
    font-size:16pt;
    }
</style>
</head>
<body>
    <h1 class="redtext">Vignan Nirula</h1>
    <h2> Andhra pradesh</h2>
    <h1 class="bluetext">Vignan Nirula</h1>
</body>
</html>
```

**3. Generic Selectors**

To specify the class of style specifications to the content of more than one tag, generic selectors are used. It is defined without a tag name in its selector. Without tag name, the name of the generic class begins with a period, as in the following example:

```
<html>
<head>
    <title> Generic Selector</title>
```

```
<style type="text/css">
.text
{
    font-family:Arial;
    color:black;
}
</style>
</head>
<body>
    <h1 class="text">Vignan Nirula</h1>
    <p> Women's</p>
    <h2 class="text">Engineering</h2>
    <p class="text">college</p>
</body>
</html>
```

#### 4. ID Selectors

The ID selector is similar to the class selector but only the difference between the two is that class selector can be applied to more than one elements where as using the ID selector the style can be applied to one specific element.

The syntax is as follows.

**#name\_of\_ID {property:value list;}**

##### Example:

```
<html>
<head>
    <title>ID selector</title>
    <style type="text/css">
        #top
        {
            font-family: monotype corsiva;
            color: red;
            font-size:14pt;
        }
    </style>
</head>
```

```
<body>
    <p id="top"> Vignan's Engineering college</p>
</body>
</html>
```

## 5. Contextual Selectors

Selectors can specify that the style should apply only to elements in certain positions in the document. The simplest form of contextual selector is the descendant selector. Element B is a descendant of element A if it appears in the content of A. And A is the ancestor of B.

A particular element in the document can be selected by listing one or more ancestors of the element in the selector, with only white space separating the element names.

Below is the example for style rule that applies its style only to the content of ordered list elements that are descendants of unordered list elements in the document:

*ul ol {property – value list}*

## 6. Pseudo Classes

Pseudo Classes specify that certain style rules apply when something happens. Some of the pseudo classes are as follows:

- a) *link* pseudo class is used to style a link that has not been visited
- b) *visited* pseudo class is used to style a link that has been visited
- c) *hover* pseudo class is used to specify the style that is to be applied when mouse cursor moves over it
- d) *focus* pseudo class is used to specify the style that is to be applied when the associated element is focused

**Example:** *h2: hover {property – value list}*

## 7. Universal Selector

If you want a style rule to apply to all elements, you can use this selector. It is denoted by an asterisk (\*)

**Example:**

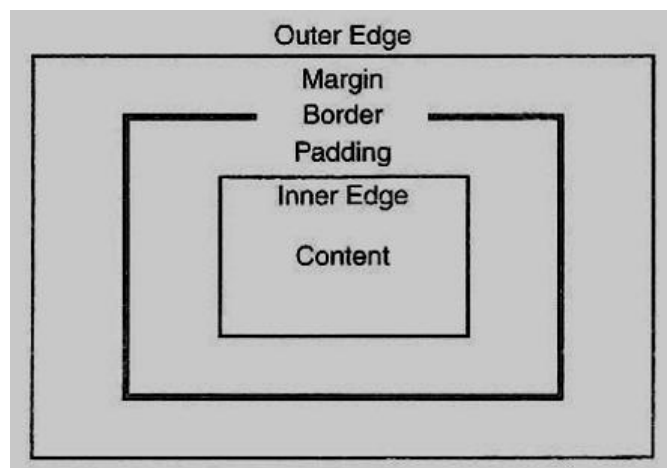
```
<html>
<head> <title> Universal Selector </title>
    <style type="text/css">
        * {
            color:green
        }
    </style>
</head>
```

```
<body>
  <h1> Vignan's Lara Engineering College</h1>
  <ul type="disc">
    <li>Dept of CSE</li>
    <li>Dept of IT</li>
    <li>Dept of ECE</li>
    <li>Dept of EEE</li>
  </ul>
</body>
</html>
```

## THE BOX MODEL

Virtually all document elements can have borders. These borders have various styles such as color and width. The amount of space between the content of an element and its border known as padding, as well as the space between the border and an adjacent element known as margin.

This model of document is shown as below:



### **Borders**

Every element has a property i.e. *border – style*, which controls whether the element content has a border as well as the style of the border. Border style may be *dotted*, *dashed*, *solid*, *double* and the default value is *none*.

The style of the each side of the border can be set with properties - *Border-top-style*, *Border-bottom-style*, *Border-left-style* and *Border-right-style*.

And the color of the border can be set with the property *border – color*. The color of the each side of the border can also be set with properties – *border – right – color*, *border – top – color*, *border – left – color*, *border – bottom – color*.



*Border – width* property is used to specify the thickness of the border. Its possible values are *thin*, *medium*, *thick* or a *value* in pixels.

**Example**

```
<html>
<head>
  <style type="text/css">
    p {    border-style:double;
           border-width:thick;
           border-color:green;
           font-size:20px;
        }
  </style>
</head>
<body>
  <p>
    Properties of border are border style, border color, border width
  </p>
</body>
</html>
```

**Margins and Padding**

Margin is the space between the border of an element and the element's neighbor. Padding is the space between the content of an element and its border. When there is no border, the margin plus the padding is the space between the content of an element and its neighbor.

The margin properties are named *margin*, which applies to all four sides of an element, four sides of the margin can also be set individually with the four properties, *margin – left*, *margin – right*, *margin – top*, *margin – bottom*. Similarly, padding properties are *padding*, *padding – left*, *padding – right*, *padding – top*, *padding – bottom*.

And the values for these properties can be specified in terms of units such as *px* for pixels, *in* for inches, *cm* for centimeters, *mm* for millimeters, *pt* for points (a point is 1/72 inch), and *pc* for picas (12 points).

**Example:**

```
<html>
<head>
  <style type="text/css">
    p.one
    {
      margin:0.2in;
      padding:0.2in;
      border-color:green;
      border-style:double;
    }
    p.two
    {
      margin:0.1in;
      padding:0.3in;
      border-color:green;
      border-style:double;
    }
    p.three
    {
      margin:0.3in;
      padding:0.1in;
      border-color:green;
      border-style:double;
    }
  </style>
</head>
<body>
  <p class="one">
    Now is the time for all good programmers to learn to use style sheets
  </p>
  <p class="two">
    Now is the time for all good programmers to learn to use style sheets
  </p>
```

```
<p class="three">  
    Now is the time for all good programmers to learn to use style sheets  
</p>  
</body>  
</html>
```

**Outcome:**

now is the time for all good programmers to learn to use style sheets

now is the time for all good programmers to learn to use style sheets

now is the time for all good programmers to learn to use style sheets

## **CONFLICT RESOLUTION**

When there are two different values for the same property on the same element then there is conflict that the browser must resolve. There are different situation where the conflict can occur:

1. When style sheets at two or more level specify different values for the same property on the same element, this kind of conflict is resolved by the precedence of the different levels of style sheets. Inline style sheets have precedence over embedded style sheets which have more precedence than external style sheets.
2. Sometimes, a conflict can occur within the single style sheet itself.

For example,

```
h3 {  
    color:blue;  
};  
body h3 {  
    color:green;  
};
```

Here it is observed that the same style rule is applied to the h3 element, but the last one has precedence over the preceding one.

3. Inheritance is another source for conflicts. These can occur if a property on a particular element has a property value and also inherits a value for the same property from its parent element. Then child element style rule has precedence over the parent element style rule.

For example,

```
<html>
<head>
  <style>
    i{
      color:green;
    }
    h1{
      color:red
    }
  </style>
</head>
<body>
  <h1>This is <i>inheritance</i> style</h1>
</body>
</html>
```

It is observed that the color property for <h1> element is set to red, which is also inherited to <i> element but it is given less precedence over the color property of <i> element, which is set to green.

4. Conflicts may come from a style sheet written by the author of the document itself, but from the browser user, and the browser too.
5. Property value specifications can be marked as being **important**, by including *!important* in the specification.

For example,

```
p.special { font – style: italic !important; font-size:14;}
```

In this specification, font-style: italic is important, but the font-size: 14 is normal. Whether a specification has been marked as being important is called weight of the specification. This weight can be normal or important. This is also one of the ways to resolve the conflicts.

Here the conflict resolution is a multi stage process:

First, it is to gather all style specifications from the three possible levels of style sheets. These specifications are sorted into order of precedence of style sheet levels.

Next, all of available specifications are sorted by origin and weight according to the following rules (rules are given in precedence order):

- a. Important declarations with user origin

- b. Important declarations with author origin
  - c. Normal declarations with author origin
  - d. Normal declarations with user origin
  - e. Any declarations with browser origin
6. If there are conflicts after sorting described above, the next step to resolve the conflicts is done through specificity rule of selectors. This sort is based on the following rules of precedence:
- a. Id selectors
  - b. Class and pseudo-class selectors
  - c. Contextual selectors
  - d. Universal selectors
7. If there are still conflicts, they are resolved by given precedence to the most recently seen specification.

This whole sorting process that is used to resolve style specification conflicts is called **cascade**.

## CSS3

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS3 is a latest standard of css earlier versions(CSS2). The main difference between css2 and css3 is follows –

- Media Queries
- Namespaces
- Selectors Level 3
- Color

### **CSS3 modules**

CSS3 is collaboration of CSS2 specifications and new specifications, we can called this collaboration is **module**. Some of the modules are shown below –

- Selectors
- Box Model
- Backgrounds
- Image Values and Replaced Content
- Text Effects
- 2D Transformations
- 3D Transformations
- Animations
- Multiple Column Layout
- User Interface

## **DHTML**

### **POSITIONING, MOVING AND CHANGING ELEMENTS**

DHTML stands for Dynamic Hypertext Markup language i.e., Dynamic HTML. Dynamic HTML is not a markup or programming language but it is a term that combines the features of various web development technologies for creating the web pages dynamic and interactive.

#### **Differences between HTML and DHTML**

- ❖ HTML (Hyper Text Markup Language) is the actual markup language web pages are displayed in. It creates static content on web page. DHTML stands for Dynamic HTML and it's using HTML, CSS and JavaScript together to create web pages that are not static displays only, but the content is actually dynamic.
- ❖ HTML does not allow altering the text and graphics on the web page unless the web page gets changed. DHTML allows altering the text and graphics of the web page and for that matter you need not have to change the entire web page.
- ❖ Creation of HTML is simplest and not interactive. Creation of DHTML is complex but more interactive.

#### **HTML**

1. It is referred as a static HTML and static in nature.
2. A plain page without any styles and Scripts called as HTML.
3. HTML sites will be slow upon client-side technologies.

#### **DHTML**

1. It is referred as a dynamic HTML and dynamic in nature.
2. A page with HTML, CSS, DOM and Scripts called as DHTML.
3. DHTML sites will be fast enough upon client-side technologies.

#### **POSITIONING ELEMENTS IN DHTML**

The position property in CSS tells about the method of positioning for an element or an HTML entity. There are five different types of position property available in CSS:

1. Absolute
2. Relative
3. Static

The positioning of an element can be done using the top and left property.

#### **ABSOLUTE**

The absolute value for the position is specified when the element is to be placed at a specific location in the document without regard to the position of other elements.

For example, if a paragraph of text is to be appeared 100 pixels from the left edge and the 200 pixels

from the top edge of the display window, the following element could be used:

```
<p style = "position: absolute; left: 100px; top: 200px;" >
```

```
-----
```

```
-----
```

```
</p>
```

**Example:**

```
<html>
```

```
<head>
```

```
<title> Absolute Positioning </title>
```

```
</head>
```

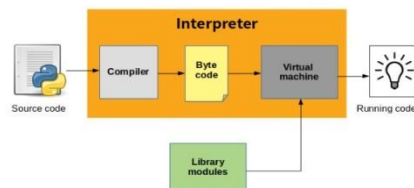
```
<body>
```

```

```

```
<body>
```

```
</html>
```



**RELATIVE**

An element with position: relative is positioned relatively with the other elements. If the left and top properties are given values, they displace the element by the specified amount from the position where it would have been placed if the top and left properties had not set.

```
<p style = "position: relative; top: 100px;">
```

```
-----  
-----
```

```
</p>
```

**Example:**

```
<html>
```

```
<head>
```

```
<title> Absolute Positioning </title>
```

```
</head>
```

```
<body>
```

```
<pre>
```

An element with position: relative is positioned relatively with the other elements.

If the left and top properties are given values, they displace the element by the specified amount from the position where it would have been placed if the top and left properties had not set.

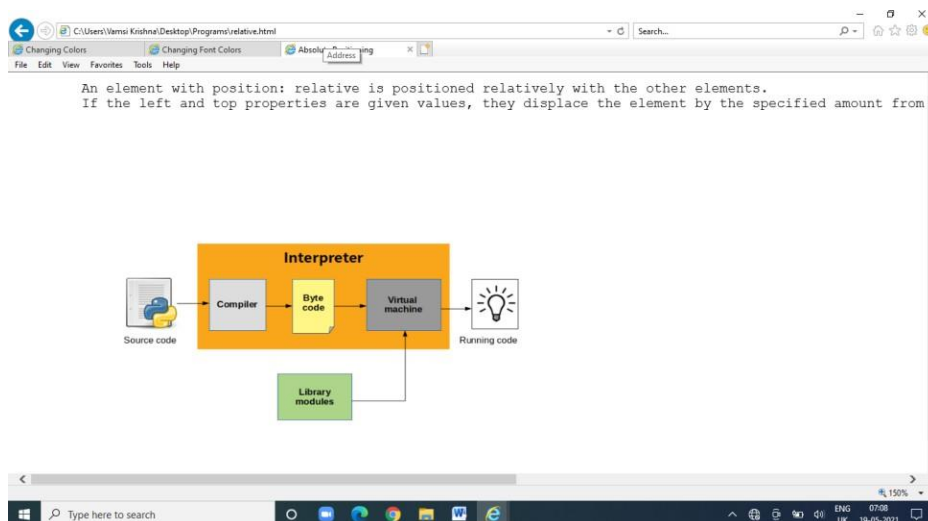
```
</pre>
```

```

```

```
<body>
```

```
</html>
```

**Outcome:****STATIC**

The default value for the position property is static. A statically positioned element is placed in the document as if it had the position of relative but no values for top and left properties is given. A statically positioned element has not left and top properties and it cannot be moved from its position later.



## **MOVING ELEMENTS IN DHTML**

An HTML Element whose position property is set to relative and absolute can be moved. Moving of an element is simple. It can be achieved by changing the top and left properties values of an element.

- If the position is absolute, then the element moves to the new values of top and left.
- If the position is relative, then the element moves from its original position by distances given by distances given by the new values of top and left.

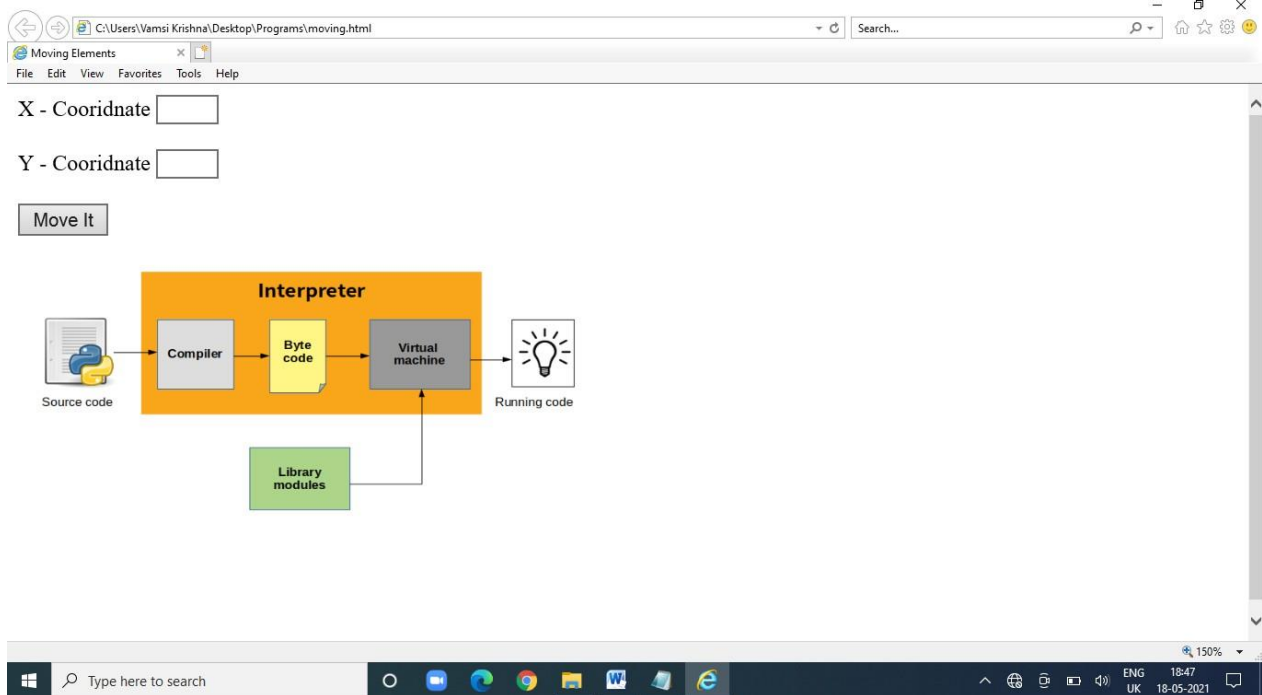
### **Example:**

```
<html>
<head>
<title> Moving Elements </title>
<script type="text/javascript">
function moveIt()
{
image=document.getElementById("python").style;
x=document.myform.xcoord.value;
y=document.myform.ycoord.value;
image.top=x;
image.left=y;

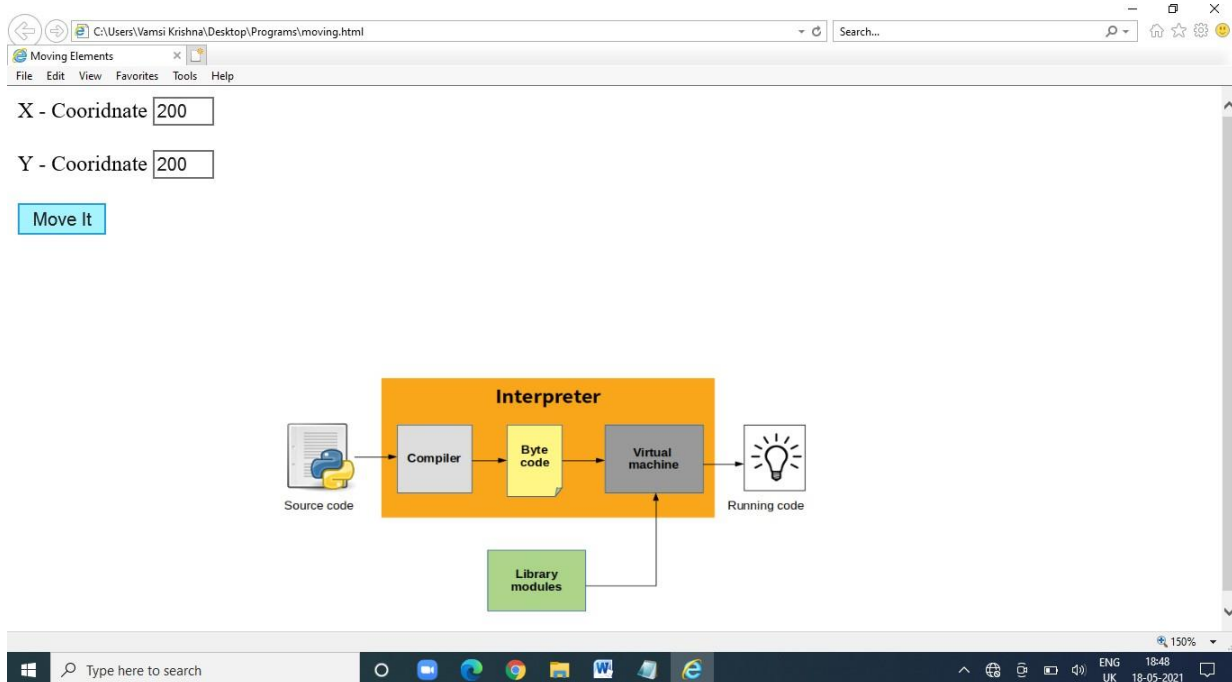
}
</script>
</head>
<body>
<form name="myform">
<label> X - Coordnate</label>
<input type="text" id="xcoord" size="3"/>
<br/><br/>
<label> Y - Coordnate</label>
<input type="text" id="ycoord" size="3"/>
<br/><br/>
<input type="button" value="Move It" onclick="moveIt()" />
</form>

</body>
</html>
```

Initially when it is loaded, it should be as follows:



Input the desired values for the X and Y Coordinates and click **Move It** button to move the image element.



## CHANGING THE COLORS AND FONTS

The background and foreground colors of the document displayed can be dynamically changed and also the font properties of the text can be changed.

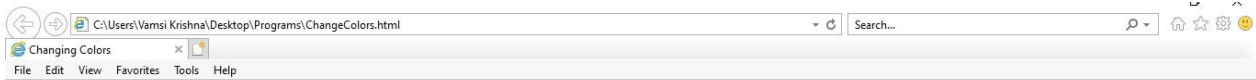
### CHANGING COLORS

- To change the background color of the document **backgroundColor** property value is changed.
- To change the foreground color of the document then the **color** property value is changed.

#### Example:

```
<html>
<head>
<title> Changing Colors </title>
<script type="text/javascript">
function changeback()
{
newColor=document.cform.background.value;
document.body.style.backgroundColor=newColor;
}
function changefore()
{
newColor=document.cform.foreground.value;
document.body.style.color=newColor;
}
</script>
</head>
<body>
<p style="fontfamily:Times; font-style:italic; font-size:e2em;">
This small page illustrated dynamic setting of the
foreground and backgorund colors for a document
</p>

<form name="cform">
<label> Background Color </label>
<input type="text" name="background" size="10" onchange="changeback()" />
<br/><br/>
<label> Foreground Color </label>
<input type="text" name="foreground" size="10" onchange="changefore()" />
</form>
</body>
</html>
```

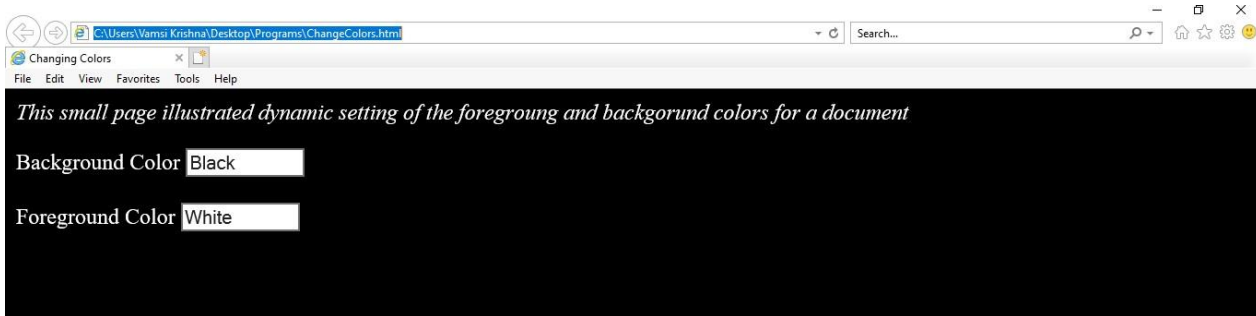
**Outcome:**

*This small page illustrated dynamic setting of the foreground and background colors for a document*

Background Color

Foreground Color

After entering desired colors for the background and foreground colors, then web page looks like as follows:

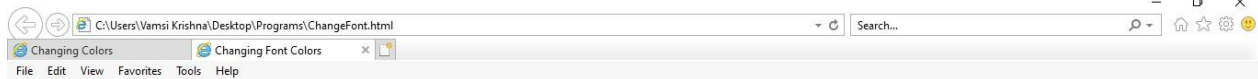
**CHANGING FONTS**

Changing the fonts in the document can be changed dynamically by making change in the font style properties – *color*, *fontStyle*, *fontSize*.

How these style properties of font values are changed in response to the user actions through mouse events – *onmouseover* and *onmouseout* is illustrated below:

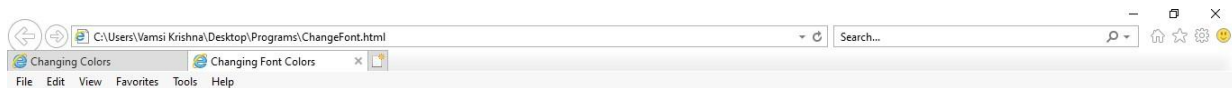
```
<html> <head> <title> Changing Font Colors </title>
<style type="text/css">
.regText{ font: 1.1em 'Times New Roman'; }
.wordText{ color:blue; }
</style> </head>
<body>
<p class="regText">The
state of
<span class="wordText"; onmouseover="
this.style.color='red';
                this.style.fontStyle='italic';
                this.style.fontSize='2em'; ";
onmouseout=" this.style.color='blue';
                this.style.fontStyle='normal';
                this.style.fontSize='1.1em'; "; >
Jammu & Kashmir
</span> produces many of our nation's Apples </p> </body> </html>
```

## Outcome



The state of **Jammu & Kashmir** produces many of our nation's Apples

On moving the mouse over the text “**Jammu & Kashmir**”, the font style properties of that text can be changed as follows:



The state of *Jammu & Kashmir* produces many of our nation's Apples

## CASCADING STYLE SHEETS

- CSS stands for Cascading Style Sheets. CSS defines how HTML elements are to be displayed.
- The purpose of CSS is to separate web content from web presentation.
- CSS is a simple mechanism for controlling the style of a web document without compromising its structure.

### Advantages of CSS:

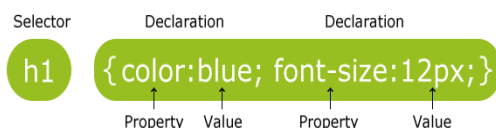
1. **CSS saves time** – when most of us first learn HTML, we taught to set the font face, size, color, style etc. every time it occurs on a page. This means typing the same thing over & over again. With CSS, you have to specify the details once for any element. CSS will automatically apply the specific whenever the element occurs.
2. **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.
3. **Easy maintenance** – To make a global change of an element, make an edit in one place, and all elements in all the web pages will be updated automatically.
4. **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.
5. **Multiple Device Compatibility** – 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.
6. It provides you a attractive look to your website.

### Disadvantages of CSS:

**Browser compatibility:** CSS works differently on different browsers. This means that some stylesheet features are supported and some are not.

### Syntax for CSS:

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:



- **Selector** – A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
- The **declaration block** contains one or more declarations separated by semicolons. Each declaration includes a property name and a value, separated by a colon.

- **Property** - A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be *color*, *border* etc.
- **Value** - Values are assigned to properties. For example, *color* property can have value blue here.

## **LEVELS OF CASCADING STYLE SHEETS**

There are three levels of style sheets, from lowest to highest in order:

1. *Inline Style Sheet*
2. *Embedded Style Sheet/Document Level Style Sheet/ Internal Style Sheet*
3. *External Style Sheet*

Inline style sheets are applied to single HTML Element. Document Level Style sheets apply to the whole body of the document. External Style sheets can apply to bodies of any number of documents. Inline style sheets have precedence over document level, which have precedence over external style sheets.

### **Inline Style Sheet:**

Internal styles are styles that are written directly in the HTML tag on the document. The normal rules of CSS apply inside the style attribute. Each CSS statement must be separated with a semicolon ";" and colons appear between the CSS property and its value.

For example,

```
<p style="background: blue; color: white;">
```

A new background and font color with inline CSS

```
</p>
```

### **Advantages of Inline Styles**

1. Inline styles have the highest precedence. That means they are going to be applied no matter what. The only styles that have higher precedence than inline style.
2. Inline styles are easy and quick to add.

### **Disadvantages of Inline Styles**

1. Inline styles must be applied to every element you want them on.
2. It's impossible to style pseudo-elements and -classes with inline styles.

### **Example:**

```
<html>
```

```
<head>
```

```
<title>Inline cascading style sheets</title>
```

```
</head>
```

```
<body>
```

```
<p> This is simple text</p>
<p style="font-size:30pt; font-family:script">This text is different</p>
<p style="font-size:50pt; color:#ff0000">This text is colored.</p>
</body>
</html>
```

**Outcome:****Embedded Style Sheet:**

Embedded Style Sheets refer to embed style sheet information into an HTML document using the style element. Embedding the style sheet information within **<style>...</style>** tags which appears only in the head section of your document.

**Advantages:**

The Embedded style sheet helps to decide the layout of the webpage.

It is Useful when we want to apply the **unique style sheet** for the webpage.

**Disadvantages:**

When we want to apply the style to more than one document at a time then this style sheet is of no use.

**Example:**

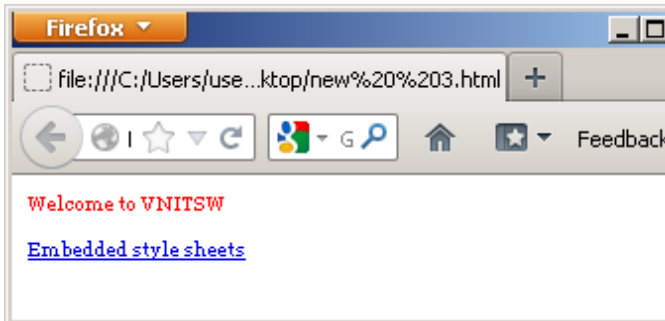
```
<html>
<head>
  <style type="text/css">
    p { font-family: georgia; font-size: x-small;color:red}
  </style>
</head>
<body>
  <p>Welcome to VNITSW</p>
  <p>
    <a href=" C:\Users\user21\Desktop\vignan.jpg" target="_blank">Embedded style sheets</a>
```



</p>

</body>

</html>



### External Style Sheet:

In those times when we need to apply particular style to more than one web document, in such cases external style sheets can be used. The central idea of style sheet is defined in .css file. The style defined in .css file will be applied to all the web pages by using LINK tag.

External Style Sheet is specified using the **LINK** element within the **HEAD** element block to specify the URL location of the External Style Sheet. URL values may be relative or absolute.

#### Usage:

**<link rel="stylesheet" type="text/css" href="[Style sheet URL]">**

#### Example:

**Below is the style sheet file (ex1.css):**

```
body {background-color:tan;}
h1 {color:maroon;font-size:20pt;}
hr {color:navy;}
p {font-size:11pt;margin-left:15px;}
```

**Below is the html file (.html)**

<html>

<head>

<link rel="stylesheet" type="text/css" href="ex1.css"/>

</head>

<body>

<h1>This is header1</h1><br>

<p>this is paragraph</p><br>

</body>

</html>

**Outcome:**

## **STYLE SPECIFICATION FORMATS**

The format of style specification depends on the level of style sheet.

- ❖ Inline style specifications appear as values of the ***style*** attributes of a tag, the general form of which is as follows:

*style = -property\_1:value\_1; property\_2:value\_2; .....property\_n:value\_n;||*

- ❖ Document style specifications appear as the content of a ***style*** element within the head section of the document. The general form of the content of a ***style*** elements is as follows:

<style type= -text/css||>

Rule list

</style>

The **type attribute** of the <style> tag tells the browser the type of the style specification for CSS.

- Each style rule in the rule list has two parts:
  1. Selector, which indicates the element or elements affected by rule
  2. A list of property – value pairs
- Following is the form of style rule:

**selector**

```
{
    property_1:value_1;
    property_2:value_2;
    .....
    property_n:value_n;
}
```

- ❖ An external style sheet consists of a list of style rules of the same form as in document style sheets, but the <style> tag is not included. All the style information is placed in separate file, created with the extension .css and this can be used for multiple html documents. This .css file is linked to the html document by including <link> tag in the head section of the html, which can be written as follows:

*<link rel="stylesheet" type="text/css" href="[Style sheet URL]">*

## **SELECTOR FORMS**

- CSS selectors allow you to select and manipulate HTML elements.
- CSS selectors are used to "find" (or select) HTML elements based on their id, class, type, attribute, and more.

### **a.The element Selector:**

The element selector selects elements based on the element name. For example, You can select all <p> elements on a page like this: (all <p> elements will be center-aligned, with a red text color).

#### **Example**

```
p {  
  text-align: center;  
  color: red;  
}
```

### **b.The Universal Selectors:**

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type .

```
* {  
  color: green;  
}
```

This rule renders the content of every element in our document in green.

### **c.The Descendant Selectors:**

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to <b> element only when it lies inside <p> tag.

```
p b {  
  color: green;  
}
```

### **d.The id Selectors:**

The id selector uses the id attribute of an HTML element to select a specific element. An id should be unique within a page, so the id selector is used if you want to select a single, unique element.

To select an element with a specific id, write a hash character, followed by the id of the element.

The style rule below will be applied to the HTML element with id="para1":

**Example**

```
<html>
<head>
<style>
p{
    color: red;
    text-align: center;
}
#para1{
    color: green;
    text-align: center;
}
</head>
</style>
<body>
<h1>this is heading</h1>
<p>this is a paragraph with centered text and font color of red.</p>
<h2>this is heading</h2>
<p id="para1">this is a paragraph with centered text and font color of green.</p>
<h3>this is heading</h3>

    <p>this is a paragraph with centered text and font color of red.</p>
</body>

</body>
```

**NOTE:** Do NOT start an ID name with a number.

**e.The class Selectors:**

The class selector selects elements with a specific class attribute. To select elements with a specific class, write a period character, followed by the name of the class. In the example below, all HTML elements with class="center" will be center-aligned.

**Example**

```
<html>
```

---

```
<head>
<style>
.center {
    text-align: center;
    color: green;
    background-color:yellow;
}
</head>
</style>
<body>
<h1 class="center">this is heading</h1>
<p>this is a paragraph.this is a paragraph.this is a paragraph.this is a paragraph.</p>
<h2>this is heading</h2>
<p>this is a paragraph.this is a paragraph.this is a paragraph.this is a paragraph.</p>
</body>
```

- You can also specify that only specific HTML elements should be affected by a class.In the example below, all <p> elements with class="center" will be center-aligned.

**Example**

```
p.center {
    text-align: center;
    color: red;
    background-color:yellow;
}
```

**NOTE:** Do NOT start a class name with a number.

**f.Grouping Selectors:**

If you have elements with the same style definitions, like this:

```
h1 {
    text-align: center;
    color: red;
}
h2 {
    text-align: center;
    color: red;
```

```
}  
p {  
  text-align: center;  
  color: red;  
}
```

- You can group the selectors, to minimize the code. To group selectors, separate each selector with a comma.

In the example below we have grouped the selectors from the code above:

**Example:**

```
h1, h2, p {  
  text-align: center;  
  color: red;  
}
```

**g.The Child Selectors:**

- You have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example:

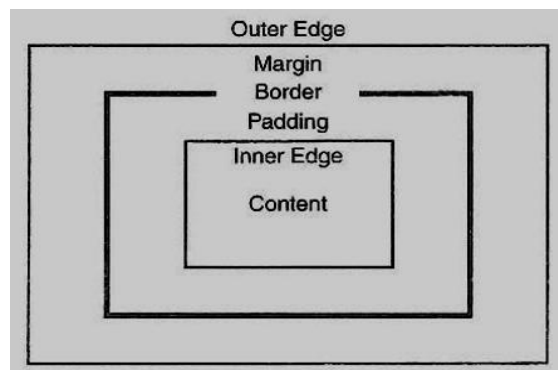
```
body > p {  
  color: black;  
}
```

- This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like <div> or <td> would not have any effect of this rule.

## **THE BOX MODEL**

Virtually all document elements can have borders. These borders have various styles such as color and width. The amount of space between the content of an element and its border known as padding, as well as the space between the border and an adjacent element known as margin.

This model of document is shown as below:



## Borders

Every element has a property i.e. *border – style*, which controls whether the element content has a border as well as the style of the border. Border style may be *dotted*, *dashed*, *solid*, *double* and the default value is *none*.

The style of the each side of the border can be set with properties - *Border-top-style*, *Border-bottom-style*, *Border-left-style* and *Border-right-style*.

And the color of the border can be set with the property *border – color*. The color of the each side of the border can also be set with properties – *border – right – color*, *border – top – color*, *border – left – color*, *border – bottom – color*.

*Border – width* property is used to specify the thickness of the border. It possible values are *thin*, *medium*, *thick* or a *value* in pixels.

## Example

```
<html>
<head>
  <style type="text/css">
    p {    border-style:double;
           border-width:thick;
           border-color:green;
           font-size:20px;
        }
  </style>
</head>
<body>
  <p>
    Properties of border are border style, border color, border width
  </p>
</body>
</html>
```

## Margins and Padding

Margin is the space between the border of an element and the element's neighbor. Padding is the space between the content of an element and its border. When there is no border, the margin plus the padding is the space between the content of an element and its neighbor.

The margin properties are named *margin*, which applies to all four sides of an element, four

sides of the margin can also be set individually with the four properties, *margin – left*, *margin – right*, *margin – top*, *margin – bottom*. Similarly, padding properties are *padding*, *padding – left*, *padding – right*, *padding – top*, *padding – bottom*.

And the values for these properties can be specified in terms of units such as *px* for pixels, *in* for inches, *cm* for centimeters, *mm* for millimeters, *pt* for points (a point is 1/72 inch), and *pc* for picas (12 points).



**Example:**

```
<html>
<head>
  <style type="text/css">
    p.one
    {
      margin:0.2in;
      padding:0.2in;
      border-color:green;
      border-style:double;
    }
    p.two
    {
      margin:0.1in;
      padding:0.3in;
      border-color:green;
      border-style:double;
    }
    p.three
    {
      margin:0.3in;
      padding:0.1in;
      border-color:green;
      border-style:double;
    }
  </style>
</head>
<body>
  <p class="one">
    Now is the time for all good programmers to learn to use style sheets
  </p>
  <p class="two">
    Now is the time for all good programmers to learn to use style sheets
```

</p>

```
<p class="three">  
    Now is the time for all good programmers to learn to use style sheets  
</p>  
</body>  
</html>
```

**Outcome:**

now is the time for all good programmers to learn to use style sheets

now is the time for all good programmers to learn to use style sheets

now is the time for all good programmers to learn to use style sheets

## **CONFLICT RESOLUTION**

When there are two different values for the same property on the same element then there is conflict that the browser must resolve. There are different situation where the conflict can occur:

1. When style sheets at two or more level specify different values for the same property on the same element, this kind of conflict is resolved by the precedence of the different levels of style sheets. Inline style sheets have precedence over embedded style sheets which have more precedence than external style sheets.
2. Sometimes, a conflict can occur within the single style sheet itself.

For example,

```
h3 {  
    color:blue;  
};  
body h3 {  
    color:green;  
};
```

Here it is observed that the same style rule is applied to the h3 element, but the last one has precedence over the preceding one.

3. Inheritance is another source for conflicts. These can occur if a property on a particular element has a property value and also inherits a value for the same property from its parent element. Then child element style rule has precedence over the parent element style rule.

For example,

```
<html>
<head>
  <style>
    i{
      color:green;
    }
    h1{
      color:red
    }
  </style>
</head>
<body>
  <h1>This is <i>inheritance</i> style</h1>
</body>
</html>
```

It is observed that the color property for <h1> element is set to red, which is also inherited to <i> element but it is given less precedence over the color property of <i> element, which is set to green.

4. Property value specifications can be marked as being **important**, by including *!important* in the specification.

For example,

```
p.special { font - style: italic !important; font-size:14;}
```

In this specification, font-style: italic is important, but the font-size: 14 is normal. Whether a specification has been marked as being important is called weight of the specification. This weight can be normal or important. This is also one of the ways to resolve the conflicts.

Here the conflict resolution is a multi stage process:

First, it is to gather all style specifications from the three possible levels of style sheets. These specifications are sorted into order of precedence of style sheet levels.

5. If there are still conflicts, they are resolved by given precedence to the most recently seen specification.

This whole sorting process that is used to resolve style specification conflicts is called **cascade**.

## CSS3

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. CSS3 is a latest standard of css earlier versions(CSS2). The main difference between css2 and css3 is follows –

- Media Queries
- Namespaces
- Selectors Level 3
- Color

### **CSS3 modules**

CSS3 is collaboration of CSS2 specifications and new specifications, we can called this collaboration is **module**. Some of the modules are shown below –

- Selectors
- Box Model
- Backgrounds
- Image Values and Replaced Content
- Text Effects
- 2D Transformations

- 3D Transformations
- Animations
- Multiple Column Layout
- User Interface

## **DHTML**

### **POSITIONING, MOVING AND CHANGING ELEMENTS**

DHTML stands for Dynamic Hypertext Markup language i.e., Dynamic HTML. Dynamic HTML is not a markup or programming language but it is a term that combines the features of various web development technologies for creating the web pages dynamic and interactive.

#### **Differences between HTML and DHTML**

- ❖ HTML (Hyper Text Markup Language) is the actual markup language web pages are displayed in. It creates static content on web page. DHTML stands for Dynamic HTML and it's using HTML, CSS and JavaScript together to create web pages that are not static displays only, but the content is actually dynamic.
- ❖ HTML does not allow altering the text and graphics on the web page unless the web page gets changed. DHTML allows altering the text and graphics of the web page and for that matter you need not have to change the entire web page.
- ❖ Creation of HTML is simplest and not interactive. Creation of DHTML is complex but more interactive.

#### **HTML**

1. It is referred as a static HTML and static in nature.
2. A plain page without any styles and Scripts called as HTML.
3. HTML sites will be slow upon client-side technologies.

#### **DHTML**

1. It is referred as a dynamic HTML and dynamic in nature.
2. A page with HTML, CSS, DOM and Scripts called as DHTML.
3. DHTML sites will be fast enough upon client-side technologies.

#### **POSITIONING ELEMENTS IN DHMTL**

The position property in CSS tells about the method of positioning for an element or an HTML entity. There are three different types of position property available in CSS:

1. Absolute
2. Relative
3. Static

The positioning of an element can be done using the top and left property.

### **ABSOLUTE**

The absolute value for the position is specified when the element is to be placed at a specific location in the document without regard to the position of other elements.

For example, if a paragraph of text is to be appeared 100 pixels from the left edge and the 200 pixels from the top edge of the display window, the following element could be used:

```
<p style = "position: absolute; left: 100px; top: 200px;" >
```

```
-----
```

```
-----
```

```
</p>
```

### **Example:**

```
<html>
```

```
<head>
```

```
<title> Absolute Positioning </title>
```

```
</head>
```

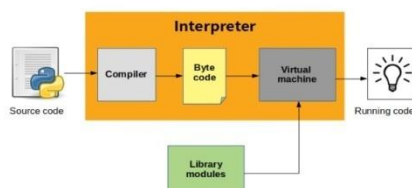
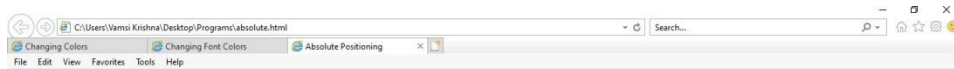
```
<body>
```

```

```

```
<body>
```

```
</html>
```



### **RELATIVE**

An element with position: relative is positioned relatively with the other elements. If the left and top properties are given values, they displace the element by the specified amount from the position where it would have been placed if the top and left properties had not set.

```
<p style = "position: relative; top: 100px;">
```

```
-----
```

```
-----
```

</p>

### **Example:**

```
<html>
<head>
<title> Absolute Positioning </title>
</head>
<body>
<pre>
```

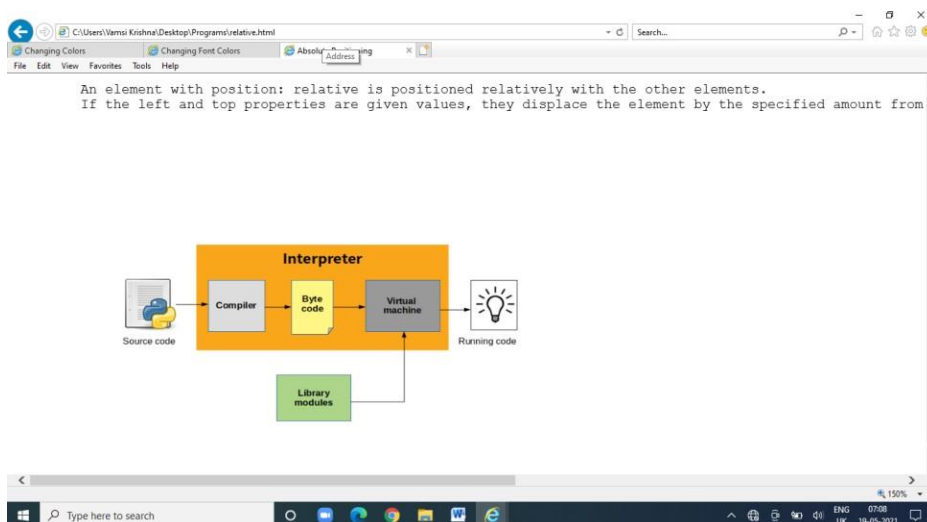
An element with position: relative is positioned relatively with the other elements.

If the left and top properties are given values, they displace the element by the specified amount from the position where it would have been placed if the top and left properties had not set.

```
</pre>

<body>
</html>
```

### **Outcome:**



## **STATIC**

The default value for the position property is static. A statically positioned element is placed in the document as if it had the position of relative but no values for top and left properties is given. A statically positioned element has not left and top properties and it cannot be moved from its position later.

## **MOVING ELEMENTS IN DHTML**

An HTML Element whose position property is set to relative and absolute can be moved. Moving of an element is simple. It can be achieved by changing the top and left properties values of an element.

- If the position is absolute, then the element moves to the new values of top and left.
- If the position is relative, then the element moves from its original position by distances given by distances given by the new values of top and left.



**Example:**

```

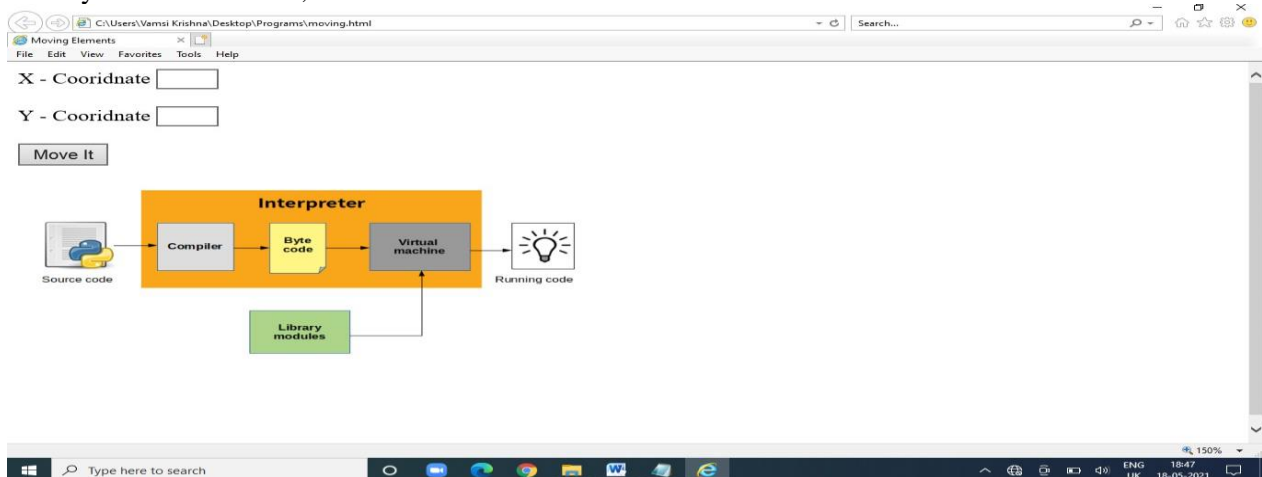
<html>
<head>
<title> Moving Elements </title>
<script type="text/javascript">
function moveIt()
{
image=document.getElementById("python").style;
x=document.myform.xcoord.value;
y=document.myform.ycoord.value;
image.top=x;
image.left=y;

}
</script>
</head>
<body>
<form name="myform">
<label> X - Coordnate</label>
<input type="text" id="xcoord" size="3"/>
<br/><br/>
<label> Y - Coordnate</label>
<input type="text" id="ycoord" size="3"/>
<br/><br/>
<input type="button" value="Move It" onclick="moveIt()" />
</form>

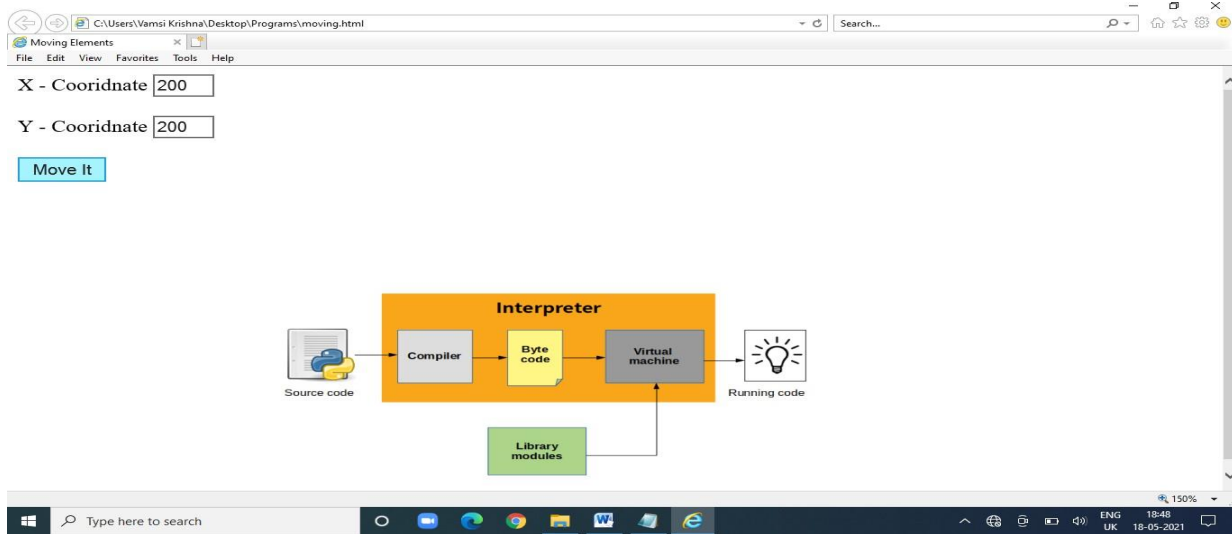
</body>
</html>

```

Initially when it is loaded, it should be as follows:



Input the desired values for the X and Y Coordinates and click **Move It** button to move the image element.



## **CHANGING THE COLORS AND FONTS**

The background and foreground colors of the document displayed can be dynamically changed and also the font properties of the text can be changed.

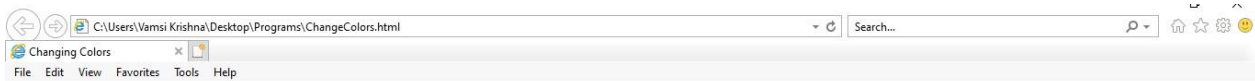
### **CHANGING COLORS**

- To change the background color of the document **backgroundColor** property value is changed.
- To change the foreground color of the document then the **color** property value is changed.

#### **Example:**

```
<html>
<head>
<title> Changing Colors </title>
<script type="text/javascript">
function changeback()
{
newColor=document.cform.background.value;
document.body.style.backgroundColor=newColor;
}
function changefore()
{
newColor=document.cform.foreground.value;
document.body.style.color=newColor;
}
</script>
</head>
<body>
<p style="fontfamily:Times; font-style:italic; font-size:e2em;">
This small page illustrated dynamic setting of the
foreground and backgorund colors for a document
</p>

<form name="cform">
<label> Background Color </label>
<input type="text" name="background" size="10" onchange="changeback()" />
<br/><br/>
<label> Foreground Color </label>
<input type="text" name="foreground" size="10" onchange="changefore()" />
</form>
</body>
</html>
```

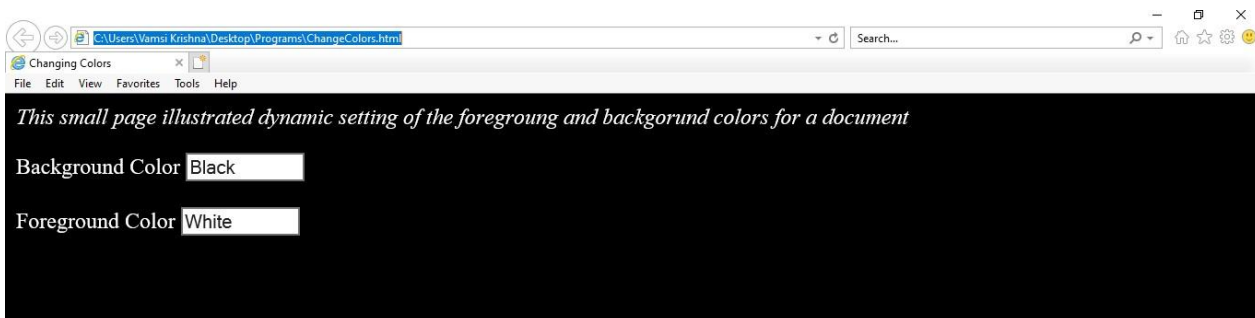
**Outcome:**

*This small page illustrated dynamic setting of the foreground and background colors for a document*

Background Color

Foreground Color

After entering desired colors for the background and foreground colors, then web page looks like as follows:



*This small page illustrated dynamic setting of the foreground and background colors for a document*

Background Color

Foreground Color

**CHANGING FONTS**

Changing the fonts in the document can be changed dynamically by making change in the font style properties – *color*, *fontStyle*, *fontSize*.

How these style properties of font values are changed in response to the user actions through mouse events – *onmouseover* and *onmouseout* is illustrated below:

```
<html> <head> <title> Changing Font Colors </title>
```

```
<style type="text/css">
```

```
.regText{ font: 1.1em 'Times New Roman'; }
```

```
.wordText{ color:blue; }
```

```
</style> </head>
```

```
<body>
```

```
<p class="regText">The
```

```
state of
```

```
<span class="wordText"; onmouseover="
```

```
this.style.color='red';
```

```
        this.style.fontStyle='italic';
```

```
        this.style.fontSize='2em'; "
```

```
onmouseout=" this.style.color='blue';
```

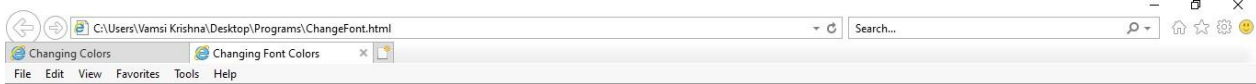
```
        this.style.fontStyle='normal';
```

```
        this.style.fontSize='1.1em'; ">
```

```
Jammu & Kashmir
```

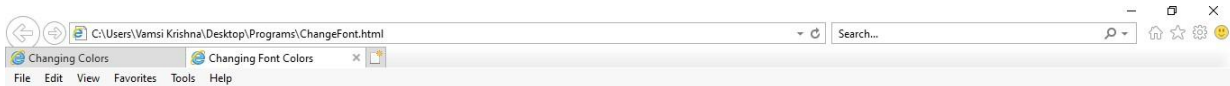
```
</span> produces many of our nation's Apples </p> </body> </html>
```

### Outcome



The state of **Jammu & Kashmir** produces many of our nation's Apples

On moving the mouse over the text “**Jammu & Kashmir**”, the font style properties of that text can be changed as follows:



The state of *Jammu & Kashmir* produces many of our nation's Apples