HTML is the standard markup language for creating Web pages.

- *HTML stands for Hyper Text Markup Language*
- *HTML describes the structure of Web pages using markup*
- *HTML elements are the building blocks of HTML pages*
- *HTML elements are represented by tags*
- *HTML tags label pieces of content such as "heading", "paragraph", "table", and so on*
- *Browsers do not display the HTML tags, but use them to render the content of the page*

HTML Tags

HTML tags are element names surrounded by angle brackets:

<tagname>content goes here...</tagname>

- HTML tags normally come **in pairs** like and
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- The end tag is written like the start tag, but with a **forward slash** inserted before the tag name

Headings:

In XHTML, there are six levels of headings, specified by the tags <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>, where <h1> specifies the highest-level heading. Headings are usually displayed in a boldface font whose default size depends on the number in the heading tag.

```
<html>
<head>
   <title> Headings </title>
</head>
<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>
```


Paragraphs:

The < p > element offers another way to structure your text. Each paragraph of text should go in between an opening < p > and closing < /p > tag, Multiple paragraphs may appear in a single document.

```
<html>
<head>
```

```
   <title> Paragraph </title>
 </head>
 <body>
  <p> Paragraph 1 </p>
  <p> Paragraph 2 </p>
  <p> Paragraph 3 </p>
 </body>
</html>
```

Creating Line Breaks Using the < br / >Element

Whenever we use the < br / > element, anything following it starts on the next line. The < br / > element is an example of an empty element ; you don ' t need opening and closing tags, because there is nothing to go in between them.

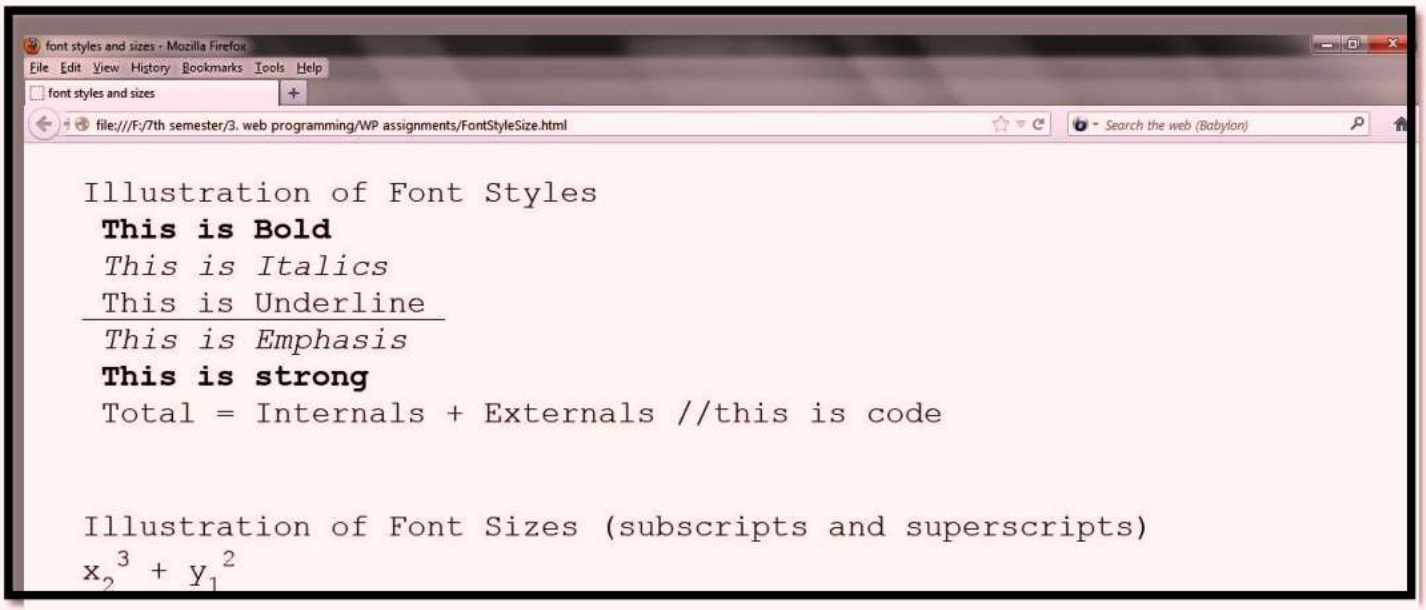
Creating Preformatted Text Using the < pre >Element

Any text between the opening < pre > tag and the closing < /pre > tag will preserve the formatting of the source document.The most common uses of the < pre > element are to represent computer source code. For example, the following shows some JavaScript inside a < pre > element

Font Styles and Sizes:

- <b>, <i> and <u> specifies bold, italics and underline respectively.
- The emphasis tag, , specifies that its textual content is special and should be displayed in some way that indicates this distinctiveness. Most browsers use italics for such content.
- The strong tag, is like the emphasis tag, but more so. Browsers often set the content of strong elements in bold.
- The code tag, <code>, is used to specify a monospace font, usually for program code.

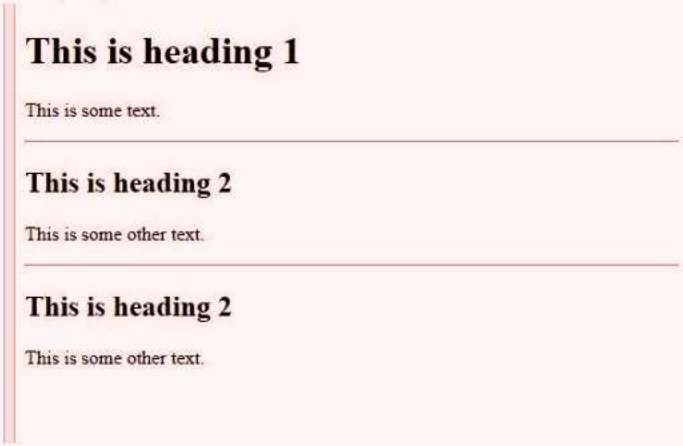
```
<html>
<head> <title> font styles and sizes </title>
</head>
<body>
 <p><pre>
  Illustration of Font Styles
  <b> This is Bold </b>
  <i> This is Italics </i>
  <u> This is Underline </u>
  <em> This is Emphasis </em>
  <strong> This is strong </strong>
  <code> Total = Internals + Externals //this is code</code>
 </pre></p>
 <p><pre>
  Illustration of Font Sizes (subscripts and
  superscripts) x<sub>2</sub><sup>3</sup> +
  y<sub>1</sub><sup>2</sup>
 </pre></p>
</body>
</html>
```



Horizontal Rules:

The parts of a document can be separated from each other, making the document easier to read, by placing horizontal lines between them. Such lines are called horizontal rules. The block tag that creates them is <hr />. The <hr /> tag causes a line break (ending the current line) and places a line across the screen.
Note again the slash in the <hr /> tag, indicating that this tag has no content and no closing tag.
Example:-

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

<h1>This is heading 1</h1>
<p>This is some text.</p>
<hr>

<h2>This is heading 2</h2>
<p>This is some other text.</p>
<hr>

<h2>This is heading 2</h2>
<p>This is some other text.</p>

</body>
</html>
```



Quotation and Citation Elements

Block Quotations:

When we want to quote a passage from another source, you should use the < blockquote > element

<q> for Short Quotations

The HTML <q> element defines a short quotation.Browsers usually insert quotation marks around the <q> element

<abbr> for Abbreviations

The HTML <abbr> element defines an abbreviation or an acronym.Marking abbreviations can give useful information to browsers, translation systems and search-engines.

<address> for Contact Information

The HTML <address> element defines contact information (author/owner) of a document or an article.The <address> element is usually displayed in italic. Most browsers will add a line break before and after the element.

<cite> for Work Title

The HTML <cite> element defines the title of a work.Browsers usually display <cite> elements in italic.

```
<!DOCTYPE html>
<html>
<body>
<p>WWF's goal is to: <q>Build a future where people live in harmony with
nature.</q></p>

<blockquote cite="http://www.worldwildlife.org/who/index.html">
For 50 years, WWF has been protecting the future of nature.
The world's leading conservation organization,
WWF works in 100 countries.y.
</blockquote>

<p>The <abbr title="World Health Organization">WHO</abbr> was founded in
1948.</p>

<p>Marking up abbreviations can give useful information to browsers,
translation systems and search-engines.</p>

<address>
Written by John Doe.<br>
Box 564, Disneyland<br>
USA
</address>
</body>
</html>
```

WWF's goal is to: "Build a future where people live in harmony with nature."

> For 50 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries.y.

The WHO was founded in 1948.

Marking up abbreviations can give useful information to browsers, translation systems and search-engines.

Written by John Doe.
Box 564, Disneyland
USA

1.3 LISTS

Three types of lists in XHTML:

- Unordered lists ,
- Ordered lists
- Definition lists

Unordered Lists:

The <ul> tag, which is a block tag, creates an unordered list. Each item in a list is specified with an <li> tag (li is an acronym for *l*ist *i*tem). Any tags can appear in a list item, including nested lists. When displayed, each list item is implicitly preceded by a bullet.

Example:-

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

<h2>An unordered HTML list</h2>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>

</body>
</html>
```

An unordered HTML list

- Coffee
- Tea
- Milk

Unordered HTML List - Choose List Item Marker

The CSS list-style-type property is used to define the style of the list item marker:

Value	Description
Disc	Sets the list item marker to a bullet (default)
Circle	Sets the list item marker to a circle
Square	Sets the list item marker to a square
None	The list items will not be marked

Ordered Lists:

Ordered lists are lists in which the order of items is important. This orderedness of a list is shown in the display of the list by the implicit attachment of a sequential value to the beginning of each item. The default sequential values are Arabic numerals, beginning with 1.
An ordered list is created within the block tag <ol>. The items are specified and displayed just as are those in unordered lists, except that the items in an ordered list are preceded by sequential values instead of bullets.

Example:-

```
<html>
<head>
<title> ordered List </title>
</head>
<body>
<h1>Chicken Masala</h1>
<ol>
<li>For 1 kg of chicken, add 20g Teju Chicken Masala</li>
<li>Fry 2 big onions with 3tbsp ghee/oil till golden brown</li>
<li>Add 2 tomato, 1tsp ginger garlic paste, 2-3 green chillies and fry</li>
<li>When tomato becomes soft, add chicken and 100ml water</li>
<li>Add 25g coriander leaves and cook till the chicken is soft and gravy turns
thick</li>
<li>Ready to serve</li>
</ol>
</html>
</head>
```

Chicken Masala

1. For 1 kg of chicken, add 20g Teju Chicken Masala
2. Fry 2 big onions with 3tbsp ghee/oil till golden brown
3. Add 2 tomato, 1tsp ginger garlic paste, 2-3 green chillies and fry
4. When tomato becomes soft, add chicken and 100ml water
5. Add 25g coriander leaves and cook till the chicken is soft and gravy turns th
6. Ready to serve

Ordered HTML List -Type Attribute

The type attribute of the <ol> tag, defines the type of the list item marker:

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

Nested Lists:

```
<html>
 <head>
  <title> nested lists </title>
 </head>
 <ol>
  <li> Information Science </li>
   <ol>
        <li>OOMD</li>
        <li>Java & J2ee</li>
        <ul>
          <li>classes and methods</li>
          <li>exceptions</li>
          <li>applets</li>
          <li>servelets</li>
        </ul>
        <li>Computer Networks</li>
        <ul>
```

```
            <li>Part 1</li>
            <li>Part 2</li>
          </ul>
          <li>DBMS</li>
          <li>Operations Research</li>
      </ol>
    <li> Computer Science</li>
      <ol>
          <li>Compiler Design</li>
          <li>FLAT</li>
          <ul>
            <li>NFA</li>
            <li>DFA</li>
            <li>CFG</li>
          </ul>
          <li>Computer Graphics</li>
          <li>Artificial Intelligence</li>
      </ol>
    </ol>
</html>
 html>
```

Block and Inline Elements

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).The element is a block-level element.

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

<div>Hello</div>
<div>World</div>

<p>The DIV element is a block element, and will start on a new line.</p>

</body>
</html>
```

Hello
World

The DIV element is a block element, and will start on a new line.

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline element inside a paragraph.

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

<span>Hello</span>
<span>World</span>

<p>The SPAN element is an inline element, and will not start on a new line.</p>

</body>
</html>
```

Hello World

The SPAN element is an inline element, and will not start on a new line.

Creating Links with the Element

Links:

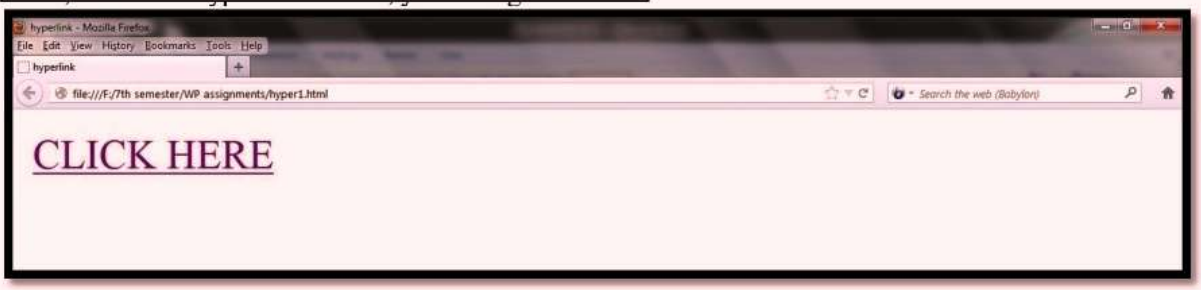
- Hyperlinks are the mechanism which allows the navigation from one page to another.
- The term "hyper" means beyond and "link" means connection
- Whichever text helps in navigation is called hypertext
- Hyperlinks cam be created using **<a>** (anchor tag)
- The attribute that should be used for <a> is **href**

Program: *hyper.html*

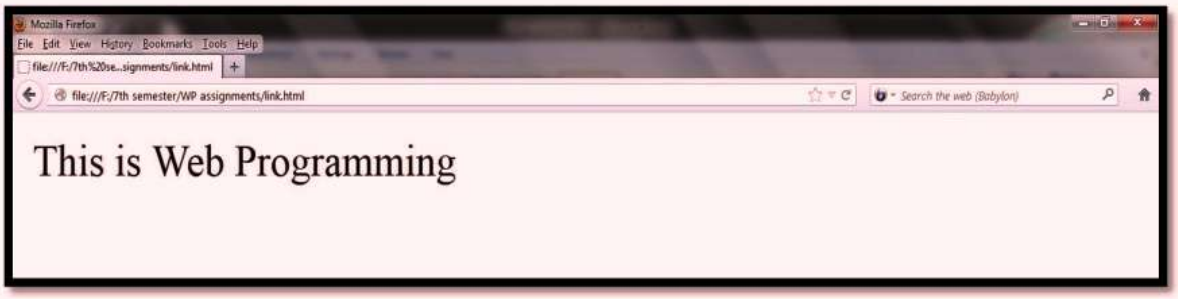
```
<html>
 <head>
  <title> hyperlink </title>
 </head>
 <a href = "link.html"> CLICK HERE </a>
</html>
```

Program: *link*

```
<html>
 <body> This is Web Programming </body>
</html>
```



After clicking on the above text, we can navigate to another page "link.html" as shown below



Images, Audio, and Video, Using Images as Links, Choosing the Right Image Format

IMAGES

- Image can be displayed on the web page using **<img>** tag.
- When the <img> tag is used, it should also be mentioned which image needs to be displayed. This is done using **src** attribute.
- Attribute means extra information given to the browser
- Whenever <img> tag is used, alt attribute is also used.
- Alt stands for alert.
- Some very old browsers would not be having the capacity to display the images.
- In this case, whatever is the message given to alt attribute, that would be displayed.
- Another use of alt is → when image display option has been disabled by user. The option is normally disabled when the size of the image is huge and takes time for downloading.

```
<html>
  <head>
    <title>display image</title>
  </head>
  <body>
    <img src="java.png" alt="cannot display"/>
  </body>
</html>
```

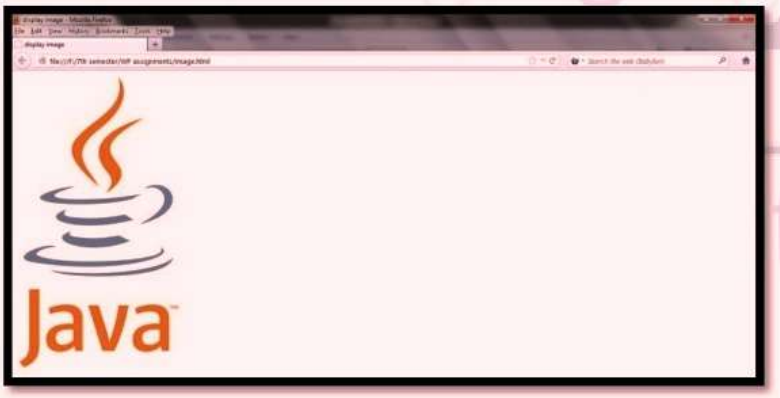


Image Size - Width and Height

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

```
<html>
<body>

<h2>Image Size</h2>

<p>Use the style attribute to specify the width and height of an image:</p>
<img src="img_girl.jpg" alt="Girl in a jacket"
style="width:500px;height:600px;">

</body>
</html>
```

Image Size

Use the style attribute to specify the width and height of an image:

The ismap and usemap Attributes

The ismap and usemap attributes are used with image maps.

The ismap attribute is a boolean attribute.

When present, it specifies that the image is part of a server-side image-map (an image-map is an image with clickable areas). When clicking on a server-side image-map, the click coordinates are sent to the server as a URL query string.

Note: *The ismap attribute is allowed only if the <img> element is a descendant of an <a> element with a valid href attribute.*

Example:-

```
<!DOCTYPE html>
<html>
<body>
<a href="/action_page.php">
  <img src="w3html.gif" alt="W3Schools.com" width="100" height="132" ismap>
</a>
<p>Click the image, and the click coordinates will be sent to the server as a URL query string.</p>
</body>
</html>
```

Output:-

Click the image, and the click coordinates will be sent to the server as a URL query string.

Submitted Form Data

Your input was received as:

66,105

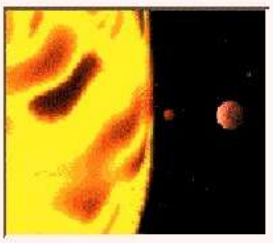
The server has processed your input and returned this answer.

The usemap attribute *specifies an image as a client-side image-map (an image-map is an image with clickable areas). The usemap attribute is associated with a <map> element's name attribute, and creates a relationship between the <img> and the <map>.*
Note: *The usemap attribute cannot be used if the <img> element is a descendant of an <a> or <button> element.*

Example:-

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

<p>Click on the sun or on one of the planets to watch it closer:</p>
<img src="planets.gif" width="145" height="126" alt="Planets" usemap="#planetmap">
<map name="planetmap">
 <area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm">
 <area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm">
 <area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm">
</map>

</body>
</html>
```

Click on the sun or on one of the planets to watch it closer:



TABLES

A table is a matrix of cells. The cells in the top row often contain column labels, those in the leftmost column often contain row labels, and most of the rest of the cells contain the data of the table. The content of a cell can be almost any document element, including text, a heading, a horizontal rule, an image, and a nested table.

Basic Table Tags:

- A table is specified as the content of the block tag **<table>.**
- There are two kinds of lines in tables: the line around the outside of the whole table is called the *border*; the lines that separate the cells from each other are called *rules*.
- It can be obtained using **border** attribute. The possible values are "border" or any number.
- The table heading can be created using **<caption>** tag.
- The table row can be created using **<tr>** tag.
- The column can be created either by using **<th>** tag (stands for table header which is suitable for headings) or **<td>** tag (stands for table data which is suitable for other data).

```
<html>
 <head>
  <title> Table with text and image </title>
 </head>
 <body>
   <table border = "border">
   <caption>PARAMATHMA Movie Details </caption>
     <tr>
         <th> Cast</th>
         <th> Image </th>
      </tr>
     <tr>
        <td> Puneeth Rajkumar </td>
        <td> <img src = "puneeth.jpg" alt = "cant display"/></td>
      </tr>
     <tr>
        <td> Deepa Sannidhi</td>
        <td> <img src = "deepa.jpg" alt = "cant display"/></td>
      </tr>
    </table>
  </body>
</html>
```

The rowspan and colspan Attributes:

Multiple-level labels can be specified with the rowspan and colspan attributes.

```
<html>
 <head>
  <title>row-span and column-span</title>
 </head>
 <body>
  <p> Illustration of Row span</p>
  <table border="border">
   <tr>
    <th rowspan="2"> RNSIT </th>
    <th>ISE</th>
   </tr>
   <tr>
     <th>CSE</th>
   </tr>
  </table>
  <p> Illustration of Column span</p>
  <table border="border">
   <tr>
    <th colspan="2"> RNSIT </th>
   </tr>
   <tr>
     <th>ISE</th>
```

Illustration of Row span

RNSIT	ISE
	CSE

Illustration of Column span

RNSIT	
ISE	CSE

```
      <th>CSE</th>
     </tr>
   </table>
  </body>
</html>
```

The align and valign Attributes:

The placement of the content within a table cell can be specified with the align and valign attributes in the <tr>, <th>, and <td> tags.

The align attribute has the possible values left, right, and center, with the obvious meanings for horizontal placement of the content within a cell. The default alignment for th cells is center; for td cells, it is left.

The valign attribute of the <th> and <td> tags has the possible values top and bottom. The default vertical alignment for both headings and data is center.

```
<html>
 <head>
  <title> Align and valign </title>
 </head>
 <body>
  <p>Table having entries with different alignments</p>
   <table border="border">
    <tr align = "center">
      <th> </th>
      <th> Puneeth Rajkumar </th>
      <th> Darshan Thoogudeep</th>
      <th> Kichcha Sudeep </th>

    </tr>
    <tr>
      <th> Ramya </th>
      <td align = "left"> Akaash </td>
      <td align = "center"> Datta </td>
      <td align = "right"> Ranga </td>
    </tr>
    <tr>
       <th> <br/>Rakshitha <br/><br/><br/></th>
        <td> Appu </td>
        <td valign = "top"> Kalasipalya </td>
        <td valign = "bottom"> Kaashi from village </td>
     </tr>
    </table>
 </body>
</html>
```

The cellpadding and cellspacing Attributes:

Cellspacing is the distance between cells.
Cellpadding is the distance between the edges of the cell to its content.

```
<html>
 <head>
  <title> cell spacing and cell padding </title>
 </head>
 <body>
  <h3>Table with space = 10, pad = 50</h3>
  <table border = "7" cellspacing = "10" cellpadding = "50">
    <tr>
     <td> Divya </td>
     <td>Chethan </td>
    </tr>
  </table>
  <h3>Table with space = 50, pad = 10</h3>
   <table border = "7" cellspacing = "50" cellpadding = "10">
    <tr>
     <td> Divya </td>
     <td>Chethan </td>
    </tr>
   </table>
 </body>
</html>
```

form. XHTML provides tags to generate the commonly used objects on a screen 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.

The <form> Tag:

All of the controls of a form appear in the content of a <form> tag. A block tag, <form>, can have several different attributes, only one of which, action, is required. The action attribute specifies the URL of the application on the Web server that is to be called when the user clicks the *Submit* button. Our examples of form elements will not have corresponding application programs, so the value of their action attributes will be the empty string ("").

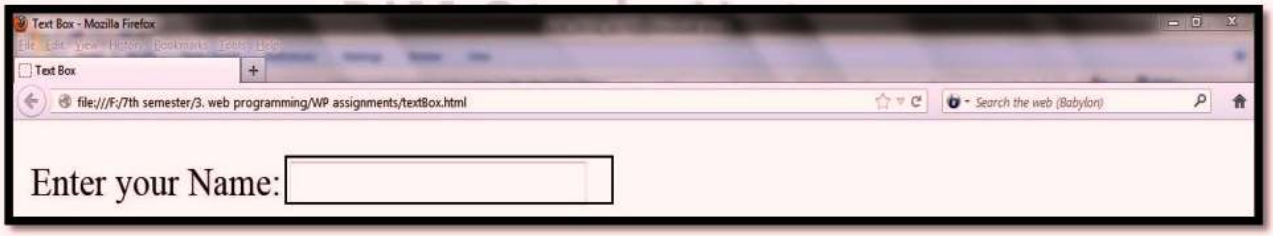
The <input> Tag:

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

❖ **Text Box**

- ✓ It is a type of input which takes the text.
- ✓ Any type of input can be created using <input>
- ✓ The *type* attribute indicates what type of input is needed for the text box, the value should be given as text.
- ✓ For any type of input, a name has to be provided which is done using *name* attribute.
- ✓ The size of the text can be controlled using *size* attribute.
- ✓ Every browser has a limit on the number of characters it can collect. If this limit is exceeded, the extra characters are chopped off. To prevent this chopping, *maxlength* attribute can be used. When maxlength is used, users can enter only those many characters that is given as a value to the attribute.

```
<html>
  <body>
  <title>Text Box</title>
 </head>
  <body>
  <form action = " ">
   <p>
    <label>Enter your Name:
     <input type = "text" name = "myname" size = "20" maxlength = "20" />
    </label>
   </p>
  </form>
 </body>
</html>
```



❖ **Password Box**

- ✓ If the contents of a text box should not be displayed when they are entered by the user, a password control can be used.
- ✓ In this case, regardless of what characters are typed into the password control, only bullets or asterisks are displayed by the browser.

```
<html>
 <head>
  <title>Password Box</title>
 </head>
  <body>
  <form action = " ">
   <p>
    <label>Enter the email id:
     <input type = "text" name = "myname" size = "24" maxlength = "25" />
```

```
    </label>
    </p>
    <p>
    <label>Enter the password:
     <input type = "password" name = "mypass" size = "20" maxlength = "20" />
    </label>
   </p>
  </form>
 </body>
</html>
```

❖ **Radio Button**

- ✓ Radio buttons are special type of buttons which allows the user to select only individual option
- ✓ Radio buttons are created using the input tag with the *type* attribute having the value **radio.**
- ✓ When radio buttons are created, values must be provided with the help of *value* attribute.
- ✓ All the radio buttons which are created would have same name. This is because the radio buttons are group elements.
- ✓ If one of the radio buttons has to be selected as soon as the web page is loaded, checked attribute should be used. The value also would be checked.

```
<html>
<head>
<title>Radio Button</title>
</head>
<body>
<h3>Who is your Favourite Actor?</h3>
<form action = " ">
<p>

<label><input type="radio" name="act" value="two"/>Chris Evans </label>

<label><input type="radio" name="act" value="three"/>Tom Cruise</label>

<label><input type="radio" name="act" value="four"/> Robert Downey Jr.</label>
</p>
</form>
</body>
</html>
```

Who is your Favourite Actor?

○ Chris Evans ○ Tom Cruise ○ Robert Downey Jr.

- **Check Box**
  - ✓ Check box is a type of input using which multiple options can be selected.
  - ✓ Check box can also be created using the <input> tag with the *type* having the value "checkbox".
  - ✓ During the creation of check box, the value should be provided using the *value* attribute.
  - ✓ All the checkbox which are created would have the same name because they are group elements.
  - ✓ If one of the check box have to be selected as soon as the page is loaded, checked attribute should be used with the value checked.

```
<html>
<head>
<title>Check Box</title>
</head>
<body>
<h3>Who is your Favourite Actress?</h3>
<form action = " ">
<p>
<label><input type="checkbox" name="act" value="one"/> Emma Stone</label>

<label><input type="checkbox" name="act" value="two"/>Gal Gadot</label>

<label><input type="checkbox" name="act" value="three"/>Brie Larson</label>

<label><input type="checkbox" name="act" value="four"/>Ana de Armas </label>


</p>
</form>
</body>
</html>
```

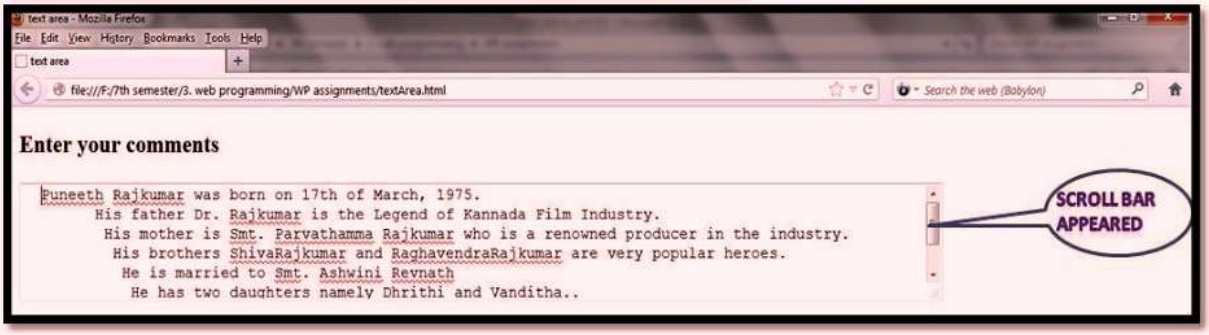
Who is your Favourite Actress?

☐ Emma Stone ☐ Gal Gadot ☐ Brie Larson ☐ Ana de Armas

The <textarea> Tag:

- Text area is a type of input using which multiple statements can be entered.
- Text area is created using <textarea> tag.
- Text area should have the name.
- During the creation of text area, it should be mentioned how many sentences can be entered. This is done using *rows* attribute.
- Similarly, it should also be mentioned how many characters can be entered in a line. This is done using *cols* attribute.
- If the value given to rows is exceeded i.e. if users enter sentences more than specified, the *scroll bar* automatically appears.

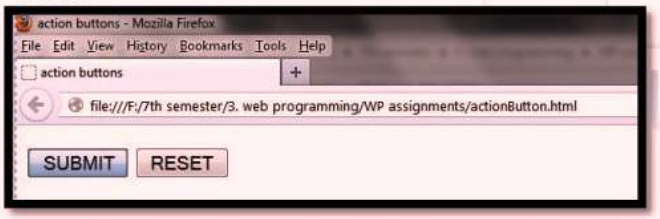
```
<html>
 <head>
  <title> text area </title>
 </head>
 <body>
 <form action=" ">
   <h3> Enter your comments</h3>
   <p>
    <textarea name="feedback" rows="5" cols="100">
    </textarea>
    </p>
 </form>
 </body>
</html>
```



The Action Buttons:

The *Reset* button clears all of the controls in the form to their initial states. The *Submit* button has two actions: First, the form data is encoded and sent to the server; second, the server is requested to execute the server-resident program specified in the action attribute of the <form> tag. The purpose of such a server-resident program is to process the form data and return some response to the user. Every form requires a *Submit* button. The *Submit* and *Reset* buttons are created with the <input> tag.

```
<html>
 <head>
  <title> action buttons </title>
 </head>
 <body>
 <form action=" ">
  <p>
     <input type="SUBMIT" value="SUBMIT"/>
     <input type="RESET" value="RESET"/>
   </p>
 </form>
 </body>
</html>
```



NOTE: A *plain* button has the type button. *Plain* buttons are used to choose an action.

Example of a Complete Form:

```
<html>
 <head>
  <title> CompleteForm</title>
 </head>
 <body>
  <h1>Registration Form</h1>
  <form action=" ">
   <p>
     <label>Enter your email id:
       <input type = "text" name = "myname" size = "24" maxlength = "25" />
     </label>
    </p>
    <p>
     <label>Enter the password:
```

```
      <input type = "password" name = "mypass" size = "20" maxlength = "20" />
     </label>
    </p>
   <p>Sex</p>
   <p>
    <label><input type="radio" name="act" value="one"/>Male</label>
     <label><input type="radio" name="act" value="two"/>Female</label>
     </p>
    <p>Which of the following Accounts do you have?</p>
   <p>
    <label><input type="checkbox" name="act" value="one"/>Gmail</label>

    <label><input type="checkbox" name="act" value="two"/>Facebook</label>
    <label><input type="checkbox" name="act" value="three"/>Twitter</label>
    <label><input type="checkbox" name="act" value="four"/>Google+</label>
   </p>
  <p> Any Suggestions?</p>
   <p>
    <textarea name="feedback" rows="5" cols="100">
    </textarea>
   </p>
   <p>Click on Submit if you want to register</p>
   <p>
     <input type="SUBMIT" value="SUBMIT"/>
     <input type="RESET" value="RESET"/>
   </p>
    </form>
  </body>
</html>
```

Registration Form

Enter your email id: 1rn09is016@gmail.com

Enter the password: ••••••••••••••

Sex

Male Female

Which of the following Accounts do you have?

Gmail Facebook Twitter Google+

Any Suggestions?

PLEASE PROVIDE A FASTER AND BETTER SERVICE..!!

Click on Submit if you want to register

SUBMIT RESET

What is CSS?

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

Types of CSS Styles

There are three types of CSS styles:

- **Inline styles**

Inline styles are styles that are written directly in the tag on the document. Inline styles affect only the tag they are applied to.

```
<html>
<body>

<h1 style="color:blue;margin-left:30px;">This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```


- **Embedded or Internal styles**

Embedded styles are styles that are embedded in the head of the document. Embedded styles affect only the tags on the page they are embedded in.

```
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head
```

```
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

- **External styles**

External styles are styles that are written in a separate document and then attached to various Web documents. External style sheets can affect any document they are attached to.

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

Chapter -2.2 Font formatting

font – family :- Specifies the typeface or family of font

font – weight :- Specifies whether the font should be normal ,bold ,bolder (0 to 500 number)

font – stretch :- sets the width of the characters in a font

(condensed /ultra-condensed /extra-condensed semi-condensed semi-expanded expanded extra-expanded ultra-expanded)

Syntax

font-family : font name (Calibri/moonscape/Cambria)
font-size : length/percentage
font-weight: normal/bold/bolder/lighter/100
font-style: normal/italic/oblique
font-variant: normal|small-caps
font-stretch: condensed

```
p {
font-family: Cambria;
font-size: 20px;
}
```

font – variant

```
p.normal {
font-variant: normal;
}
```

```
p.small {
font-variant: small-caps;
}

<p class="normal">My name is HegeRefsnes.</p>
<p class="small">My name is HegeRefsnes.</p>
```

My name is HegeRefsnes.

MYNAME IS HEGE REFSNES.

Chapter 2.3:- Text Formatting

text – align	Specifies the horizontal alignment of the text
vertical – align	Specifies the vertical alignment of text within containing element
text – decoration	specifies the decoration added to text.
text – indent	Specifies an indent from the left border for the text
text – shadow	adds shadow to text
text – transform	controls the capitalization of text
letter – spacing	Controls the width between letters
word – spacing	Controls the amount of space between each word
white – space	specifies how white-space inside an element is handled.

Syntax

color:	#ff0000; (color name/hex code)
direction:	ltr|rt
text-align:	left|right|center|justify
vertical-align:	top/middle/bottom/text-bottom
text-shadow:	2px 2px #ff0000; }
text-indent:	*length/percentage*
text-decoration:	underline|overline|line-through
text-transform:	capitalize|uppercase|lowercase
letter-spacing:	*length*(2px,-3px)
word-spacing:	*length*(30px,-5px)
white-space:	nowrap/normal/pre;

vertical-align

```
img.top {
vertical-align: top;
}

img.bottom {
vertical-align: middle;
}
```


```
<p>An <img class="top" src="w3schools_logo.gif" alt="W3Schools" width="270" height="50" /> image with a text-top alignment.</p>

<p>An <img class="bottom" src="w3schools_logo.gif" alt="W3Schools" width="270" height="50" /> image with a text-bottom alignment.</p>
```

Chapter 2.4 CSS Selectors

Selector to specify which elements a style sheet rule

Types:-

- universal *
- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

2.4.1 Universal Selector

The universal selector is an asterisk; it is like a wildcard and matches all element types in the document.

2.4.2 The Type Selector

The type selector matches all of the elements specified in the comma - delimited list. It allows us to apply the same rules to several elements.

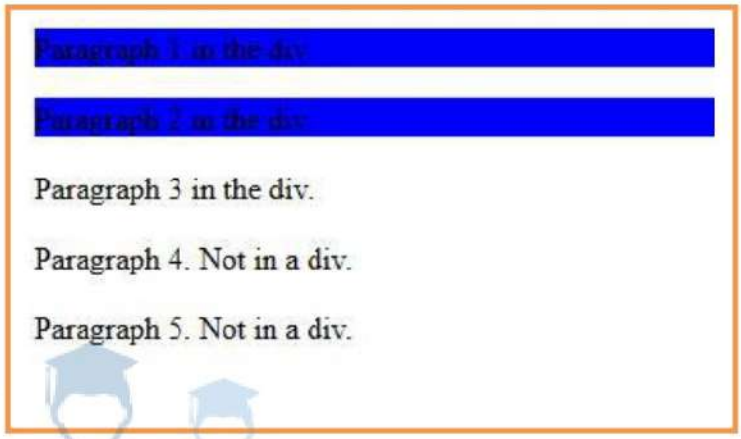
```
h1, h2, h3 {}
```

2.4.5 **Child Selector**

The child selector selects all elements that are the immediate children of a specified element.

The following example selects all <p> elements that are immediate children of a element:

```
<html>
<head>
<style>
div> p {
background-color: blue;
}
</style>
</head>

<body>
<div>
<p>Paragraph 1 in the div.</p>
<p>Paragraph 2 in the div.</p>
<span><p>Paragraph 3 in the div.</p></span><!-- not Child but Descendant -->
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```



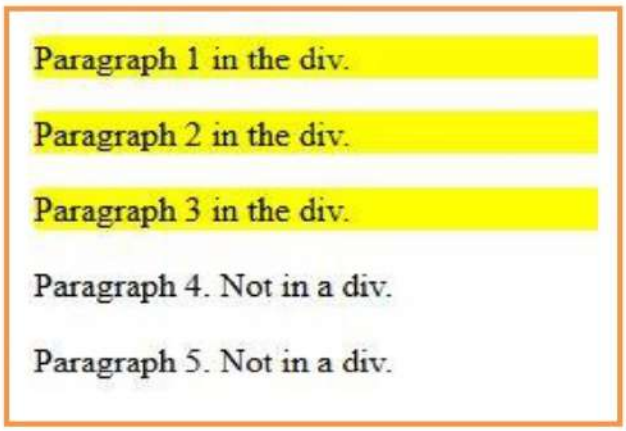
2.4.6 **Descendant Selector**

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all <p> elements inside elements:

```
<html>
<head>
```

```
<style>
div p {
background-color: yellow;
}
</style>
</head>

<body>
<div>
<p>Paragraph 1 in the div.</p>
<p>Paragraph 2 in the div.</p>
<span><p>Paragraph 3 in the div.</p></span>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```



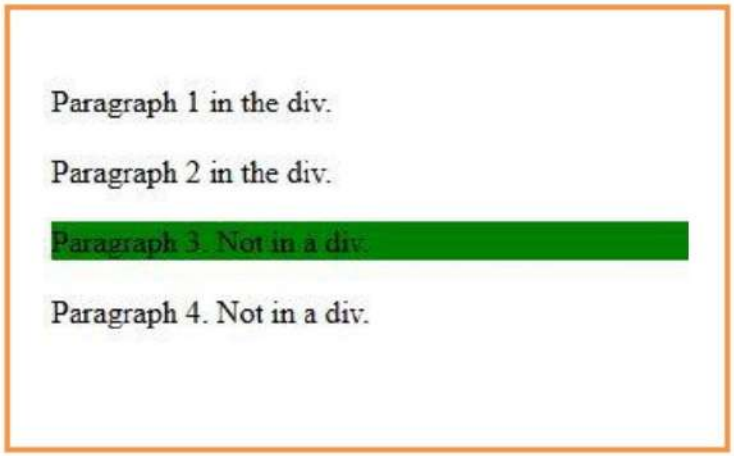
2.4.7 Adjacent Sibling Selector

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects all <p> elements that are placed immediately after elements:

```
<style>
div + p {
background-color: green;
}
</style>
</head>
<body>

<div>
```



```
<p>Paragraph 1 in the div.</p>
<p>Paragraph 2 in the div.</p>
</div>
<p>Paragraph 3. Not in a div.</p>
<p>Paragraph 4. Not in a div.</p>
```

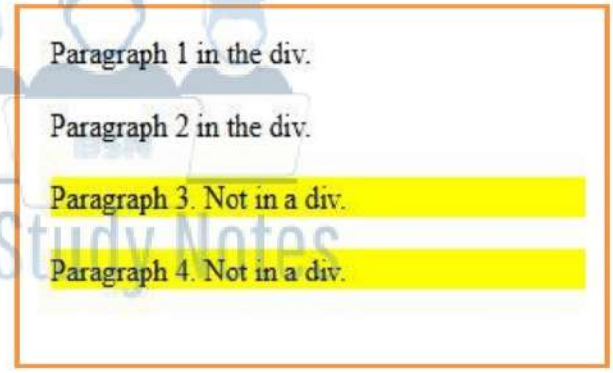
Chapter -5General Sibling Selector

2.5.1 Using Child and Sibling Selectors To Reduce Dependence on Classes in Markup

The general sibling selector selects all elements that are siblings of a specified element.

The following example selects all <p> elements that are siblings of elements:

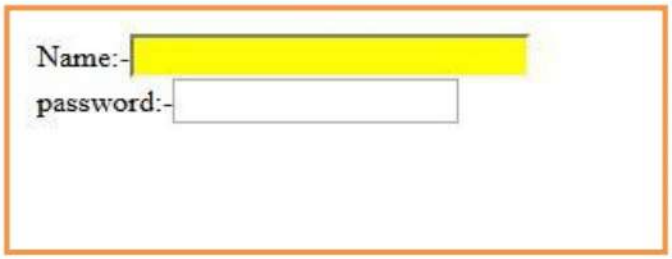
```
<html>
<head>
<style>
div ~ p {
background-color: yellow;
}
</style>
</head>

<body>
<div>
<p>Paragraph 1 in the div.</p>
<p>Paragraph 2 in the div.</p>
</div>
<p>Paragraph 3. Not in a div.</p>
<p>Paragraph 4. Not in a div.</p>

</body>
</html>
```



2.5.2Attribute Selector

used to select elements with a specified attribute.

```
<html>
<head>
<style>
input[type="text"]{
background-color: yellow;
width: 200px;
}
</style>

</head>
<body>
Name:-<input type="text"><br/>
password:-<input type="password"><br/>

</body>
</html>
```



Chapter 2-9 list

2.9.1 CSS list-style-type Property

specifies the type of list-item marker in a list.<html>

Syntax

list-style-type: *value;* (disc/circle/square)

```
<head>
<style>
ul.a {list-style-type: circle;}
ol.B {list-style-type: upper-roman;}
```

```
</style>
</head>

<body>
<p>Example of unordered lists:</p>
<ul class="a">
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>
</ul>

<p>Example of ordered lists:</p>
<ol class="b">
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>
</ol>

</body>
</html>
```

Example of unordered lists:

- Coffee
- Tea
- Coca Cola

Example of ordered lists:

I. Coffee
II. Tea
III. Coca Cola

2.9.3 CSS list-style-image Property

Specify an image as the list-item marker in a list

Syntax

list-style-image: *url*

```
<html>
<head>
<style>
ul {
list-style-image: url('sqpurple.gif');
}
</style>
```

- Coffee
- Tea
- Coca Cola

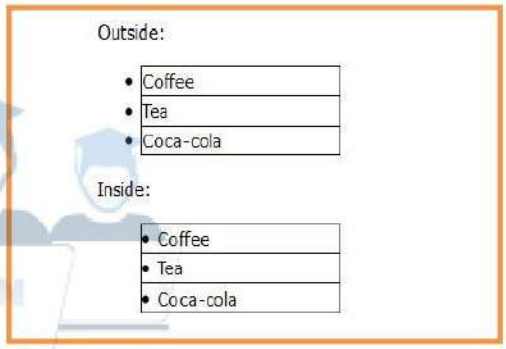
```
</head>
<body>
<ul>
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>
</ul>
</body>
</html>
```

2.9.2 CSS list-style-position Property

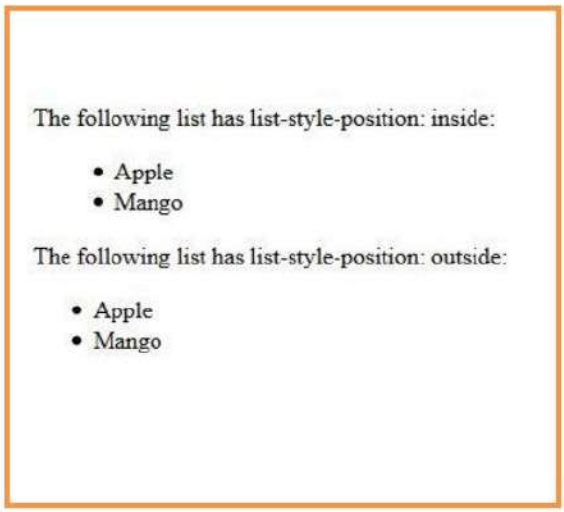
specifies if the list-item markers should appear inside or outside the content flow.

SS Syntax

```
list-style-position: inside|outside|initial|inherit;
```




```
<html>
<head>
<style>
ul.a {
list-style-position: inside;
}

ul.b {    list-style-position: outside;}
</style>
</head>

<body>
<p> list-style-position: inside:</p>
<ul class="a">
<li>Apple</li>
<li>Mango</li>
</ul>
<p> list-style-position: outside:</p>
```



```
<ul class="b">
<li>Apple</li>
<li>Mango</li>
</ul>

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

Chapter 2.6 Lengths

Lengths define as such as the size of fonts, height of lines of text, and gaps between words and letter.

three ways lengths

a) ***Relative Units***

There are three types of relative units: pixels, which relate to the resolution of the screen, and em ' s and ex ' s both of which relate to the size of fonts.

b) Absolute Units

Pt :-A point , pc:- A pica, in:-An inch ,cm:-A centimeter , mm:-A millimeter

c) Percentages(%)

Chapter 2.10 Table

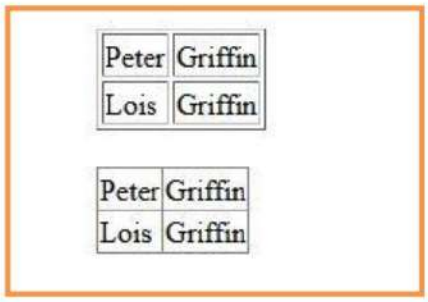
2.10.1Table - Specific Properties

1. **border - collapse**

sets whether the table borders are collapsed into a single border or detached as in standard HTML.

syntax

border-collapse: separate|collapse




```
<html>
<head>
<style>
table.ex1 {
border-collapse: separate;
```

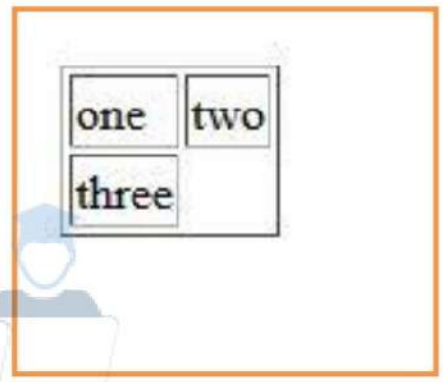
```
}
table.ex2 {
border-collapse: separate;
}
</style>
</head>
<body>
table class="ex1" border="1">
<tr>
<td>Peter</td>
<td>Griffin</td>
</tr>
<tr>
<td>Lois</td>
<td>Griffin</td>
</tr>
</table>
<br>

<table class="ex2" border="1">
<tr>
<td>Peter</td>
<td>Griffin</td>
</tr>
<tr>
<td>Lois</td>
<td>Griffin</td>
</tr>
</table>
</body>
</html>
```

2.10.3 CSS empty-cells Property

The empty-cells property sets whether or not to display borders and background on empty cells in a table (only for the "separated borders" model).

empty-cells: show|hide

```
<html>
<head>
<style>
table {
empty-cells: hide;
}
</style>
</head>

<body>
<table border="1">
<tr>
<td>one</td>
<td>two</td>
</tr>
<tr>
<td>three</td>
<td></td>
</tr>
</table>
</body>

</html>
```



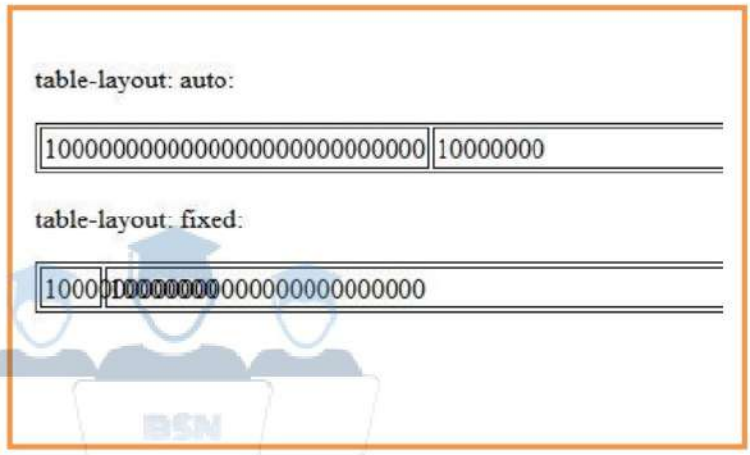
caption-side Property

The caption-side property specifies the placement of a table caption.

caption-side: top|bottom|left|right

12.10.5 CSS table-layout Property

The table – layout property allows you to force the browser to stick to the widths you specify, even if this makes the content unreadable.




```
<html>
<head>
<style>

table {
width: 100%;
}

table.ex1 {
table-layout: auto;
}

table.ex2 {
table-layout: fixed;
}</style>
</head>
<body>

<p>table-layout: auto:</p>
<table class="ex1">
<tr>
<td width="5%">10000000000000000000000000000</td>
<td width="95%">10000000</td>
</tr>
```

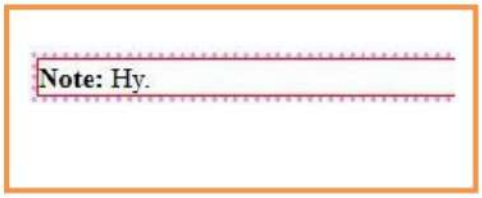
```
</table>

<p>table-layout: fixed:</p>
<table class="ex2">
<tr>
<td width="5%">10000000000000000000000000000</td>
<td width="95%">10000000</td>
</tr>
</table>

</body>
</html>
```

2.11 Outlines

Outlines are similar to the borders. An outline is a line that is drawn around elements (outside the borders) to make the element "stand out".

outline-width	specifies the width of an outline.
float	specifies whether or not a box (an element) should float
clear Property	Elements after a floating element will flow around it.To avoid this, use the clear property
outline-width:	medium|thin|thick|*length*
outline-style:	dotted|dashed|solid
outline-color:	*color name*
float:	left/right
clear:	left|right|both

```
<html>
<head>
<style>
p {
border: 1px solid red;
outline-style: dotted;
outline-color: violet;
}
</style>
</head>
<body>

<p><b>Note:</b>Hy.</p>

</body>
</html>
```



Chapter 2.12 The :focus and :active Pseudo Classes

Pseudo-classes:- A pseudo-class is used to define a special state of an element.

- Style an element when a user mouse over it
- Style visited and unvisited links differently
- Style an element when it gets focus

CSS :active Selector

syntax

```
a:active {
  background-color: yellow;
}
```

```
<html>
<head>
<style>
/* unvisited link */
a:link {    color: red;}

/* visited link */
a:visited {    color: green;}

/* mouse over link */
a:hover {    color: hotpink;}

/* selected link */
a:active {    color: blue;}
</style>
</head>
<body>

<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>

</body>
</html>
```

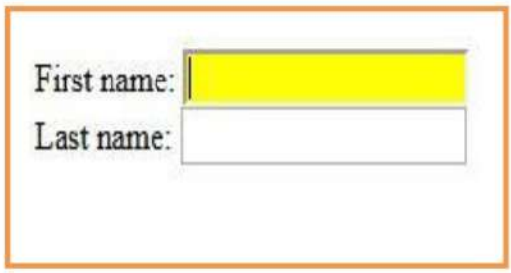

CSS :focus Selector

is used to select the element that has focus.

CSS Syntax

```
:focus {
   css declarations;
}
```

```
<html>
<head>
<style>
input:focus {
background-color: yellow;
}
</style>
</head>
<body>

<form>
First name: <input type="text" name="firstname"><br>
Last name: <input type="text" name="lastname">
</form>

</body>
</html>
```



2.12.2 content Property

```
p::before {
content: normal|counter|attr|string|open-quote|url|initial|inherit;
}
```

A string	Inserts plain text.
A URL	The URL can point to an image, text file, or HTML file to be included at this point.
A counter	A counter for numbering elements on the page (discussed in the next section).
atrr(x)	The value of an attribute named Xthat is carried on that element

open- quote	nserts the appropriate opening quote symbol
close - quote	Inserts the appropriate closing quote symbol
no - open - quote	Do not use any opening quotes.
no - close - quote	Do not use a closing quote (of particular use in prose where one person is speaking for a long while and style dictates the quote is closed only on the last paragraph).

2.12.3CSS cursor Property

The cursor property specifies the type of cursor to be displayed when pointing on an element.

cursor: crosshair/help/auto/...

```
<html>
<body>
<p>Mouse over the words to change the cursor.</p>
<span style="cursor:auto">auto</span><br>
<span style="cursor:crosshair">crosshair</span><br>
<span style="cursor:help">help</span><br>
</body>
</html>
```

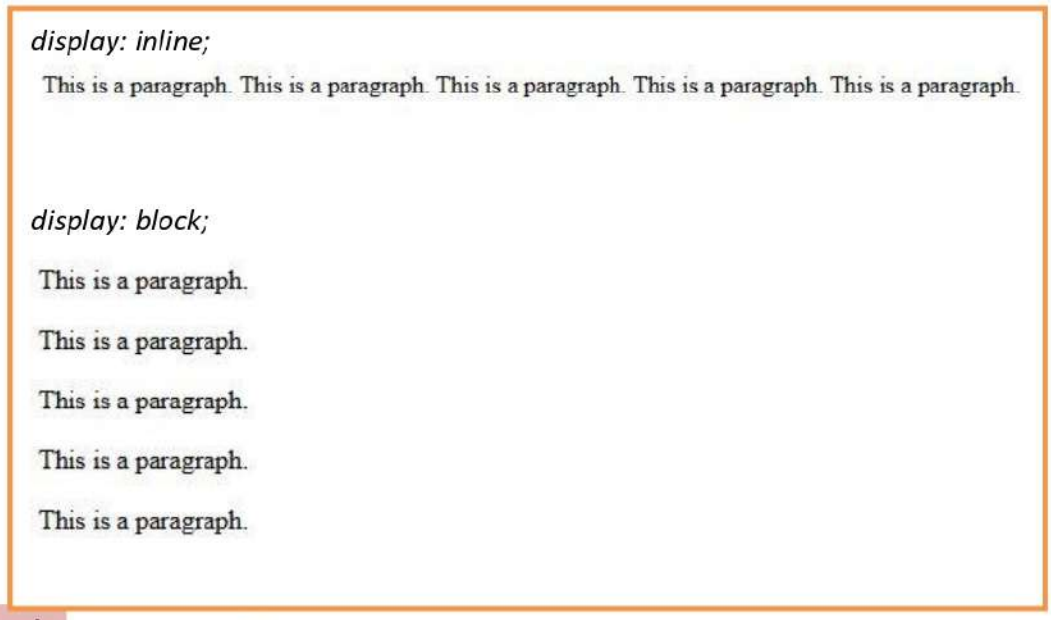

2.12.4 display Property

property specifies the type of box used for an HTML element.

display: inline/ block

```
<html>
<head>
<style>
p {
display: inline;
}
</style>
```

```
</head>
<body>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>

</body>
</html>
```

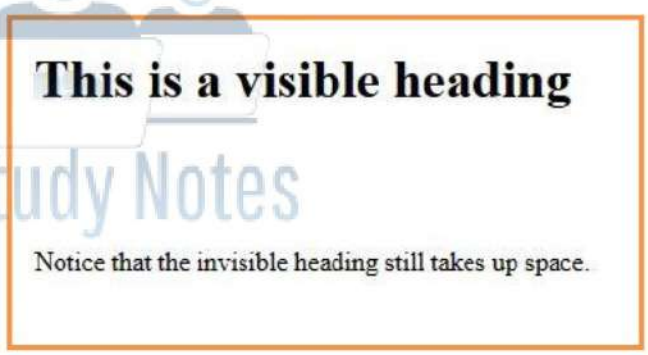


2.12.5 CSS visibility Property

specifies whether or not an element is visible.

visibility: visible|hidden

```
<html>
<head>
<style>
h1.visible {
visibility: visible
}

h1.hidden {
visibility: hidden
}
</style>
</head>
<body>
```



```
<h1 class="visible">This is a visible heading</h1>
<h1 class="hidden">This is an invisible heading</h1>
<p>Notice that the invisible heading still takes up space.</p>

</body>
</html>
```

2.13.1 Normal flow,

In normal flow, the block - level elements within a page will flow from top to bottom.

Chapter 2.13 Positioning and Layout with CSS

2.13.6 z-index Property

The z-index property specifies the stack order of an element.
An element with greater stack order is always in front of an element with a lower stack order.

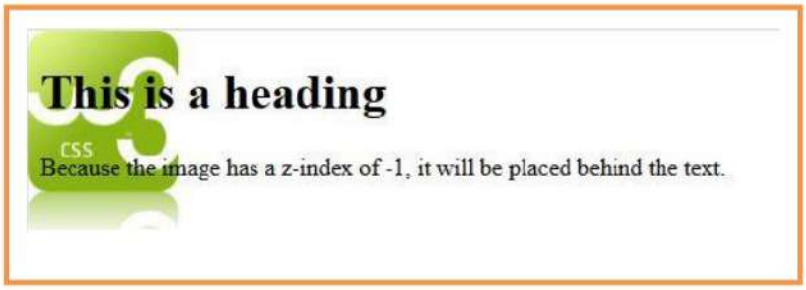
Note: z-index only works on positioned elements (position:absolute, position:relative, or position:fixed).

CSS Syntax

z-index:auto|*number*|initial|inherit;

```
<html>
<head>
<style>
img {
position: absolute;
```

```
left: 0px;
top: 0px;
z-index: -3;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<imgsrc="w3css.gif" width="100" height="140">
<p>Because the image has a z-index of -3, it will be placed behind the text.</p>
</body>
</html>
```



2.13.2 The position Property

The position property allows you to specify how you want to control the position for a box (and is generally used to take items out of normal flow).

It can take the four values

Static This is the same as normal flow, and is the default, so you will rarely (if ever) see it specified.

Relative The position of the box can be offset from where it would be if it were left in normal flow.

Absolute The box is positioned exactly using x and y coordinates from the top - left corner of the containing element.

Fixed The position is calculated from the top - left corner of a browser window and does not change position if the user scrolls the window.

What is JavaScript?

JavaScript is a cross-platform, object-oriented scripting language. It is a small and lightweight language Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.

JavaScript contains a standard library of objects, such as `Array`, `Date`, and `Math`, and a core set of language elements such as operators, control structures, and statements.

So finally, JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else.

Difference between JavaScript and Java

JavaScript is a very free-form language compared to Java. You do not have to declare all variables, classes, and methods. You do not have to be concerned with whether methods are public, private, or protected, and you do not have to implement interfaces. Variables, parameters, and function return types are not explicitly typed.

Java is a class-based programming language designed for fast execution and type safety. Type safety means, for instance, that you can't cast a Java integer into an object reference or access private memory by corrupting Java bytecodes. Java's class-based model means that programs consist exclusively of classes and their methods. Java's class inheritance and strong typing generally require tightly coupled object hierarchies. These requirements make Java programming more complex than JavaScript programming.

What Is Programming About?

Programming is largely about performing different types of calculations upon various types of data (including numbers, text and graphics) in all programming languages.

You can perform tasks such as:

- *Performing mathematical calculations on numbers such as addition, subtraction, multiplication and division.*
- *Working with text to find out how long a sentence is, or where the first occurrence of a specified letter is within a section of text.*
- *Checking if one value (numbers or letters) matches another.*
- *Checking if one value is shorter or longer, lower or higher than another.*
- *Performing different actions based on whether a condition (or one of several conditions) is met. For example, if a user enters a number less than 10, a script or program can perform one action; otherwise it will perform a different action.*
- *Repeating an action a certain number of times or until a condition is met (such as a user pressing a button).*

In object - oriented programming languages, real - life objects are represented (or modeled) using a set of objects , which form an object model . For example, a car object might represent a car, a basket object might represent a shopping basket, and a document object could represent a document such as a web page.

How to Add a Script to Your Pages

JavaScript can either be embedded in a page or placed in an external script file (rather like CSS). You add scripts to your page inside the < script > element. The type attribute on the opening < script > tag indicates what scripting language will be found inside the element, so for JavaScript you use the value text/JavaScript .

Example

```
< html >
 < body >
      < p >
           < script type="text/javascript" >
                document.write("My first JavaScript")
           < /script >
      < /p >
 < /body >
< /html >
```

You can also write JavaScript in external documents that have the file extension .js (just in the same way that you can write external style sheets). This is a particularly good option because:

- *If your script is used by more than one page you do not need to repeat the script in each page that uses it.*
- *If you want to update your script you need only change it in one place.*
- *It makes the XHTML page cleaner and easier to read.*

When you place your JavaScript in an external file, you need to use the src attribute on the < script > element; the value of the src attribute should be an absolute or relative URL pointing to the file containing the JavaScript.

For example:

```
< script type="JavaScript" src="scripts/validation.js" > < /script >
```

So there are three places where you can put your JavaScripts , and a single XHTML document can use all three because there is no limit on the number of scripts one document can contain:

- **In the < head > of a page:** These scripts will be called when an event triggers them.
- **In the < body > of a page:** These scripts will run as the page loads.
- **In an external file:** If the link is placed inside the < head > element, the script is treated the same as when the script lives inside the head of the document waiting for an event to trigger it, whereas if it is placed in the < body > element it will act like a script in the body section and execute as the page loads.

JavaScript Comments

Single Line Comment

Single line comments start with //.
Any text between // and the end of the line will be ignored by JavaScript (will not be executed).
This example uses a single-line comment before each code line:

Example

```
// Change heading:
document.getElementById("myH").innerHTML = "My First Page";
// Change paragraph:
document.getElementById("myP").innerHTML = "My first paragraph.";
```

Multiline Comment

Multi-line comments start with /* and end with */.

Any text between /* and */ will be ignored by JavaScript.

This example uses a multi-line comment (a comment block) to explain the code:

Example

```
/*
The code below will change
the heading with id = "myH"
and the paragraph with id = "myP"
in my web page:
*/
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

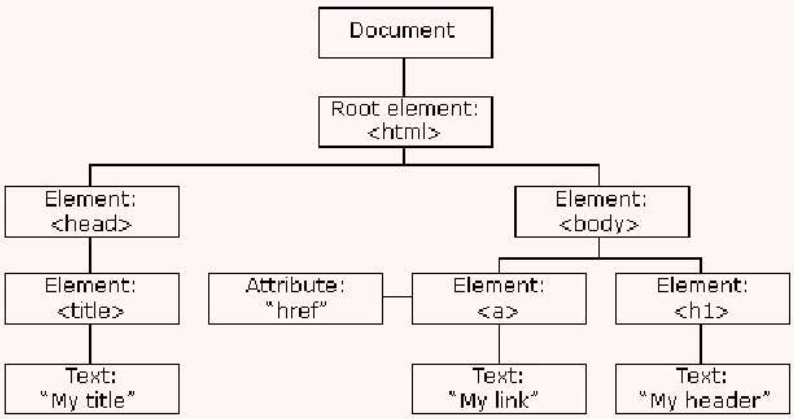
What is the HTML DOM?

*The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:*

- *The HTML elements as **objects***
- *The **properties** of all HTML elements*
- *The **methods** to access all HTML elements*
- *The **events** for all HTML elements*

*In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.The **HTML DOM** model is constructed as a tree of **Objects**:*

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

The DOM Programming Interface

The HTML DOM can be accessed with JavaScript (and with other programming languages).In the DOM, all HTML elements are defined as ***objects****.The programming interface is the properties and methods of each object.*
A ***property*** *is a value that you can get or set (like changing the content of an HTML element).A* ***method*** *is an action you can do (like add or deleting an HTML element).*

```
Example:
<html>
<body>
 <h2>My First Page</h2>
 <p id="demo"></p>
 <script>
       document.getElementById("demo").innerHTML = "Hello World!";
 </script>
</body>
</html>
```

The getElementById Method

The most common way to access an HTML element is to use the id of the element. In the example above the getElementById method used id="demo" to find the element.

The innerHTML Property

The easiest way to get the content of an element is by using the ***innerHTML*** *property.The innerHTML property is useful for getting or replacing the content of HTML elements.*

The HTML DOM Document Object

The document object represents your web page.If you want to access any element in an HTML page, you always start with accessing the document object.Below are some examples of how you can use the document object to access and manipulate HTML.

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element
document.write(text)	Write into the HTML output stream

JavaScript test() Method

The test() method tests for a match in a string.This method returns true if it finds a match, otherwise it returns false.

Syntax

***RegExpObject*.test(string)**

Return Value

Type	Description
Boolean	Returns true if it finds a match, otherwise it returns false

RegExp Object

A regular expression is an object that describes a pattern of characters.Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.

Syntax

***/pattern/modifiers*;**

Example:
var pattern = /hello/i;
var str="hellow world";
var result=pattern.test(str);
// result will be true;

Modifiers

Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\char	Find a char

Quantifiers

Quantifier	Description
n+	Matches any string that contains at least one *n*
n$	Matches any string with *n* at the end of it
^n	Matches any string with *n* at the beginning of it

Starting to Program with JavaScript

JavaScript Variables

Variables are containers for storing data values. All variables are declared by var keyword not need to define data type. After the declaration, the variable has no value. (Technically it has the value of ***undefined****). To* ***assign*** *a value to the variable, use the equal sign:*

All JavaScript variables must be identified with unique ***names****. These unique names are called* ***identifiers****. Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).*

The general rules for constructing names for variables (unique identifiers) are:

- *Names can contain letters, digits, underscores, and dollar signs.*
- *Names must begin with a letter*
- *Names can also begin with $ and _ (but we will not use it in this tutorial)*
- *Names are case sensitive (y and Y are different variables)*
- *Reserved words (like JavaScript keywords) cannot be used as names*

JavaScript Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator syntax (condition?expr1:exper2)

JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4

JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

JavaScript String Operators

The + operator can also be used to add (concatenate) strings.

Example:

```
var pattern=/^h/g;
var str='hellow world';
var result=pattern.test(str);
//result=true
```

Example of Email Validation:

```
var email= "ram@shta.com";
var patt1 = /^[a-z]\S+@\S+\.\S+/i;
```

```
 var result = patt1.test(email);
//result=true

Example of number only validation:
var num1 = "20122013";
var num2= "2012and2013";
 var patt1 =/[^0-9]/g
 var result1 = patt1.test(num1);
var result2=patt1.test(num2);
//result1=false
//result2=true
Example of String starts with capital-letter and end with digit validation
var result;
var result;
 var num1 = "Ram9";
var startpatt =/^[A-Z]/;
var endpatt =/[0-9]$/;
var startresult = startpatt.test(num1);
var endresult = endpatt.test(num1);
if(startresult==true && endresult==true){
      result=true;
}
else{
      reslut=false;
}
      //result=true;
```

String Methods and Properties

String Length

The **length** property returns the length of a string:

```
Example
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
//returns 26
```

indexOf()

The **indexOf()** method returns the index of (the position of) the **first** occurrence of a specified text in a string: if not found returns -1.

```
Example
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
//returns 7
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate",15);
//returns 21
```

lastIndexOf()

The **lastIndexOf()** method returns the index of the **last** occurrence of a specified text in a string: if not found returns -1.

```
Example:
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
//returns 21
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate",20);
//returns 7
```

substring(start, end)

substring() extracts a part of a string and returns the extracted part in a new string. substring() cannot accept negative indexes.

```
Example
var str = "Apple, Banana, Kiwi";
var res = str.substring(7, 13);
//retuns Banana
```

substr()

substr() is similar to substring(). The difference is that the second parameter specifies the **length** of the extracted part.

```
Example:
var str = "Apple, Banana, Kiwi";
var res = str.substr(7, 6);
//returns Banana
```

toLowerCase()

Converts a string to lowercase.

toUpperCase()

Converts a string to uppercase.

trim()

The trim() method removes whitespace from both sides of a string.

```
Example:
 var str = myTrim("        Hello World!        ");
    alert(str);
```

More Methods

Method	Description
charAt()	Returns the character at the specified index (position)
endsWith()	Checks whether a string ends with specified string/characters
includes()	Checks whether a string contains the specified string/characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
repeat()	Returns a new string with a specified number of copies of an existing string. Syntax :*string*.repeat(*count*)
replace()	Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced. Syntax: *string*.replace(*searchvalue, newvalue*)
startsWith()	Checks whether a string begins with specified characters
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLowerCase()	Converts a string to lowercase letters
toUpperCase()	Converts a string to uppercase letters
trim()	Removes whitespace from both ends of a string

Date Object Methods

Method	**Description**
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1 1970, and a specified date
getYear()	Returns the year in 2 digit
now()	Returns the number of milliseconds since midnight Jan 1, 1970
parse()	Parses a date string and returns the number of milliseconds since January 1, 1970
setDate()	Sets the day of the month of a date object
setFullYear()	Sets the year of a date object
setHours()	Sets the hour of a date object
setMilliseconds()	Sets the milliseconds of a date object
setMinutes()	Set the minutes of a date object
setMonth()	Sets the month of a date object
setSeconds()	Sets the seconds of a date object
setTime()	Sets a date to a specified number of milliseconds after/before January 1, 1970
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string

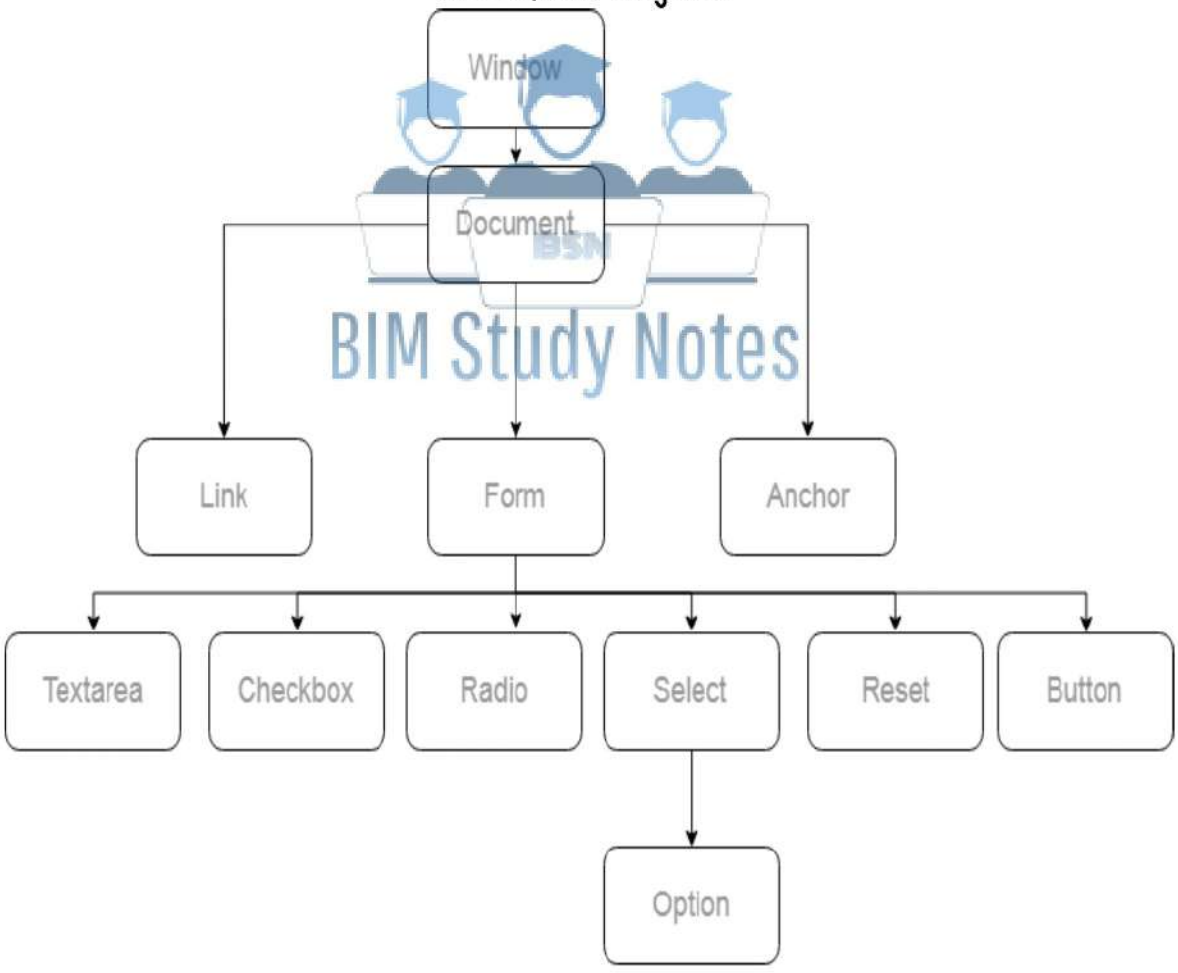
DOM (Document Object Model) (2020 Make-up)

A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.

The way a document content is accessed and modified is called the **Document Object Model**, or **DOM**. The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document. It defines the **logical structure** of documents and the way a document is accessed and manipulated.

Properties of DOM:

Let's see the properties of document object that can be accessed and modified by the document object.



1. **Window Object:** Window Object is at always at top of hierarchy.
2. **Document object:** When HTML document is loaded into a window, it becomes a document object.
3. **Form Object:** It is represented by ***form*** tags.
4. **Link Objects:** It is represented by ***link*** tags.
5. **Anchor Objects:** It is represented by ***a href*** tags.
6. **Form Control Elements:**: Form can have many control elements such as text fields, buttons, radio buttons, and checkboxes, etc.

***Methods* of Document Object**:

1. **write("string"):** writes the given string on the document.
2. **getElementById():** returns the element having the given id value.
3. **getElementsByName():** returns all the elements having the given name value.
4. **getElementsByTagName():** returns all the elements having the given tag name.
5. **getElementsByClassName():** returns all the elements having the given class name.

Accessing field value by document object

In this example, we are going to get the value of input text by user. Here, we are using **document.form1.name.value** to get the value of name field.
Here, **document** is the root element that represents the html document.
form1 is the name of the form.
name is the attribute name of the input text.
value is the property, that returns the value of the input text.

Let's see the simple example of document object that prints name with welcome message.

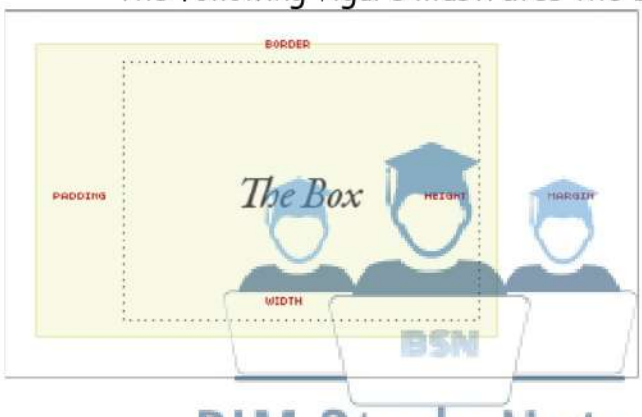
```
<script type="text/javascript">
function printvalue(){
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>

<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

CSS box model (2020-Makeup)

CSS box model is a container which contains multiple properties including borders, margin, padding and the content itself. It is used to create the design and layout of web pages. It can be used as a toolkit for customizing the layout of different elements. The web browser renders every element as a rectangular box according to the CSS box model.
Box-Model has multiple properties in CSS. Some of them are given below:

- borders
- margins
- padding
- Content

The following figure illustrates the box model.



Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

Example **Demonstration of the box model:**

```
div {
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```