HTML Introduction: -

1. HTML stands for Hyper Text Markup Language.
2. HTML is used to create static web pages.
3. HTML is widely used language on the web.
4. We can create static website by HTML only.
5. An HTML document is made of many HTML tags and each HTML tag contains different content.
6. Browsers do not display the HTML tags, but use them to render the content of the page.
7. You can use any text editor like notepad, notepad++, or Edit plus etc.

HTML SYNTAX: -

1. An HTML element usually consists of a start tag and end tag, with the content inserted in between:<tagname>Content goes here...</tagname>.
2. All HTML documents must start with a document type declaration: <! DOCTYPE html>.
3. The HTML document itself begins with <html>and ends with </html>.
4. Head tag represents the document's header and its start with <head> and close with </head>.
5. The <title> tag is used inside the head tag to mention the document title.
6. The visible part of the HTML document is between <body> and </body>.

```
<! DOCTYPE HTML>
<html>
 <head>
  <title>Your title goes here</title>
 </head>

 <body>
  Your content goes here
 </body>
</html>
```

Eg: -

```
<! DOCTYPE html>

<html>

<head>

          <title>hello</title>

</head>

<body>
```

bhargav reddy welcome to hyderabad

</body>

</html>

Output: -

Bhargav reddy welcome to hyderabad

HTML BASIC TAGS: -

1. <br> br stands for break line, it breaks the line of the code. It is useful for writing a poem or an address, where the division of lines is significant. <br> tag has no closing tag; it is used to break the line.
2. <pre> tag represents preformatted text. Whitespace inside this element is displayed as typed.
3. <code> tag is used for indicating a piece of code. The code tag surrounds the code being marked up. The code being marked up could represent an XML element name, a filename, a computer program, or any other string that a computer would recognize.

```
! DOCTYPE HTML>
<head>
 <title>pre tag</title>
</head>
<body>
 <pre>
  Text in a pre-element
  is displayed in a fixed-width
  font, and it preserves
  both spaces and
  line breaks
 </pre>
This text contains<br>a line break.
 <code>A piece of computer code</code>
</body>
</html>
```

Example: - (break tag)

<! DOCTYPE html>

<html>

```
<head>
        <title>break tag</title>
</head>
<body>
hello bhargav <br>
today u r learning html<br>
basics
</body>
</html>
```

Output: -

```
hello bhargav
today u r learning html
basics
```

Example for pre tags: -pre tags printed the content as it is on the browser.

```
<! DOCTYPE html>
<html>
<head>
        <title>bhargav reddy</title>
</head>
<body>
<pre>
        bhargav   reddy
reddy
kudala
</pre>
</body>   </html>
```

Output: -

```
bhargav   reddy
reddy
kudala
```

Note:-Instead of using many break tags we are using pre-tags to display the output as it is on browser.

Code-tags: - It is used to display the code like java, java script as it is

Syntax: -

```
<code> </code>
```

Example: -

```
<! DOCTYPE html>
<html>
<head>
        <title>bhargav reddy</title>
</head>
<body>
<pre>
        <code>
                void main ()
                {
                printf ("bhargav reddy ");
                }
        </code>
</pre>
</body>
</html>
```

Output: -

```
        void main ()
                {
                        printf ("bhargav reddy ");
                }
```

HEADING –TAGS: -

1. A HTML heading or HTML h tag can be defined as a title or a subtitle which you want to display on the webpage. When you place the text within the heading tags it is displayed on the browser in the bold format and size of the text depends on the number of heading.
2. There are six different HTML headings which are defined with the <h1> to <h6> tags.
3. <h1> defines the most important heading. <h6> defines the least important heading. Search engines use the headings to index the structure and content of your web pages.

```
<! DOCTYPE HTML>
<head>
 <title>heading -tags </title>
</head>

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

Example: -

```
<! DOCTYPE html>

<html>

<head>

        <title>heading-tags</title>

</head>

<body>

<h1> WELCOME TO HTML5 </h1>

        <H2>WELCOME TO HTML5</H2>

        <H3>WELCOME TO HTML5</H3>

        <H4>WELCOME TO HTML5</H4>

        <H5>WELCOME TO HTML5</H5>

        <H6>WELCOME TO HTML5</H6>

        <pre>

        <h1>    <a
href="http://kothaabhishek.co.in/HTML.html#11">kothaabhishek</a></h1>

       <h2> *above link is the reference notes for html-5*</h2>

        </pre>

</body>

</html>
```

Output: -

WELCOME TO HTML5

WELCOME TO HTML5

WELCOME TO HTML5

WELCOME TO HTML5

WELCOME TO HTML5

WELCOME TO HTML5

[kothaabhishek]()

above link is the reference notes for html-5

Paragraph-tags: -

1. HTML paragraph or HTML p tag is used to define a paragraph in a webpage.
2. It is a notable point that a browser itself add an empty line before and after a paragraph. <p> is opening tag and </p> is closing tag. The HTML <p> element defines a paragraph.

```
<! DOCTYPE HTML>
<head>
 <title>Paragraph tag</title>
</head>

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

Example: -

```
<! DOCTYPE html>

<html>

<head>

            <title>paragraph tag</title>

            </head>

            <body>

<p>
```

```
Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.

</p>

<p>

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.

            </p>

            <p>

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.

            </p>

            </body> </html>
```

Output: -

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.

Formatting –tags: -

1. HTML also defines special elements for defining text with a special meaning.
2. HTML uses elements like <b> and <i> for formatting output, like bold or italic text.
3. Formatting elements were designed to display special types of text.

```
<! DOCTYPE HTML>
<head>
 <title>formatting –tags in html </title>
</head>

<body>
 <b>This text is bold</b>
 <strong>This text is strong</strong>
 <i>This text is italic</i>
 <em>This text is emphasized</em>
 <h2>HTML <small>Small</small> Formatting</h2>
 <h2>HTML <mark>Marked</mark> Formatting</h2>
 <p>My favorite color is <del>blue</del> red.</p>
 <p>My favorite <ins>color</ins> is red.</p>
 <p>This is <sub>subscripted</sub> text. </p>
 <p>This is <sup>superscripted</sup> text. </p>
</body>
</html>
```

Types of formatting-tags: -

a) Bold tag: - It is used to display the content in bold.

Syntax: - <b> </b>

b) Italic –tag:-It is used to display the content in Italic style.

Syntax:-<i> </i>

c)Marked tag:-It is used to highlight the content on browser.

syntax:-<mark> </mark>

d)Underline-tag:-It is used to underline the content on browser.

Syntax:-<u> </u>

e) Superscript-tag:-It is used to display the mathematical or chemical reactions on browser.

Syntax:-<sup> </sup>

f) Subscript tag: - It is used to display the chemical reactions on browser.

Syntax:-<sub> </sub>

g) Small and big-tags: - It is used to display the small or big size of the content on browser.

Syntax:-<small> </small>

<big> </big>

f) deleted –tag:-It is used to cross the content on the browser.

Syntax:-<del> </del>

Example: -

<! DOCTYPE html>

```
<html>
<head>
        <title>formatting tags</title>
        </head>
        <body>
                <pre>
BHARGAV REDDY KUDALA
<b> BHARGAV REDDY </b>
<i>BHARGAV REDDY </i>
<mark> BHARGAV REDDY </mark>
<u> BHARGAV REDDY </u>
X<sup>2</sup>+Y<sup>2</sup>
H<sub>2</sub>O<sub>2</sub>
BHARGAV REDDY <small>KUDALA</small>
<big> VENNA PUSALA </big>
Welcome to <del> CSS</del> HTML Class
</pre>
</body> </html>
```

Output: -

BHARGAV REDDY KUDALA

HTML

Hitech city

Biryani

hyderabad

X^2+Y^2

H_2O_2

BHARGAV REDDY KUDALA

VENNA PUSALA

Welcome to ~~css~~ HTML Class

Background attribute: -

1. These are the attributes to change web page looking.
2. Attributes are used inside a tag and it follows att_name="value" format Ex: text="red"
3. bgcolor attribute is used to change the background-color in a web page.
4. text attribute is used to change the text color in a web page.
5. background attribute is used to set an image as a background in a web page. Here we have to give the whole image url- with extension.

```
<! DOCTYPE HTML>
<head>
 <title>background attribute</title>
</head>

<body bgcolor="red" text="white">
 Hello Welcome To HTML Tutorials
</body>
</html>
<! DOCTYPE HTML>
<head> <title> background attribute </title>
</head><body background="html1.png" text="yellow">
 Hello Welcome to HTML Tutorials
</body></html>
```

Example: -

```
<! DOCTYPE html>

<html>

<head>

            <title>background</title>

</head>

<body bgcolor="aqua" text="white" background="C:\Users\user\OneDrive\Pictures\rama family pic.jpeg">

            <mark><b><i>BHARGAV REDDY KUDALA</b></i></mark>

            <p>

                        Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.

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

Note: -

1)**background attribute** is used to insert the image on the background.

2)**bgcolor attribute** is used to change the background color on the web page.

3)**text attribute** is used to change the color of the text.

Align attributes: -

1)DE faulty the content written is on left side of the page.

2)Align attributes is used to write the content on the browser at center of the page/right of the page.

Example: -

```
<! DOCTYPE html>

<html>

<head>

        <title>align attributes</title>

</head>

<body>

<p align="right"> bhargav reddy kudala <br>

        welcome to hyderabad

</p>

<p align="left"> bhargav reddy kudala <br>

        welcome to hyderabad

        </p>

        <h2 align="center"> MURARI PALLI </h2>

</body>

</html>
```

Output: -

```
bhargav reddy kudala                MURARI PALLI        bhargav reddy kudala
```

welcome to hyderabad welcome to hyderabad

IMAGE-TAGS: -

1. You can insert any image in your web page by using <img> tag.
2. The simple syntax to use this tag is <img src="Image URL" ... attributes-list/>
3. The src attribute is used to give the address of the image with extension.
4. The alt attribute is a mandatory attribute which specifies an alternate text for an image, if the image cannot be displayed.
5. You can set image width and height based on your requirement using width and height attributes.
6. By default, image will have a border around it, you can specify border thickness in terms of pixels/value using border attribute. A thickness of 0 means, no border around the picture.
7. By default, image will align at the left side of the page, but you can use align attribute to set it in the center or right.
8. Image tag has only open tag.

```
<! DOCTYPE HTML>
<head>
 <title>image tags </title>
</head>

<body>
 <img src="html.png" alt="HTML5" height="200" width="250">
 <img src="css3.png" alt="css3" height="200" width="250">
</body>
</html>
```

Example: -

```
<! DOCTYPE html>

<html>

<head>

<title>image tag</title>

</head>

<body>

<img src="C:\Users\user\OneDrive\Pictures\rama family pic.jpeg"alt="rama family logo"
width="300 "height="300" align="left" border="5"> </body> </html>
```

Note: -

<img src=” location of the file with extension(.jpg/.jpeg)” alt=” main logo”>

1)If browser is not displaying the image, then with the help of Alt attribute is we can display the image on the web page.

2)width and height attributes are used to control the size of the image based on the requirement, align attribute is used to fix the location (left/right/center) of the image on the webpage.

3)border attribute is used to make a border of the image, default border order is 0.

TABLE –TAGS: -

1. The HTML tables allow to arrange data like text, images, links, other tables, etc. into rows and columns of cells.
2. The HTML tables are created using the <table> tag in which the <tr> tag is used to create table rows and <td>tag is used to create data cells.
3. Here border is an attribute of <table> tag and it is used to put a border across all the cells. If you do not need a border, then you can use border="0”. You can also set border color also using border color attribute.
4. Table heading can be defined using <th> tag. This tag will be put to replace <td> tag, which is used to represent actual data cell.
5. There are two attributes called cell padding and cell spacing which you will use to adjust the white space in your table cells.
6. **cell padding** represents the distance between cell borders and the content within a cell.
7. The **cell spacing** attribute defines the width of the border.
8. You will use **colspan** attribute if you want to merge two or more columns into a single column. Similar way you will use **rows pan** if you want to merge two or more rows.
9. bgcolor attribute is used to set background color for whole table or just for one cell.
10. background attribute is used to set background image for whole table or just for one cell.
11. You can set a table width and height using width and height attributes.
12. The caption tag will serve as a title or explanation for the table and it shows up at the top of the table.

```
<! DOCTYPE HTML>
<head>
 <title>Table tags </title>
</head>
<body>
   <table border="2" cell padding="3" cell spacing="3"
  bgcolor="aqua">
      <tr>
         <th colspan="3" bgcolor="red">Programming Tutorials</th>
     </tr>
     <tr>
         <td bgcolor="white">HTML</td>
      <td rowspan="4" bgcolor="orange">In Telugu</td>
      <td bgcolor="white">CSS</td>
      </tr>
      <tr>
      <td bgcolor="pink">SQL</td>
      <td bgcolor="pink">Java</td>
      </tr>
      <tr>
      <td bgcolor="light green">JavaScript</td>
      <td bgcolor="light green">Python</td>
      </tr>
      <tr>
      <td bgcolor="yellow">PHP</td>
      <td bgcolor="yellow">go</td>
      </tr>
 </table>
</body>
</html>
```

Example: -

```
<! DOCTYPE html>

<html>

<head>

        <title>table tags th=table header (bold instead of bold tag) </title>

</head>

<body>

        <table border="4">

                <tr>

                        <td><b> Student name</td></b>

                        <td><b>Father Name</td></b>

                        <th>Marks</th>

                        <th>Grade</th>

        </tr>
        <tr>

                <td>Bhargav reddy </td>

<td>Siva Prasad reddy</td>

<td>550</td>

<td>Distinction</td>

        </tr></table>

<table bgcolor="pink" border="3" cellpadding="5" cellspacing="5" width="300"
height="400" bordercolor="green" align="center">

        <caption> <b>Student details</b></caption>

        <tr>

        <th rowspan="2"> Student Name </th>

        <th colspan="3">Father Name </th>

        <th>Marks</th>

        <th>Grade</th>

        </tr>

        <tr bgcolor="aqua">

<td>Bhargav reddy </td>

<td>Siva prasad reddy</td>

<td>550</td>

<td bgcolor="yellow">Distinction</td> </tr>

<table bgcolor="green" background="C:\Users\user\OneDrive\Pictures\rama family pic.jpeg"
border="3" cellpadding="5" cellspacing="5" width="300" height="400" bordercolor="red"
align="left">

        <tr>

        <th rowspan="2"> Student Name </th>

        <th colspan="3">Father Name </th>

        <th>Marks</th>
```

```
          <th>Grade</th>

          </tr>

          <tr>

<td>Bhargav reddy </td>

<td>Siva Prasad reddy</td>

<td>550</td>

<td>Distinction</td>

          </tr>

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

NOTE: -

1)**Cell padding** →It is used to increase the space between word and table /border .

2)**Cell spacing**→It is used to increase the space between the word to word i.e. first table content to second table content

3)**Row span**→It is used to display the content of two rows into one row

4)**Col span**→It is used to display the content of multiple columns into single column.

HTML LIST TAGS: -

1. HTML offers three ways for specifying lists of information. All lists must contain one or more list elements.
2. <ul> - An unordered list. This will list items using plain bullets.
   The Unordered list starts with <ul> tag and list items start with the <li> tag.
   You can use type attribute for <ul> tag to specify the type of bullet you like. By default, it is a disc. But you can change to square or circle or none.
3. <ol> - An ordered list. This will use different schemes of numbers to list your items.
   The numbering starts at one and is incremented by one for each successive ordered list tagged with <li>.
   You can use type attribute for <ol> tag to specify the type of numbering you like. By default it is a number.
   But you can change to Roman Numbers(I),Small roman numbers(i), alphabets(A),small alphabets (a).
   You can use start attribute for <ol> tag to specify the starting point of numbering you need.
4. <dl> - A definition list. This arranges your items in the same way as they are arranged in a dictionary.
   <dl> - Defines the start of the list.

<dt> - Defines a term.
<dd> - Defines term definition

```
<! DOCTYPE HTML>
<head>
 <title>list tags </title>
</head>

<body>
 <h2>An Unordered HTML List</h2>
 <ul>
   <li>HTML</li>
   <li>CSS</li>
   <li>JavaScript</li>
 </ul<
 <h2>An Ordered HTML List</h2>
 <ol>
   <li>SQL</li>
   <li>Java</li>
   <li>Python</li>
 </ol>
 <h2>HTML Description Lists</h2>
 <dl>
  <dt>Coffee</dt>
    <dd>- black hot drink</dd>
   <dt>Milk</dt>
    <dd>- white cold drink</dd>
 </dl>
 </body>
</html>

<! DOCTYPE HTML>
<head>
 <title>list tags </title>
</head>

<body>
<ul>
 <ul type="square">
   <li>Coffee</li>
   <li>Tea</li>
  <li>Milk</li>
 </ul>
 <ul type="circle">
   <li>Coffee</li>
   <li>Tea</li>
   <li>Milk</li>
 </ul>
 <ol type="a">
   <li>Coffee</li>
   <li>Tea</li>
    <li>Milk</li>
 </ol>
<ol type="i">
   <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
 </ol></body> </html>
```

Example: -

```
<! DOCTYPE html>

<html>

<head>

                <title>list; order and un-order list </title>

</head>

<body>

<ul type="disc">

                <li>Bhargav reddy </li>

                <li>Naveen</li>

                <li>Bhanu Prakash reddy</li>

                <li>Siva Prasad reddy</li>

                <li>Rukmini</li>

</ul>

<hr>

<ul type="square">
```

```
        <li>Bhargav reddy </li>
        <li>Naveen</li>
        <li>Bhanu Prakash reddy</li>
        <li>Siva Prasad reddy</li>
        <li>Rukmini</li>
</ul>
<hr>
<ul type="circle">
        <li>Bhargav reddy </li>
        <li>Naveen</li>
        <li>Bhanu Prakash reddy</li>
        <li>Siva Prasad reddy</li>
        <li>Rukmini</li>
</ul>
<hr>
<ol>
        <li>Siva Prasad reddy</li>
        <li>Rukmini</li>
        <li>Bhargav reddy</li>
        <li>Bhanu Prakash reddy</li>
        <li>Naveen Kumar</li>
</ol>
<hr>
<ol type="a">
        <li>Siva Prasad reddy</li>
        <li>Rukmini</li>
        <li>Bhargav reddy</li>
        <li>Bhanu Prakash reddy</li>
        <li>Naveen Kumar</li>
</ol>
<hr>
<ol type="A">
        <li>Siva Prasad reddy</li>
        <li>Rukmini</li>
        <li>Bhargav reddy</li>
        <li>Bhanu Prakash reddy</li>
        <li>Naveen Kumar</li>
</ol>
<hr>
<ol type="i">
        <li>Siva Prasad reddy</li>
        <li>Rukmini</li>
        <li>Bhargav reddy</li>
        <li>Bhanu Prakash reddy</li>
        <li>Naveen Kumar</li>
</ol>
<hr>
<ol type="I">
        <li>Siva Prasad reddy</li>
        <li>Rukmini</li>
        <li>Bhargav reddy</li>

        <li>Bhanu Prakash reddy</li>

        <li>Naveen Kumar</li>

</ol>

<hr>

<ol start="100">

        <li>Siva Prasad reddy</li>

        <li>Rukmini</li>

        <li>Bhargav reddy</li>

        <li>Bhanu Prakash reddy</li>

        <li>Naveen Kumar</li>

</ol>

</body>

</html>
```

Definition tags: -

Syntax: -

```
<! DOCTYPE html>

<html>

<head>

        <title>definition list tags</title>

</head>

<body>

<dl>

<dt> definition title</dt>

<dd> definition data </dd> </dl>
```

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

Example: -

```
<! DOCTYPE html>

<html>

<head>

        <title>definition list tags</title>

</head>

<body>

<dl>

        <dt>HTML</dt>

        <dd>Hypertext markup language</dd>

        <dt>CSS</dt>

        <dd>Cascading style sheets</dd>

        <dt> JS</dt>

        <dd>Java script</dd>

</dl>

</body>

</html>
```

Note: -

1) dl is definition list.

2)**dt** is definition title and **dd** is definition data.

Hyperlinks tags: -

1. A webpage can contain various links that take you directly to other pages and even specific parts of a given page. These links are known as hyperlinks.
2. Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images.

3. A link is specified using HTML tag <a>. This tag is called anchor tag and anything between the opening <a>tag and the closing </a> tag becomes part of the link.
4. Following is the simple syntax to use <a> tag. <a href="Document URL" ... attributes-list>Link Text</a>
5. target attribute is used to specify the location where linked document is opened. Possible options are
   _blank Opens the linked document in a new window or tab.
   _self Opens the linked document in the same frame.
   _parent Opens the linked document in the parent frame.
   _top Opens the linked document in the full body of the window.
6. You can set colors of your links, active links and visited links using link, a-link and v-link attributes of <body>tag.
7. You can create text link to make your PDF, or DOC or ZIP files downloadable. This is very simple; you just need to give complete URL of the downloadable file.
8. It's simple to use an image as hyperlink. We just need to use an image inside hyperlink at the place of text.

```
<! DOCTYPE HTML>
<head>
 <title>hyperlinks tag </title>
</head>

<body>
If you want to go to google
<a href="http://www.google.com" target="_blank">Click Here</a>
</body>
</html>
```

Example: -

```
<! DOCTYPE html>

<html>

<head>

<title>hyperlinks by using anchor tag</title>

</head>

<body link="red" alink="green" vlink="blue">

<a href="Listtags.html">click here</a>

<a href="C:\Users\user\OneDrive\Pictures\img5.jpg" target="_blank"><img
src="C:\Users\user\OneDrive\Pictures\img5.jpg" border="3"></a>
```

```
<a href="Listtags.html"target="_blank">download here </a>

</body>

</html>
```

FONT TAGS: -

1. Fonts play very important role in making a website more user friendly and increasing content readability.
2. HTML tag to add style, size, and color to the text on your website.
3. You can set content font size using size attribute. The range of accepted values is from 1(smallest) to 7(largest). The default size of a font is 3.
4. You can set font face (font style) using face attribute.
5. You can set font color using color attribute.

```
<! DOCTYPE HTML>
<head>
 <title>font tags </title>
</head>

<body>
<font size="3" color="red">This is some text! </font> <br>
<font size="7" color="blue">This is some text! </font> <br>
<font face="Times new roman " color="green">This is some text! </font><br>
</body>
</html>
```

Example: -

```
<! DOCTYPE html>

<html>

<head>

            <title>fonts tag like size of the font, style of the font, color of the font </title>

</head>

<body>
```

```
welcome to <font color="blue">Hyderabad</font> <font color="red"> Bhargav reddy
kudala<br><hr>

 <font face="Algerian"><font color="sky blue"> welcome to <font face="Colonna
MT">html</font></font></font> </font><br><hr>

 <font size="5" face="Algerian">welcome to css and java script </font><hr>

</body> </html>
```

Output: -

welcome to Hyderabad (blue color) Bhargav reddy kudala (red color)

---

welcome to html (red color)

---

welcome to css and java script (font size is 5)

MARQUEE TAG: -

1. An HTML marquee is a scrolling piece of text displayed either horizontally across or vertically down your webpage depending on the settings.
2. This is created by using HTML <marquee> tag.
3. <marquee att_name="att_value". more attributes> Text here </marquee>
4. Attributes are:
   width This specifies the width of the marquee. This can be a value like 10 or 20% etc.
   height This specifies the height of the marquee. This can be a value like 10 or 20% etc.
   direction This specifies the direction in which marquee should scroll. This can be a value like up, down, left or right.
   →Scroll delay This specifies how long to delay between each jump. This will have a value like 10 etc.
   →**scrollamount** This specifies the speed of marquee text. This can have a value like 10 etc.
   loop This specifies how many times to loop. The default value is INFINITE, which means that the marquee loops endlessly.
   →**bgcolor** This specifies background color in terms of color name or color hex value.
   hspace This specifies horizontal space around the marquee. This can be a value like 10 or 20% etc.
   →vspace This specifies vertical space around the marquee. This can be a value like 10 or 20% etc.
5. Default the speed of the scrolling content is 5.

```
<! DOCTYPE HTML>
<head>
 <title>Marquee tag (used to scroll the content /image on the webpage) </title>
```

```
</head>

<body>
   <marquee>This is basic example of marquee</marquee>
   <marquee direction = "right">The direction of text
      will be from left to right. </marquee>
   <marquee scrolldelay="10">Using Scroll Delay</marquee>
   <marquee scrollamount="2">Using Scroll Amount</marquee>
   <marquee bgcolor="yellow">Using Background Color</marquee>
</body></html>
```

Example: -

```
<! DOCTYPE html>

<html>

<head>

            <title>Marquee tags (scrolling the line in 'n' no. of times in the website) </title>

</head>

<body>

<marquee scrollamount="10"> <font color="red"><font face="Algerian">RAM
LAKSHMAN BHARATH SHATRUGN </marquee></font>

</font><hr>

<marquee scrollamount=50><img src="C:\Users\user\OneDrive\Pictures\rama family
pic.jpeg" width="300"height="300"></marquee><hr>

<marquee width="600" height="600" direction="down"><img
src="C:\Users\user\OneDrive\Pictures\img8.jpg"
width="200"height="200"></marquee><hr>

<marquee scroll amount="10" direction="up"><img
src="C:\Users\user\OneDrive\Pictures\rama family pic.jpeg"
width="200"height="200"></marquee><hr>

<marquee direction="down" bgcolor="black"><font color="white"><b>WELCOME TO
HTML CSS JAVA SCRIPT</marquee></font></b><hr>

</body>

</html>
```

AUDIO-TAGS: -

1. HTML audio tag is used to define sounds such as music and other audio clips. Currently there are three supported file format for HTML 5 audio tag.
2. auto play Specifies that the audio will start playing as soon as it is ready.
3. loop Specifies that the audio will start over again, every time it is finished.
4. src Specifies the URL of the audio file.
5. controls Specifies that audio controls should be displayed (such as a play/pause button etc.)
6. autoplay is an attribute.

```
<! DOCTYPE HTML>
<head>
 <title>Audio tags</title>
</head>

<body>
<audio controls autoplay loop>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>
</body>
</html>
```

Video-tags: -

1. HTML 5 supports <video> tag also. The HTML video tag is used for streaming video files such as a movie clip, song clip on the web page.
2. autoplay Specifies that the audio will start playing as soon as it is ready.
3. controls Specifies that audio controls should be displayed (such as a play/pause button etc.)
4. loop Specifies that the audio will start over again, every time it is finished.
5. src Specifies the URL of the audio file.
6. height pixels Sets the height of the video player.
7. widthsets the width of the video player.
8. poster Specifies an image to be shown while the video is downloading, or until the user hits the play button

```
<! DOCTYPE HTML>
<head>
 <title>video tag</title>
</head>

<body>
<video width="300" height="200" controls autoplay loop>
  <source src="video.mp4" type="video/mp4">
  Your browser does not support the video tag.
```

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

Example (For audio and video tags): -

```
<! DOCTYPE html>

<html>

<head>

<title>Audio tag</title>

</head>

<body>

<audio autoplay=""controls loop="">

<source src="C:\Users\user\OneDrive\Documents\beeshma song.mp3">

</audio><hr>

<video controls loop="" width="300" height="300">

<source src="C:\Users\user\Videos\movies\1560329342r4uhe.mp4">

</video>

<hr>

<video controls=""loop=""width="200" height="200">

<source src="C:\Users\user\Videos\movies\www.3MovieRulz.ws - Gaddalakonda Ganesh (Valmiki) (2019) 720p Telugu Proper.mp4">

</video>

</body>

</html>
```

Note: -

1.**<source src=”   ”>** represents the location of the video with extension

2. <**audio autoplay=""controls loop**=""> represents the controls of the audio like sound increase or decrease button, ***Autoplay*** is used to play the song automatically.

CSS

Html--------->Display the content on Browser

Javascript---->is used to perform operations on HTML Elements

CSS:- Apply the styles for HTML

CSS:- Cascading Style Sheet

we can apply styles in 3 ways

1.Tag as selector

2. id as selector

3. class as selector

Tag as selector:- Applying the style based on tag name

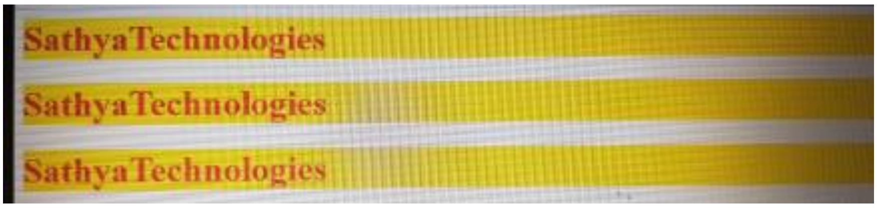
syntax:-

```
  tagname
  {
   propertyname:propertyvalue;
  }
```

Ex:-

```
<html>
<head>
    <style type="text/css">
        h1
        {
           color:red;
           background-color:yellow;
        }
    </style>
</head>
<body>
  <h1>  SathyaTechnologies </h1>
  <h1>  SathyaTechnologies </h1>
  <h1>  SathyaTechnologies </h1>
</body>
</html>
```

Output:



SathyaTechnologies

SathyaTechnologies

SathyaTechnologies

id as selector:- Applying the style based on Element id

if we want to apply different styles for same element then we have to apply id as selcetor

syntax:-

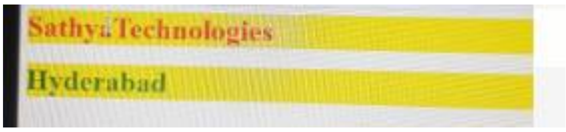
```
   #elementid
   {
     propertyname:propertyvalue;
   }
```

Ex:-

```
<html>
<head>
   <style type="text/css">
     #x
     {
       color:red;
       background-color:yellow;
     }
     #y
     {
       color:green;
       background-color:yellow;
     }
   </style>
</head>
<body>
<h1 id="x"> SathyaTechnologies  </h1>
<h1 id="y"> Hyderabad </h1>
</body>
```

```
</html>
```

Output:



Sathya Technologies

Hyderabad

class as selector:- Applying the style based on class name

if we want to apply same style for different Elements then we have to use class as selector

syntax:-

```
.classname
{
  propertyname:propertyvalue;
}
```

Ex:-

```
<html>
<head>
  <style type="text/css">
   .x
   {
    color:red;
    background-color:yellow;
   }
 </style>
</head>
<body>
<h1 class="x"> SathyaTechnologies </h1>
<h2 class="x"> SathyaTechnologies </h2>
<h3 class="x"> SathyaTechnologies </h3>
</body>
</html>
```

Output:

SathayTechnologies

SathayTechnologies

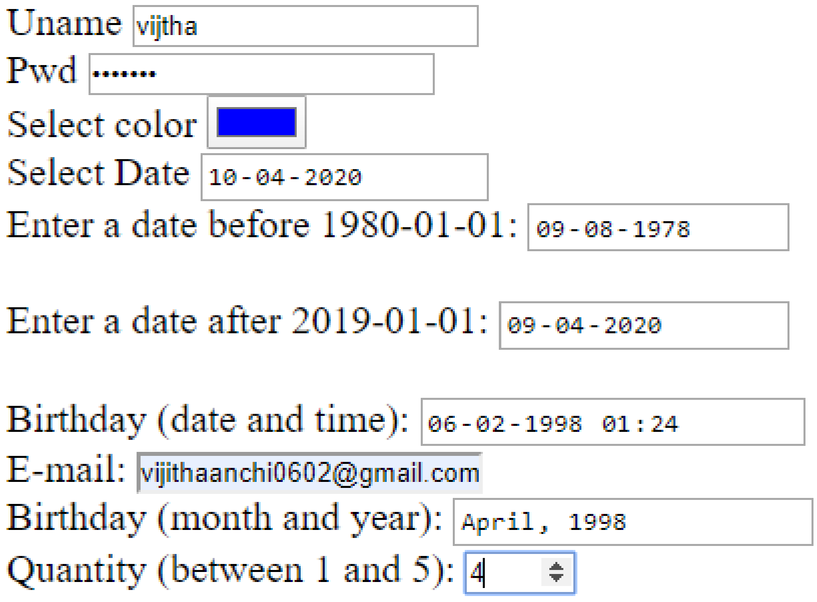
SathayTechnologies

Registration page2 using html5:

```
<!DOCTYPE html>
<html>
<head>
<title></title>
<meta charset="utf-8" />
</head>
<body>
Uname
<input type="text">
<br />
Pwd
<input type="password">
<br />
Select color
<input type="color">
<br />
Select Date
<input type="date">
<br />
Enter a date before 1980-01-01:
<input type="date" name="bday" max="1979-12-31"><br>
<br />
Enter a date after 2019-01-01:
<input type="date" name="bday" min="2020-01-02"><br>
<br />
```

```
Birthday (date and time):
<input type="datetime-local" name="bdaytime">
<br />
E-mail:
<input type="email" name="email">
<br>
Birthday (month and year):
<input type="month" name="bdaymonth">
<br />
Quantity (between 1 and 5):
<input type="number" name="quantity" min="1" max="5">
</body>
</html>
```

Output:



Uname vijtha

Pwd •••••••

Select color 🟦

Select Date 10-04-2020

Enter a date before 1980-01-01: 09-08-1978

Enter a date after 2019-01-01: 09-04-2020

Birthday (date and time): 06-02-1998 01:24

E-mail: vijithaanchi0602@gmail.com

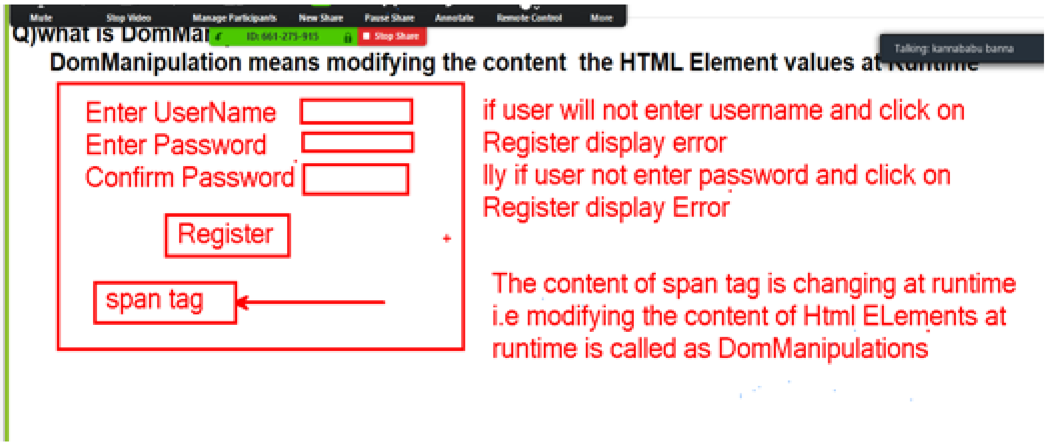
Birthday (month and year): April, 1998

Quantity (between 1 and 5): 4

Java Script

- Html purpose is to display content on Browser
- Javascript is scripting Language. It is used to perform client side validations. Javascript code will execute on Browser.

- JavaScript is used to perform Client side Activities like Validations, Dom Traversal, Dom Manipulations, Effects, Animations, Ajax Calls etc…
- Javascript was developed by netscape.
- Java script is case sensitive.(a!=A)
- Java Script program must save with .js extension.
- Javascript is scripting language and was developed by Netscape.
- whenever Html page is Loaded in Browser then DOM is ready
- once when Dom exists then only javascript can communicate with HTML Elements to perform Operations

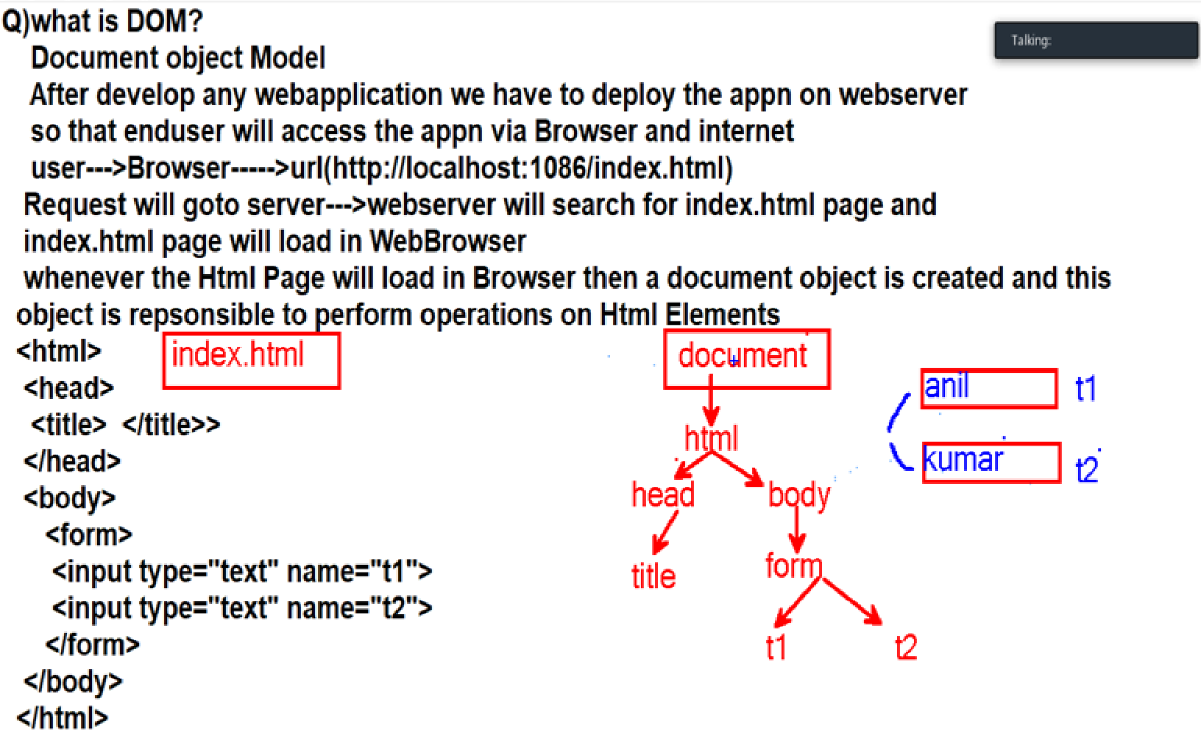


Q)what is DomMa

DomManipulation means modifying the content the HTML Element values at Runtime

if user will not enter username and click on Register display error
Ily if user not enter password and click on Register display Error

The content of span tag is changing at runtime i.e modifying the content of Html ELements at runtime is called as DomManipulations



Q)what is DOM?

Document object Model

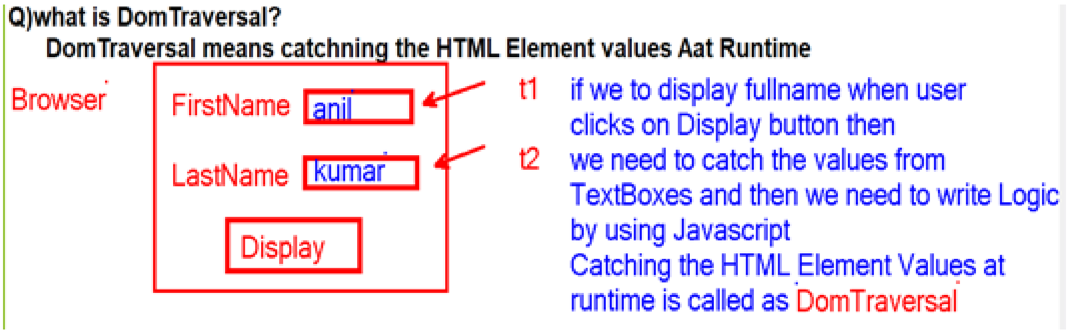
After develop any webapplication we have to deploy the appn on webserver so that enduser will access the appn via Browser and internet

user--->Browser----->url(http://localhost:1086/index.html)

Request will goto server--->webserver will search for index.html page and index.html page will load in WebBrowser

whenever the Html Page will load in Browser then a document object is created and this object is repsonsible to perform operations on Html Elements

```
<html>
 <head>
  <title>  </title>>
 </head>
 <body>
   <form>
    <input type="text" name="t1">
    <input type="text" name="t2">
   </form>
 </body>
</html>
```



Q)what is DomTraversal?

DomTraversal means catchning the HTML Element values Aat Runtime

if we to display fullname when user clicks on Display button then we need to catch the values from TextBoxes and then we need to write Logic by using Javascript

Catching the HTML Element Values at runtime is called as DomTraversal

Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

Applications:

JavaScript is used to create Interactive websites. It is mainly used for:

- client-side validation : If password and confirm passwords are empty, or some required fields left empty, then errors will be displayed.
- Dynamic drop-down menus
- Displaying date and time
- Displaying popup windows and dialog boxes.

Syllabus:-

Tokens

1. Operators

2. Variables and Datatypes

3. Typecasting

4. Conditional Statements

5. Loops

6.Arrays

7.Lambda Expresssion

8.Objects in Javascript

9.Exception handling

10. DOm Traversal

11. DOM Manipulations

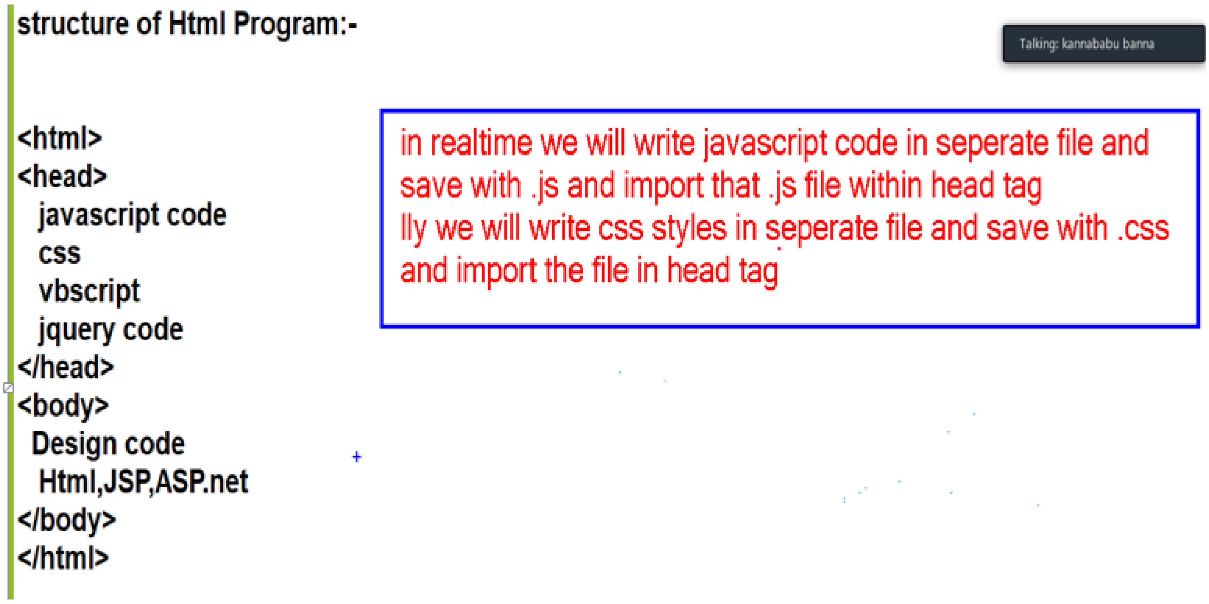
12. Arrays

13. Validations

14. Events

15. Ajax calls

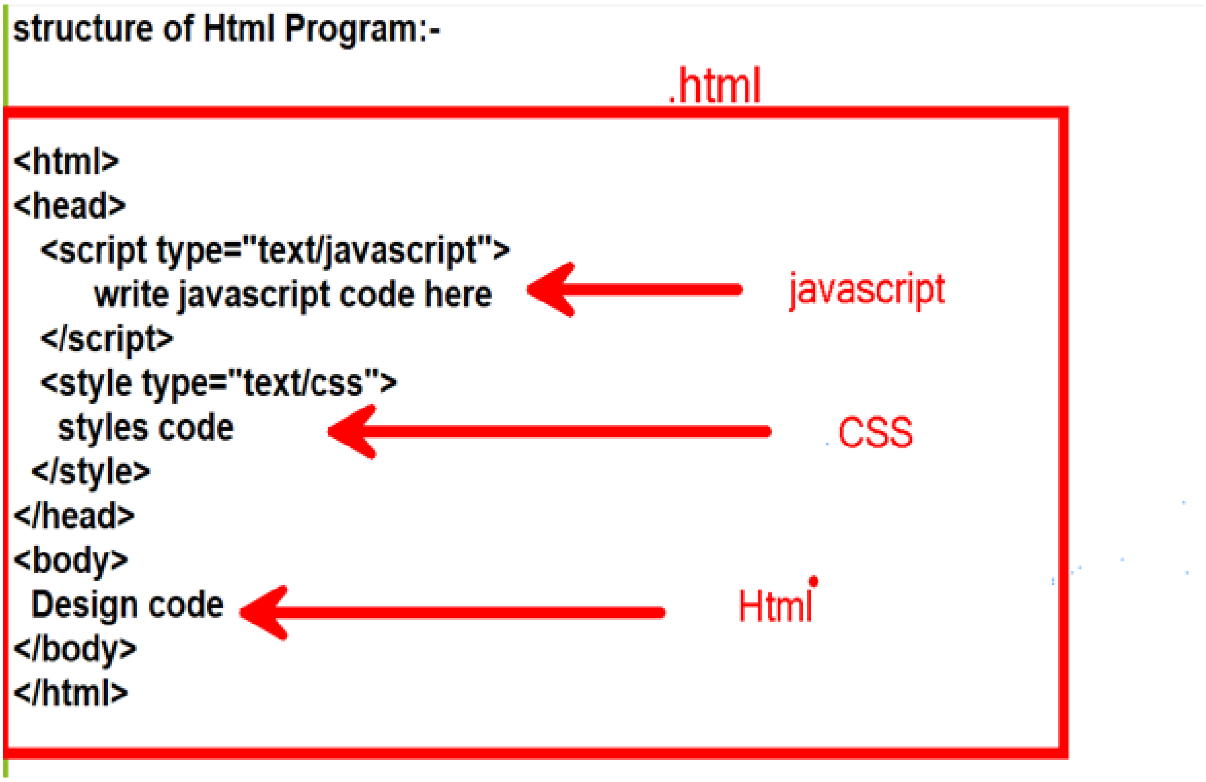
16. HTML with Javascript



structure of Html Program:-

```
<html>
<head>
  javascript code
  css
  vbscript
  jquery code
</head>
<body>
 Design code
  Html,JSP,ASP.net
</body>
</html>
```

in realtime we will write javascript code in seperate file and save with .js and import that .js file within head tag
lly we will write css styles in seperate file and save with .css and import the file in head tag



structure of Html Program:-

```
<html>
<head>
  <script type="text/javascript">
      write javascript code here
  </script>
  <style type="text/css">
    styles code
  </style>
</head>
<body>
  Design code
</body>
</html>
```

Example:

```
<html>
<head>
  <script type="text/javascript">
      document.write("welcome to Javascript");
  </script>
</head>
<body>
  </body>
</html>
```

SDLC:- Software Development Life Cycle

PDLC:- Program Development Life Cycle

PDLC is an approach which is used to develop programs fastly and efficiently

Steps to followed:-

1. problem Definition

2. Creating solution for the problem

3. Algorithm

4. Flowchart

5. PRS(Program Requirement Specification)

6. identify the Programming Concepts

7. coding

8. Testing

1. problem Definition:- identify the problem

   Q)consider 3 subject marks of the students are 70,80,90

      display total,percentage?

2. Creating solution for the problem :-

   inorder to identify the solution for the problem we have to apply Basic Mathematics

      total=70+80+90=240

      p=240/3=80.0

3. Algorithm:- stepwise refinement of a solution is algorithm

```
1. start
2. consider m1 is 70     m2 is 80     m3 is 90
3. add m1,m2,m3 and store the result in total
4.divide total with 3 and store the result in p
5. print total
6. print percentage
7. stop
```

4. Flowchart:- The diagramatical Representation of Algorithm is Flowchart

5. identify the Programming Concepts

   a. variables

   b. operators (Arthimetic operators)

6.PRS(Program Requirement Specification)

   a. identify the i/p variables that required

      m1=70,m2=80,m3=90

   b. identify the o/p variables

      total,p

   c. identify the operators

+,/,=

d. identify the operations we are performing

caltotal

calpercentage

e. expected i/p expected o/p

m1=70	Total is 240
m2=80	percentage is 80
m3=90	

7. coding

```
<script type="text/javascript">
   var m1=70;
   var m2=80;
   var m3=90;
   var total=m1+m2+m3;
   var p=total/3;
   document.writeln("Total is "+total);
   document.writeln("Percentage  is "+p);
</script>
```

8. Testing

if Expected o/p is Equal to Actual O/p then Test case is passed otherwise Failed

Ex:-

```
<html>
<head>
 <script type="text/javascript">
    var m1=70;
    var m2=80;
    var m3=90;
    var total=m1+m2+m3;
    var p=total/3;
    document.write("<h1>"+"Total is "+total+"</h1>");
```

```
      document.write("<h1>"+"Percentage  is "+p+"</h1>");
    </script>
   </head>
   </html>
```

==================

Sample Programs:

1. Consider a is 5 and b is 3, store a in b print a and b?
2. Consider a is 3 b is “abc” print a and b?
3. Consider a is 2.3 , b is 5 store sum of a,b in c print c?
4. Consider a is "sathya" b is "tech" store fullname in c print c?
5. Consider a is 2, b is 5 store a and b in c print c?
6. Consider a is 2.3f b is 3.4 store a and b in c print c?
7. Consider a is 2.3f b is 3.4 store product of a and b in c print c?
8. Consider a is ‘x’ b is ‘y’ print x,y?
9. Consider a is “10” b is 5 store a and b in c print c?
10. Consider x is 5 y is 2 x divide by y store, in z print z?
11. Consider x is 5 product of x and x store in y print y?
12. Consider x is 3 cube of x store in y print x,y?
13. Consider x is 3 square of x store in s,cube of x store in c   print s,c?
14. Consider x is 2,square of x store in s,cube of x store in c

   Sum of s and c store in sum

   Print sum?

15. Consider x is 5

   y is 3

store product of x and y store in z

sum of x,y,z store in p

print p?

JavaScript Example

1. JavaScript Example
2. Within body tag

3. Within head tag

Javascript example is easy to code. JavaScript provides 3 places to put the JavaScript code: within body tag, within head tag and external JavaScript file.

Let’s create the first JavaScript example.

```
<script type="text/javascript">
document.write("JavaScript is a simple language for javatpoint learners");
</script>
```

The **script** tag specifies that we are using JavaScript.

The **text/javascript** is the content type that provides information to the browser about the data.

The **document.write()** function is used to display dynamic content through JavaScript. We will learn about document object in detail later.

---

3 Places to put JavaScript code

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external javaScript)

---

1) JavaScript Example : code between the body tag

In the above example, we have displayed the dynamic content using JavaScript. Let’s see the simple example of JavaScript that displays alert dialog box.

```
<script type="text/javascript">
 alert("Hello Javatpoint");
</script>
```

Test it Now

---

2) JavaScript Example : code between the head tag

Let’s see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

In this example, we are creating a function msg(). To create function in JavaScript, you need to write function with function_name as given below.

To call function, you need to work on event. Here we are using onclick event to call msg() function.

```
<html>
<head>
<script type="text/javascript">
function msg(){
 alert("Hello Javatpoint");
}
</script>
</head>
<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
</html>
```

External JavaScript file

We can create external JavaScript file and embed it in many html page.

It provides **code re usability** because single JavaScript file can be used in several html pages.

An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.

Let's create an external JavaScript file that prints Hello Javatpoint in a alert dialog box.

message.js

```
function msg(){
 alert("Hello Javatpoint");
}
```

Let's include the JavaScript file into html page. It calls the JavaScript function on button click.

index.html

```
<html>
<head>
<script type="text/javascript" src="message.js"></script>
</head>
<body>
```

```
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
</html>
```

Advantages of External JavaScript

There will be following benefits if a user creates an external javascript:

1. It helps in the reusability of code in more than one HTML file.
2. It allows easy code readability.
3. It is time-efficient as web browsers cache the external js files, which further reduces the page loading time.
4. It enables both web designers and coders to work with html and js files parallelly and separately, i.e., without facing any code conflictions.
5. The length of the code reduces as only we need to specify the location of the js file.

Disadvantages of External JavaScript

There are the following disadvantages of external files:

1. The stealer may download the coder's code using the url of the js file.
2. If two js files are dependent on one another, then a failure in one file may affect the execution of the other dependent file.
3. The web browser needs to make an additional http request to get the js code.
4. A tiny to a large change in the js code may cause unexpected results in all its dependent files.
5. We need to check each file that depends on the commonly created external javascript file.
6. If it is a few lines of code, then better to implement the internal javascript code.

JavaScript Comment

1. JavaScript comments
2. Advantage of javaScript comments
3. Single-line and Multi-line comments

The **JavaScript comments** are meaningful way to deliver message. It is used to add information about the code, warnings or suggestions so that end user can easily interpret the code.

The JavaScript comment is ignored by the JavaScript engine i.e. embedded in the browser.

Advantages of JavaScript comments

There are mainly two advantages of JavaScript comments.

1. **To make code easy to understand** It can be used to elaborate the code so that end user can easily understand the code.
2. **To avoid the unnecessary code** It can also be used to avoid the code being executed. Sometimes, we add the code to perform some action. But after sometime, there may be need to disable the code. In such case, it is better to use comments.

Types of JavaScript Comments

There are two types of comments in JavaScript.

1. Single-line Comment
2. Multi-line Comment

JavaScript Single line Comment

It is represented by double forward slashes (//). It can be used before and after the statement.

Let’s see the example of single-line comment i.e. added before the statement.

```
<script>
// It is single line comment
document.write("hello javascript");
</script>
```

Test it Now

Let’s see the example of single-line comment i.e. added after the statement.

```
<script>
var a=10;
var b=20;
var c=a+b;//It adds values of a and b variable
document.write(c);//prints sum of 10 and 20
</script>
```

Test it Now

JavaScript Multi line Comment

It can be used to add single as well as multi line comments. So, it is more convenient.

It is represented by forward slash with asterisk then asterisk with forward slash. For example:

```
/* your code here  */
```

It can be used before, after and middle of the statement.

```
<script>
/* It is multi line comment.
It will not be displayed */
document.write("example of javascript multiline comment");
</script>
```

variables in Javascript:-

variable:- variable is the name given for a particular memory location

Q)what is the purpose of variable?

The purpose of variable is to store the value in javascript we can declare variable by using var keyword.

syntax to declare variable:-

var varname=value;

Q) what is Data?

Data is collection of raw facts

Data is of 2 types: Numeric and Character.

Numeric: integer (int) and floating point(float)

Character: SC   GC   ANC (char and string)

whenever we want to perform any operations on Data we need to store the data

in order.3 to store data we require variable

in order to declare variable data types is required

in javascript we can declare variable by using var keyword

var x=10;   x is int

var y=2.3;   y is float

```
  var z="abc";  z is string
```

Points to remember:-

1. we can store only one value in single variable

```
        var x=5;   valid
        var x=6,3;  Error
```

2. whenever we modify the variable value then the previous value will be erased

```
        var x=5;
               x=6;
               x=3;
    document.write(x); o/p:-   3
```

3. LeftHand side of = operator consider as variable

   Right Hand side of = operator consider as value

```
       var x=5;    x is variable     5 is value
       var y=x;               y is variable    x is 5
                        i.e we are assigning 5 in y
       document.write(y);     y is value print 5
```

```
<html>
   <head>
     <script type="text/javascript">
            var x=5;
            var y=3;
            var sum=x+y;
            var diff=x-y;
            var p=x*y;
           document.write("sum is"+sum+"<br>");
           document.write("Diff is"+diff+"<br>");
           document.write("product is"+p+"<br>");
    </script>
   </head>
   </html>
```

Javascript Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

```
var a=40;//holding number
var b="Rahul";//holding string
```

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript non-primitive data types

The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

We will have great discussion on each data type later.

Javascript Tokens:-

Tokens are programming Element which are used to construct the program

Differen Types of Tokens are:-

1. Literals
2. Constants
3. identifiers
4. Keywords
5. operators

Literal:- Literal is a value which can be modified

```
  var x=5;
      x=4;
```

Literals are Different Types:-

1. integer Literal	:- 10,200,40000
2. Floating point Literal	:- 2.3,4.5,678.99
3. charcater Literal	:- single alphabet,symbol
4. string literal	:- "abc","abc2345","234","abc1234@gmail.com"
5. boolean literal	:- true,false
6. object literal	:- {eno:101,ename:"anil",sal:20000}

Constant:- constants are literals which cannot be modified constants are immutable in javascript we declare declare constants by using const keyword

const pi=3.14;

pi=3.5;   Error

keywords:- keyword is a reserved word which have a special meaning

Ex:- var,function,if,else,for,finally,try,catch,this etc...

identifiers:-identifier is the name given for variable,object,function in javascript

Rules to be followed while declaring identifiers:-

1. identifier name must not be a number

   var 10; Error

2. identifier name must not consists of spaces

   var emp   no; Error

3. identifier name must not consists of special characters except _

   var emp_no;   valid

   var emp?no;   invalid

4. identifier name must not be a keyword

   var if;   Error

JavaScript Operators

JavaScript operators are symbols that are used to perform operations on operands.

An operator is used to perform Operation in 2 more operands

For example:

```
1. var sum=10+20;
```

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators

5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows: s:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false

!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

JavaScript Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.

,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

Arthimetic Operators are used to perform Arthimetic Operations

5+3 ---------->8

5-3   ---------->2

5*3 ----------->15

5/3 ------------>1

5%3 ------------>2

+ operator is called as overloaded operator

+operator will perform addition operation between numbers and concadination operation between strings

2+3=5	number+number=number
2+3.5= 5.5	number+number=number
"ab"+"cd"="abcd"	string+string=string
2+"ab"="2ab"	number+string=string
"2"+"3"="23"	string+string=string

Q)what is the difference between / and %?

/will gives us quotient

% will gives us remainder

```
7/2 =3.5                      Here 7 is integer   2 is integer

      2)7(3.5

        6

      --------                int/int=int

        10

        10                    7/3=3

         ---------                   7/3.0=3.5

        0

var x=7/2;                           2)7(3   Q

document.write(x);    o/p:- 3                6

                              --------

var x=7%2;                      1    R

document.write(x);    o/p:- 1
```

JavaScript If-else

The **JavaScript if-else statement** is used *to execute the code whether condition is true or false*. There are three forms of if statement in JavaScript.

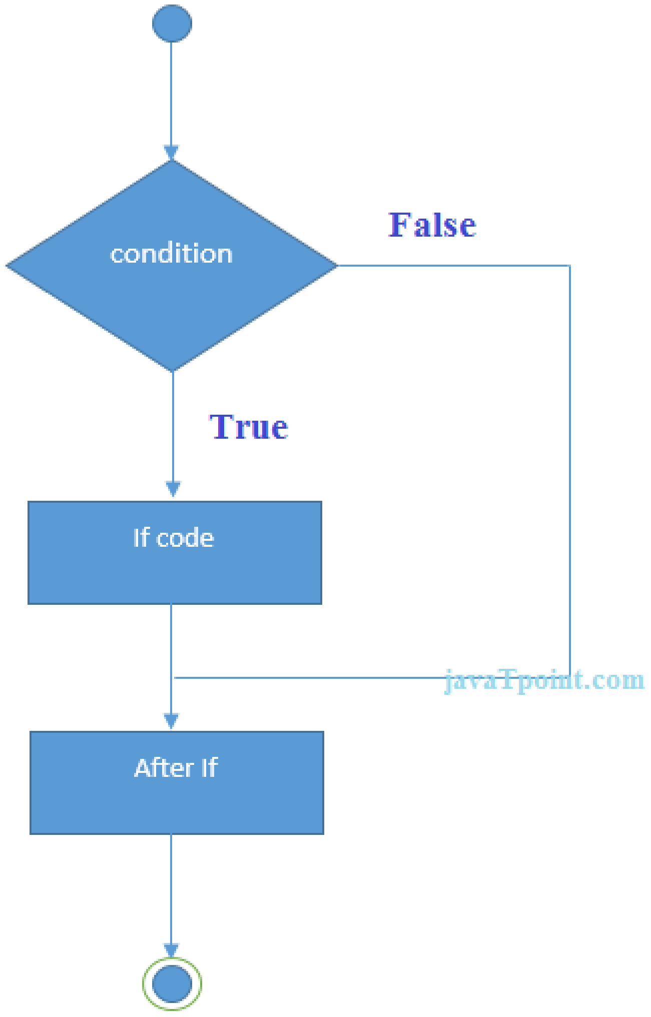
1. If Statement
2. If else statement
3. if else if statement

JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

```
if(expression){
//content to be evaluated
}
```

Flowchart of JavaScript If statement



Let’s see the simple example of if statement in javascript.

```
<script>
var a=20;
if(a>10){
document.write("value of a is greater than 10");
}
</script>
```

Test it Now

Output of the above example

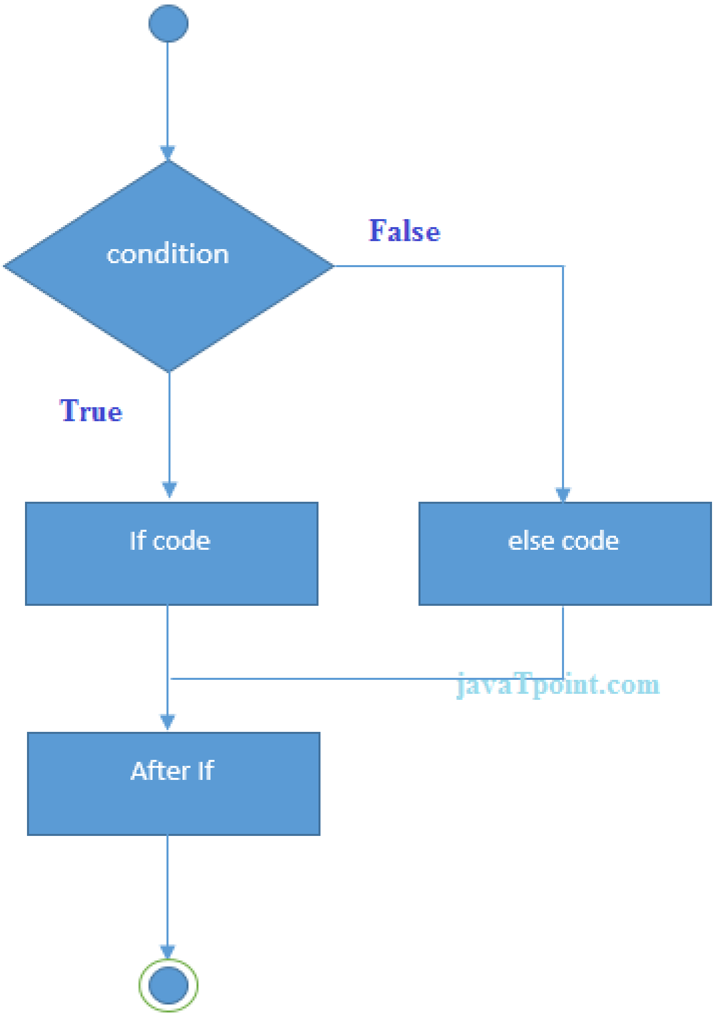
```
value of a is greater than 10
```

JavaScript If...else Statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

```
if(expression){
//content to be evaluated if condition is true
}
else{
//content to be evaluated if condition is false
}
```

Flowchart of JavaScript If...else statement



Let’s see the example of if-else statement in JavaScript to find out the even or odd number.

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

Test it Now

Output of the above example

```
a is even number
```

JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

```
if(expression1){
//content to be evaluated if expression1 is true
}
else if(expression2){
//content to be evaluated if expression2 is true
}
else if(expression3){
//content to be evaluated if expression3 is true
}
else{
//content to be evaluated if no expression is true
}
```

Let's see the simple example of if else if statement in javascript.

```
<script>
var a=20;
if(a==10){
document.write("a is equal to 10");
}
else if(a==15){
document.write("a is equal to 15");
}
else if(a==20){
document.write("a is equal to 20");
}
else{
document.write("a is not equal to 10, 15 or 20");
}
</script>
```

Test it Now

Output of the above example

```
a is equal to 20
```

Q)what is condition?

condition is used to compoare 2 values or variables or expressions and return a boolean value either true or false

Conditional operators or Relational Operators:-

<	Lessthan	5<3	False
>	Greaterthan	5>3	True
<=	Lessthan or Equal	5<=5	True
>=	Greaterthan or Equal	5>=5	True
==	Equal	5==5	True
!=	Not Equal	5!=5	False

Q)consider a consutmer purchasing product in amazon through online

if the cust totalbill >4000 amazon annunce 20% discount on tbill?

2000,3000,1000,4000

2000+3000+1000+4000

=10000

check whether tbill >10000 then give 20% discount

10000*20/100=2000

disamt --->2000

10000-2000=8000/-

Conditional Statements :- The statements that will gets executed based on the result of the conditon

only if

if-else

else-if

multiple if

nested if

===============

only if:- The statements that will executed when the condition is true

```
  if(true)                              if(10>5)
  {                              document.write("welcome");
        statements
  }
```

if-else:- it is used to check the condition and return either true or false

if the condition result is true then the statements that was written inside if will gets executed

otheriwse the the statements that was written inside else will gets executed

```
if(true)        Ex:-
{               if(10<5)
                {
}                 hi
else            }
{               else
                {
}                 Bye
                }
```

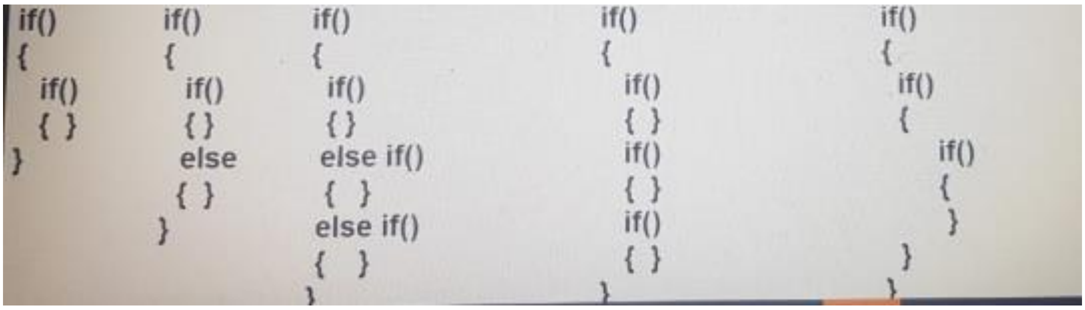
else if:- it is used to check multiple conditions and execute only one condition

```
if()
{
}
else if()
{
}
else if()
{
}
else if()
{
}
```

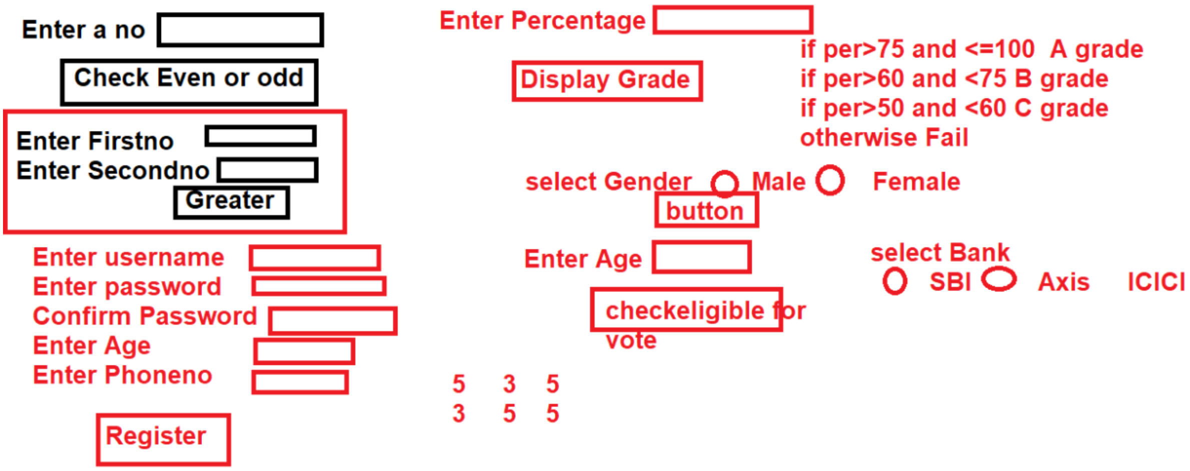
multiple if:- it is used to check multiple conditions and execute multiple conditions

```
if()
{  }
if()
{  }
if()
{  }
if()
{  }
```

nested if:- The inner condition will gets executed when the outer condition is true



```
if()        if()        if()             if()        if()
{           {           {                {           {
  if()        if()        if()             if()        if()
  { }         { }         { }              { }         {
}             else        else if()        if()          if()
              { }         {  }             { }           {
            }             else if()        if()          }
                          {  }             { }         }
                        }                }           }
```



if per>75 and <=100 A grade
if per>60 and <75 B grade
if per>50 and <60 C grade
otherwise Fail

we can catch Html Element Values at runtime in 2 ways :-

1. formname formname.ElementName.value
2. Element ID document.getElementById("elementid").value;

===
===========

validation:-it is a process validating the user i/p before submitting request to server

```
if()   if()   if()     if()     if()     if()     if()     if()
{      {        {      {        {        {        {        {
}      }        }      }          if()     if()     if()     if()
else    else if()          if()       {          {          { }       {  }
{         {       {          }          }            else if()    if()
}         }       }        }          else       { }          {  }
        else if()  if()                 {        else if()   if()
        {          {                    }        { }        {  }
        }          }                  }          else if()   if()
                                                 { }        { }
                                               }          }
```

JavaScript Switch

The **JavaScript switch statement** is used *to execute one code from multiple expressions*. It is just like else if statement that we have learned in previous page. But it is convenient than *if..else..if* because it can be used with numbers, characters etc.

The signature of JavaScript switch statement is given below.

```
switch(expression){
case value1:
 code to be executed;
 break;
case value2:
 code to be executed;
 break;
......

default:
 code to be executed if above values are not matched;
}
```

Let’s see the simple example of switch statement in javascript.

```
<script>
var grade='B';
var result;
switch(grade){
case 'A':
result="A Grade";
break;
case 'B':
result="B Grade";
break;
case 'C':
```

```
result="C Grade";
break;
default:
result="No Grade";
}
document.write(result);
</script>
```

Test it Now

Output of the above example

```
B Grade
```

> ***The switch statement is fall-through i.e. all the cases will be evaluated if you don't use break statement.***

Let’s understand the behaviour of switch statement in JavaScript.

```
<script>
var grade='B';
var result;
switch(grade){
case 'A':
result+=" A Grade";
case 'B':
result+=" B Grade";
case 'C':
result+=" C Grade";
default:
result+=" No Grade";
}
document.write(result);
</script>
```

Test it Now

Output of the above example

```
undefined B Grade C Grade No Grade
```

JavaScript Loops

The **JavaScript loops** are used *to iterate the piece of code* using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

1) JavaScript For loop

The **JavaScript for loop** *iterates the elements for the fixed number of times*. It should be used if number of iteration is known. The syntax of for loop is given below.

```
for (initialization; condition; increment)
{
    code to be executed
}
```

Let’s see the simple example of for loop in javascript.

```
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
```

Test it Now

Output:

```
1
2
3
4
5
```

2) JavaScript while loop

The **JavaScript while loop** *iterates the elements for the infinite number of times*. It should be used if number of iteration is not known. The syntax of while loop is given below.

```
while (condition)
{
    code to be executed
}
```

Let’s see the simple example of while loop in javascript.

```
<script>
var i=11;
while (i<=15)
{
document.write(i + "<br/>");
i++;
}
</script>
```

Test it Now

Output:

```
11
12
13
14
15
```

3) JavaScript do while loop

The **JavaScript do while loop** *iterates the elements for the infinite number of times* like while loop. But, code is *executed at least* once whether condition is true or false. The syntax of do while loop is given below.

```
do{
    code to be executed
}while (condition);
```

Let’s see the simple example of do while loop in javascript.

```
<script>
var i=21;
do{
document.write(i + "<br/>");
i++;
}while (i<=25);
</script>
```

Test it Now

Output:

```
21
22
23
24
25
```

4) JavaScript for in loop

The **JavaScript for in loop** is used *to iterate the properties of an object*. We will discuss about it later.

===

JavaScript Functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

Function is a subprogram which is used to perform a specific operation.function will gets executed when we call it. function is used to write some Logic. single function is used to perform single operation. It is always recomended to declare the function in head tag and

calling of the function in body tag.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don’t need to write many lines of code each time to perform a common task.

---

syntax to declare function:-

```
function functionname()
{
  write some logic here
}
```

syntax to call the function:-

```
  functionname();
```

Ex:-

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
```

```
  function f1()
  {
     document.write("i am function ");
  }
   f1();
</script>
</body>
</html>
```

Function will have Parameters:-

At the time of declaration of function if we declare parameters then at the time of calling the function we have to pass values

The no of ,order of,type of values that we pass must match with no of,order of,type of parameters

syn:-

```
 function funname(x,y)
{


}
funname(6,4);
```

Function with Return Value

We can call function that returns a value and use it in our program. Let’s see the example of function that returns value.

```
<script>
function getInfo(){
return "hello javatpoint! How r u?";
}
</script>
<script>
document.write(getInfo());
</script>
```

Output of the above example

```
hello javatpoint! How r u?
```

Ex for function which will return value:-

```
function f1(x,y)
{
 return x+y;
}
```

if the function will return the value at the time of calling the function we have to store the result of the function in variable:-

```
var sum=f1(6,3);
document.write(sum);
```

Q)Advantage of function?

Reusability

instead of repeatedly same code we can reuse code

Q)what is Local Variable?

the variable that was declared inside the function is called as Local variable

```
      function f1()
      {
        var x=5;
       document.write(x);    valid
      }
```

note:- if we declare variable inside thje function and that variable cannot be accessable outside the function

```
      function f1()
      {
        var x=5;
      }
       document.write(x);    Error
```

Global variable:- The variable that was declared outside the function is called as Global variables

The scope of Global variable in multiple functions

```
    <script>
         var x=5;   global variables
```

```
     var y=4;
 function Add()
 {
    var sum=x+y;
  }
 function Sub()
 {
    var diff=x-y;
 }
</script>
```

Q)what is variable declaration?

declaring the variable without assigning the value

```
 var x;
```

Q)what is variable initialization?

Assigning the value to the variable at the time of declaration

```
 var x=5;
```

Q)what is variable Assignment?

Assigning the value to the variable after declaration

```
var x;
  x=5;
```

note:-

1. we can declare global variable outside the function and we can initialize values inside the function

```
var x;                              variable decration
var y;
function f1()
{
  x=5;                              assignment
  y=3;
}
function f1()
```

```
{
  x=4;                                    assignment
  y=2;
}
```

JavaScript Function Object

In JavaScript, the purpose of **Function constructor** is to create a new Function object. It executes the code globally. However, if we call the constructor directly, a function is created dynamically but in an unsecured way.

Syntax

```
1. new Function ([arg1[, arg2[, ....argn]],] functionBody)
```

Parameter

arg1, arg2, ... , argn - It represents the argument used by function.

functionBody - It represents the function definition.

JavaScript Function Methods

Let's see function methods with description.

Method	Description
apply()	It is used to call a function contains this value and a single array of arguments.
bind()	It is used to create a new function.
call()	It is used to call a function contains this value and an argument list.
toString()	It returns the result in a form of a string.

JavaScript Function Object Examples

Example 1

Let's see an example to display the sum of given numbers.

```
<script>
var add=new Function("num1","num2","return num1+num2");
document.writeln(add(2,5));
</script>
```

Test it Now

Output:

```
7
```

Example 2

Let's see an example to display the power of provided value.

```
<script>
var pow=new Function("num1","num2","return Math.pow(num1,num2)");
document.writeln(pow(2,3));
</script>
```

Test it Now

Output:

```
8
```

Functions are of 3 Types:-

1. Named Functions

2. Anonymous Functions

3. Arrow Functions

Named Functions:- Declaring the function with some name

function will gets executed when we call it

Ex:-

```
function GetSum(x,y)
{
    document.write(x+y);
}
```

syn to call the function:-

funname(values);

```
GetSum(6,3);
```

Anonymous Functions:- Anonymous Functions are also called as nameless functions

i.e declaring a function without any name is called as Anonymous Function

```
 var x= function()
            {
               document.write("i am function");
            }
```

invoke the function:-

```
    x();
```

Ex to create Anonymous function with parameters:-

```
    var sum=function(x,y)
             {
                      document.write(x+y);
             }
    sum(6,3);
```

Q)what is variable?

variable is the name given for a particular memory location

Q)what is the purpose of variable?

The purpose of variable is to store the value

we can assign value,expression,functioncall,anonymousfunction to variable

```
    var x=10;               assigning value to variable
    var y=5-3+2*3;                  assigning Expression to variable
    --------------------
    function GetSum(x,y)
    {
       return x+y;
    }
    var sum=GetSum(6,3);        The value 9  was assigned to sum
    ========================
     function GetTotal(m1,m2,m3)
```

```
   {
      return m1+m2+m3;
   }
  function GetPer()
  {
    var t=GetTotal(70,70,70);
     var p=t/3;
  }
```

======================

we can assign anonymous function to a variable

```
   var sum =function(x,y)                var sum =function(x,y)
    {                               {
       console.log(x+y);                        return x+y;
    }                                }
   sum(6,3);    o/p:=   9                sum(6,3);     o/p:- no output
=====================                        var s=sum(6,3);   The value 9 is stored in s
                                document.write(sum(6,3));  o/p is 9
```

Q)what is the difference between document.write() and console.log()?

document.write() is used to display the o/p on Browser

console.log() is used to display the o/p on console

Arrrow Functions:-

Arrow function is the shorthand syntax of Anonymous function

syn:-

```
()=> logic;
lambda operator (or) goes to
i/p=> logic
```

Arrow functions are of 2 types:-

1. single line functions

```
   var x=function()
    {
      document.write("welcome sathya");
     }
    -------------------------------
    var x=()=>console.log("welcome sathya");
==================================================
```

2. multi line function

```
     var x=function()
     {
       console.log("welcome sathya");
        console.log("Ameerpet");
      }
   ==================
    var x=()=>
    {
       console.log("welcome sathya");
        console.log("Ameerpet");
    }
=====================
```

Ex:-

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
 var x=()=>document.write("welcome to Arrowfunction"+"<br>");
    x();              ------------->o/p is welcome to Arrow function
 var sum=(a,b)=>document.write("sum is"+(a+b)+"<br>");
  sum(6,3);                  -------------->    sum is 9
 var product=(k,l)=> k*l;
  var p=product(6,3);
```

```
    document.write(p);        ----------> 18
</script>
</body>
</html>
========================
<html>
 <head>
<script type="text/javascript">
  var per=(m1,m2,m3)=>
        {
            var t=m1+m2+m3;
            document.write(t/3);
        }
</script>
 </head>
 <body>
<script type="text/javascript">
     per(80,80,80);
</script>
 </body>
</html>
```

JavaScript Objects

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.

JavaScript is an object-based language. Everything is an object in JavaScript.

JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

Creating Objects in JavaScript

There are 3 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

1) JavaScript Object by object literal

The syntax of creating object using object literal is given below:

```
object={property1:value1,property2:value2.....propertyN:valueN}
```

As you can see, property and value is separated by : (colon).

Let’s see the simple example of creating object in JavaScript.

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

Test it Now

Output of the above example

```
102 Shyam Kumar 40000
```

2) By creating instance of Object

The syntax of creating object directly is given below:

```
var objectname=new Object();
```

Here, **new keyword** is used to create object.

Let’s see the example of creating object directly.

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

Test it Now

Output of the above example

```
101 Ravi 50000
```

3) By using an Object constructor

Here, you need to create function with arguments. Each argument value can be assigned in the current object by using this keyword.

The **this keyword** refers to the current object.

The example of creating object by object constructor is given below.

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

Test it Now

Output of the above example

```
103 Vimal Jaiswal 30000
```

Defining method in JavaScript Object

We can define method in JavaScript object. But before defining method, we need to add property in the function with same name as method.

The example of defining method in object is given below.

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;

this.changeSalary=changeSalary;
function changeSalary(otherSalary){
this.salary=otherSalary;
}
}
e=new emp(103,"Sonoo Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
e.changeSalary(45000);
document.write("<br>"+e.id+" "+e.name+" "+e.salary);
</script>
```

Test it Now

Output of the above example

```
103 Sonoo Jaiswal 30000
103 Sonoo Jaiswal 45000
```

JavaScript Object Methods

The various methods of Object are as follows:

S.No	Methods	Description
1	Object.assign()	This method is used to copy enumerable and own properties from a source object to a target object
2	Object.create()	This method is used to create a new object with the specified prototype object and properties.
3	Object.defineProperty()	This method is used to describe some behavioral attributes of the property.
4	Object.defineProperties()	This method is used to create or configure multiple object properties.
5	Object.entries()	This method returns an array with arrays of the key, value pairs.
6	Object.freeze()	This method prevents existing properties from being removed.
7	Object.getOwnPropertyDescriptor()	This method returns a property descriptor for the specified property of the specified object.

8	Object.getOwnPropertyDescriptors()	This method returns all own property descriptors of a given object.
9	Object.getOwnPropertyNames()	This method returns an array of all properties (enumerable or not) found.
10	Object.getOwnPropertySymbols()	This method returns an array of all own symbol key properties.
11	Object.getPrototypeOf()	This method returns the prototype of the specified object.
12	Object.is()	This method determines whether two values are the same value.
13	Object.isExtensible()	This method determines if an object is extensible
14	Object.isFrozen()	This method determines if an object was frozen.
15	Object.isSealed()	This method determines if an object is sealed.
16	Object.keys()	This method returns an array of a given object's own property names.
17	Object.preventExtensions()	This method is used to prevent any extensions of an object.
18	Object.seal()	This method prevents new properties from being added and marks all existing properties as non-configurable.
19	Object.setPrototypeOf()	This method sets the prototype of a specified object to another object.
20	Object.values()	This method returns an array of values.

objects in javascript:-

variable:- variable is the name given for a particular memory location

purpose of variable is to store the value

object:- object is used to hold group of values

object consists of data

syn to store the values in object:-

```
   var objectname={varname:value,varname:value};
    var e1={eno:101,ename:"anil",sal:20000};
```

Every object consists of variables and functions

Q)what is object Reference?

object Reference is the name given for object

Every object will have 2 References

1. Default reference   this
2. userdefined reference

Q)what is this?

this is the default reference variable given for the object

Q) what is the purpose of this?

the purpose of this is to access variables

Q)what is the purpose of userdefined reference ?

The purpose of userdefined reference is to access functions

```
<html>
<head>
</head>
<body>
  <script type="text/javascript">
       var e1={ eno:101,ename:"anil",salary:20000 };
        document.write(e1.eno);
        document.write(e1.ename);
        document.write(e1.salary);
  </script>
```

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

Ex:-

```
<html>
<body>
<script>
// Create an object:
var person = {
  firstName: "Anil",
  lastName : "Kumar",
  id     : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName+" "+"Personid is "+this.id;
  }
};
// Display data from the object:
document.write(person.fullName());
</script>
</body>
</html>
================
var e1={Eno:101,Ename:"anil",Salary:20000};
var e2={Eno:102,Ename:"sunil",Salary:23000};
document.write(e1.Eno+e1.Ename+e1.Salary);
document.write(e2.Eno+e2.Ename+e2.Salary);
```

Objects are of 2 types

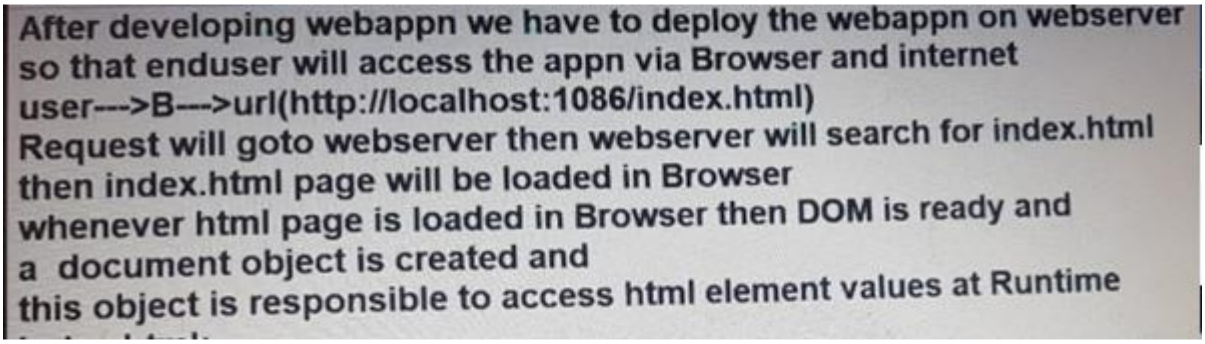
Predefined		userdefined
window	document	var e1={eno:101.ename:"anil",sal:20000 };
alert()	write()	
confirm()	getElementById()	

prompt() innerHTML

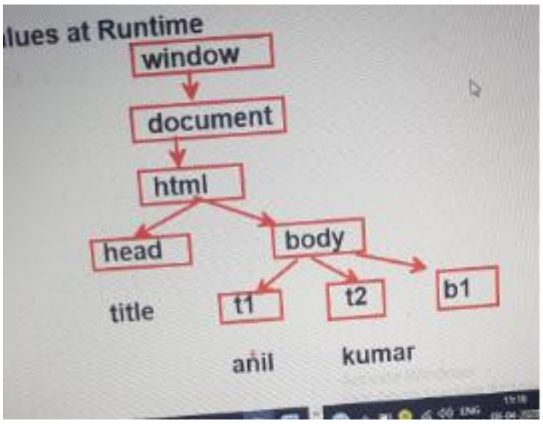
getElementByName()

we can access predefined functions by using objectname:-

window.alert("hi welcome");   :- it is used to display message box in Browser



After developing webappn we have to deploy the webappn on webserver
so that enduser will access the appn via Browser and internet
user--->B--->url(http://localhost:1086/index.html)
Request will goto webserver then webserver will search for index.html
then index.html page will be loaded in Browser
whenever html page is loaded in Browser then DOM is ready and
a document object is created and
this object is responsible to access html element values at Runtime

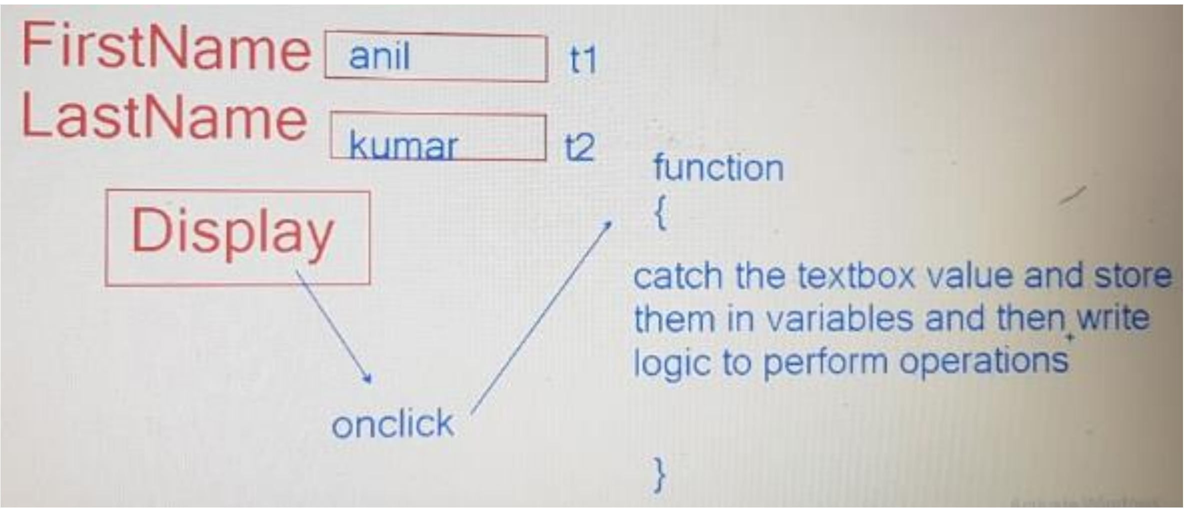


Q)What is DOM Traversal?

Catching the HTML Element values at runtime is called as Dom Trsaversal.

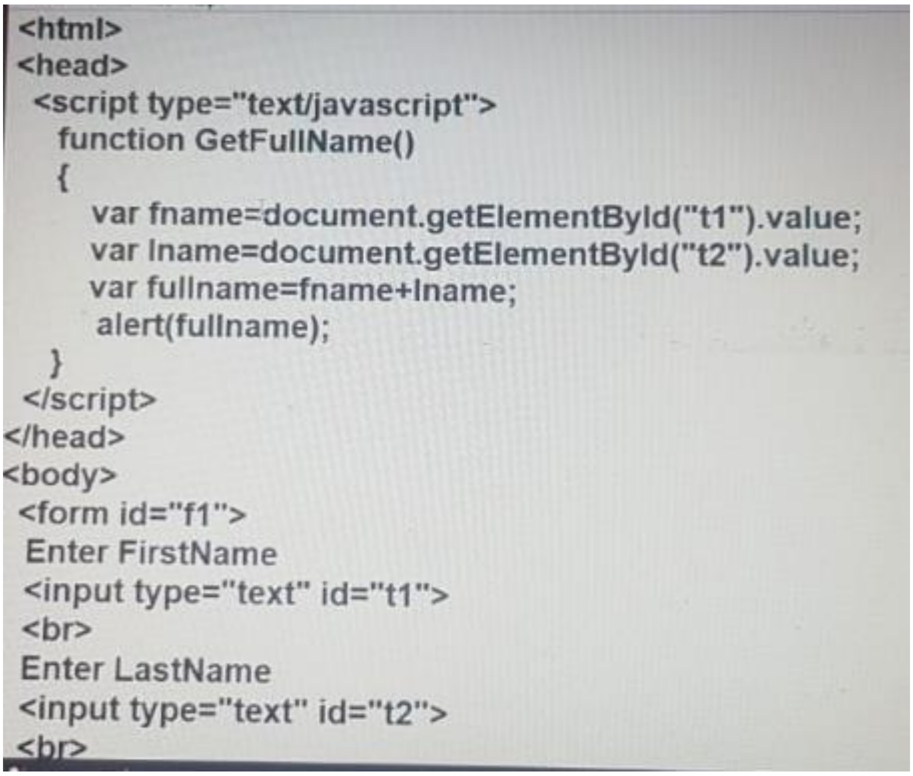
Q) What is DOM Manipulation?

Modifying of Html Element attributes at runtime.

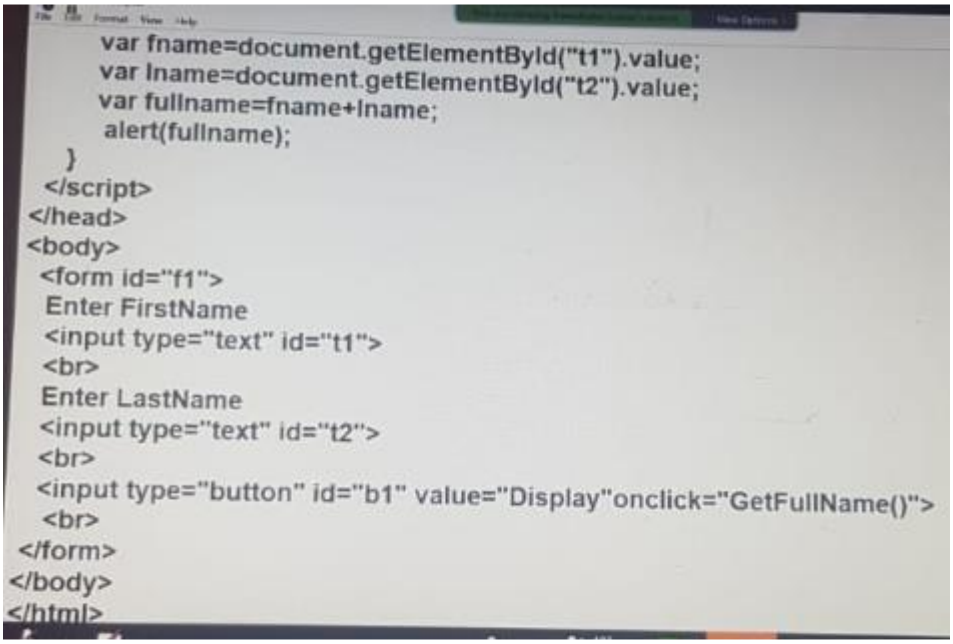

FirstName
anil
t1
LastName
kumar
t2
Display
onclick
function
{
catch the textbox value and store them in variables and then write logic to perform operations
}

Syntax to catch HTML Element values atruntime based on id:

document.getElementById("Elementid").value;

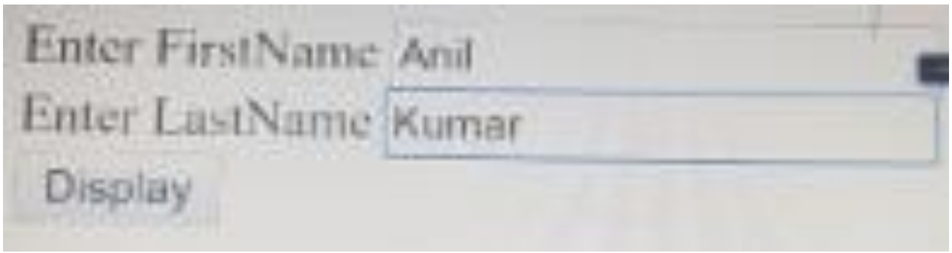


```
<html>
<head>
 <script type="text/javascript">
   function GetFullName()
   {
       var fname=document.getElementById("t1").value;
       var lname=document.getElementById("t2").value;
       var fullname=fname+lname;
        alert(fullname);
   }
 </script>
</head>
<body>
 <form id="f1">
  Enter FirstName
  <input type="text" id="t1">
  <br>
  Enter LastName
  <input type="text" id="t2">
  <br>
```



```
        var fname=document.getElementById("t1").value;
        var lname=document.getElementById("t2").value;
        var fullname=fname+lname;
         alert(fullname);
    }
  </script>
</head>
<body>
  <form id="f1">
   Enter FirstName
   <input type="text" id="t1">
   <br>
  Enter LastName
  <input type="text" id="t2">
  <br>
 <input type="button" id="b1" value="Display"onclick="GetFullName()">
  <br>
 </form>
</body>
</html>
```

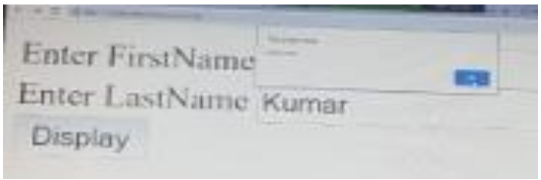
Output:



Enter FirstName Anil

Enter LastName Kumar

Output when we click on display a dialog box is displayed.



Enter FirstName

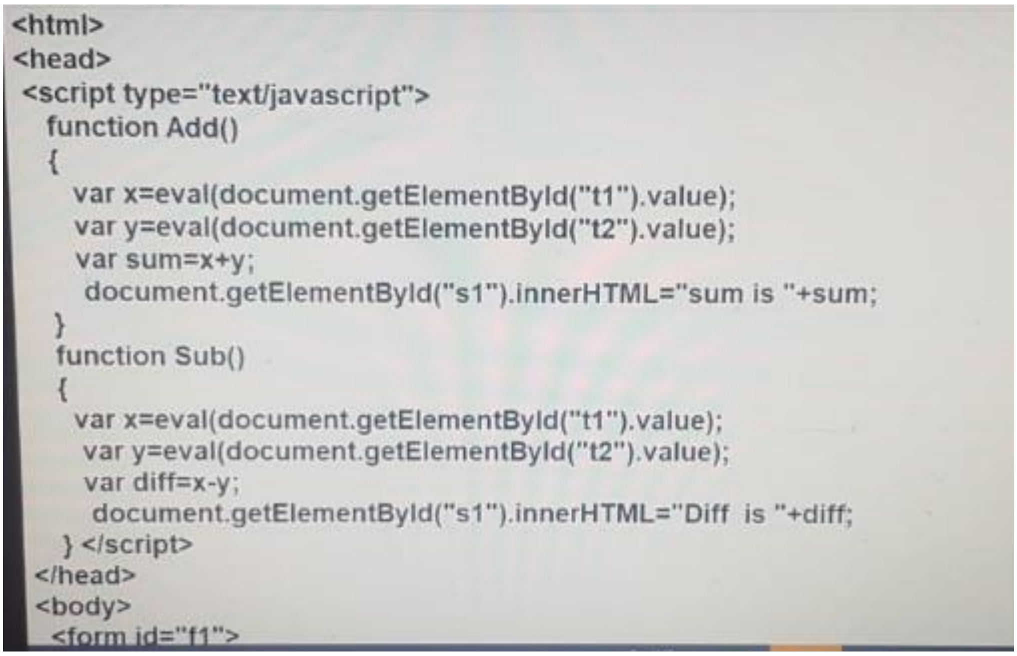
Enter LastName Kumar

The Text displayed in dialog is (Its typed here bcoz not clear in pic)

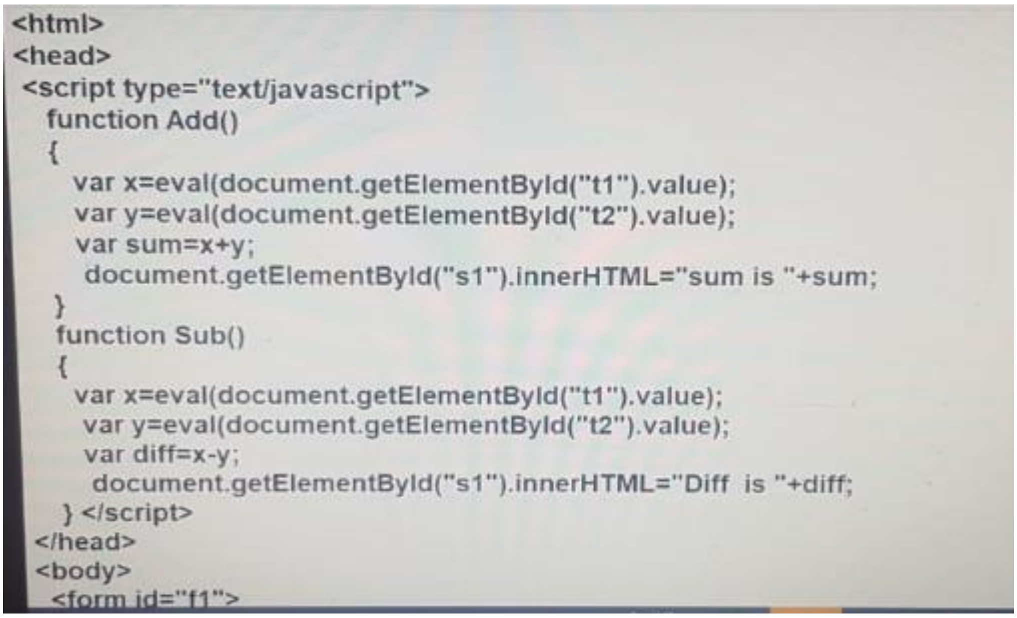
“The page says

 Anil Kumar”

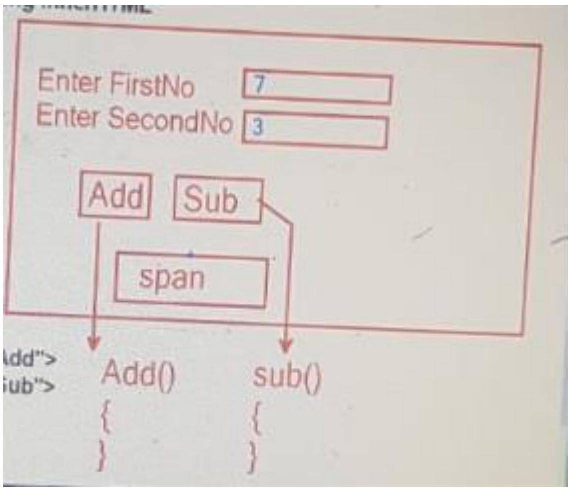
InnerHTML: InnerHtml   is a property of document object. It is used to modify the content of HTML Element values at runtime. We can achieve Dom Manipulation by using innerHTML.



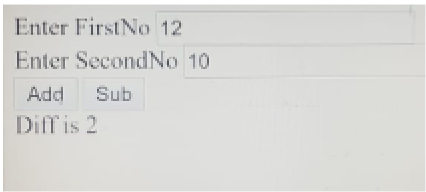
```
<html>
<head>
 <script type="text/javascript">
   function Add()
   {
      var x=eval(document.getElementById("t1").value);
      var y=eval(document.getElementById("t2").value);
      var sum=x+y;
       document.getElementById("s1").innerHTML="sum is "+sum;
    }
    function Sub()
    {
      var x=eval(document.getElementById("t1").value);
       var y=eval(document.getElementById("t2").value);
       var diff=x-y;
        document.getElementById("s1").innerHTML="Diff  is "+diff;
     } </script>
  </head>
  <body>
    <form id="f1">
```



```
<html>
<head>
 <script type="text/javascript">
   function Add()
   {
      var x=eval(document.getElementById("t1").value);
      var y=eval(document.getElementById("t2").value);
      var sum=x+y;
       document.getElementById("s1").innerHTML="sum is "+sum;
    }
    function Sub()
    {
      var x=eval(document.getElementById("t1").value);
       var y=eval(document.getElementById("t2").value);
       var diff=x-y;
        document.getElementById("s1").innerHTML="Diff  is "+diff;
     } </script>
  </head>
  <body>
    <form id="f1">
```



Output:



Enter FirstNo 12

Enter SecondNo 10

Diff is 2

JavaScript Array

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

1) JavaScript array literal

The syntax of creating array using array literal is given below:

```
var arrayname=[value1,value2.....valueN];
```

As you can see, values are contained inside [ ] and separated by , (comma).

Let's see the simple example of creating and using array in JavaScript.

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

Test it Now

The .length property returns the length of an array.

Output of the above example

```
Sonoo
Vimal
Ratan
```

2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

```
var arrayname=new Array();
```

Here, **new keyword** is used to create instance of array.

Let's see the example of creating array directly.

```
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Test it Now

Output of the above example

```
Arun
Varun
John
```

3) JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

The example of creating object by array constructor is given below.

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Test it Now

Output of the above example

```
Jai
Vijay
Smith
```

JavaScript Array Methods

Let's see the list of JavaScript array methods with their description.

Methods	Description
concat()	It returns a new array object that contains two or more merged arrays.
copywithin()	It copies the part of the given array with its own elements and returns the modified array.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
filter()	It returns the new array containing the elements that pass the provided function conditions.

find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
keys()	It creates an iterator object that contains only the keys of the array, then loops through these keys.
lastIndexOf()	It searches the specified element in the given array and returns the index of the last match.
map()	It calls the specified function for every array element and returns the new array
of()	It creates a new array from a variable number of arguments, holding any type of argument.
pop()	It removes and returns the last element of an array.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.

reduce(function, initial)	It executes a provided function for each value from left to right and reduces the array to a single value.
reduceRight()	It executes a provided function for each value from right to left and reduces the array to a single value.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.
toLocaleString()	It returns a string containing all the elements of a specified array.
toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.

JavaScript String

The **JavaScript string** is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

1. By string literal
2. By string object (using new keyword)

1) By string literal

The string literal is created using double quotes. The syntax of creating string using string literal is given below:

```
var stringname="string value";
```

Let's see the simple example of creating string literal.

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

Test it Now

Output:

```
This is string literal
```

2) By string object (using new keyword)

The syntax of creating string object using new keyword is given below:

```
var stringname=new String("string literal");
```

Here, **new keyword** is used to create instance of string.

Let's see the example of creating string in JavaScript by new keyword.

```
<script>
var stringname=new String("hello javascript string");
document.write(stringname);
</script>
```

Test it Now

Output:

```
hello javascript string
```

JavaScript String Methods

Let's see the list of JavaScript string methods with examples.

Methods	Description

charAt()	It provides the char value present at the specified index.
charCodeAt()	It provides the Unicode value of a character present at the specified index.
concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndexOf()	It provides the position of a char value present in the given string by searching a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.
match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.
toLocaleLowerCase()	It converts the given string into lowercase letter on the basis of host?s current locale.
toUpperCase()	It converts the given string into uppercase letter.

toLocaleUpperCase()	It converts the given string into uppercase letter on the basis of host?s current locale.
toString()	It provides a string representing the particular object.
valueOf()	It provides the primitive value of string object.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.

1) JavaScript String charAt(index) Method

The JavaScript String charAt() method returns the character at the given index.

```
<script>
var str="javascript";
document.write(str.charAt(2));
</script>
```

Test it Now

Output:

```
v
```

2) JavaScript String concat(str) Method

The JavaScript String concat(str) method concatenates or joins two strings.

```
<script>
var s1="javascript ";
var s2="concat example";
var s3=s1.concat(s2);
document.write(s3);
</script>
```

Test it Now

Output:

```
javascript concat example
```

3) JavaScript String indexOf(str) Method

The JavaScript String indexOf(str) method returns the index position of the given string.

```
<script>
var s1="javascript from javatpoint indexof";
var n=s1.indexOf("from");
document.write(n);
</script>
```

Test it Now

Output:

```
11
```

4) JavaScript String lastIndexOf(str) Method

The JavaScript String lastIndexOf(str) method returns the last index position of the given string.

```
<script>
var s1="javascript from javatpoint indexof";
var n=s1.lastIndexOf("java");
document.write(n);
</script>
```

Test it Now

Output:

```
16
```

5) JavaScript String toLowerCase() Method

The JavaScript String toLowerCase() method returns the given string in lowercase letters.

```
<script>
var s1="JavaScript toLowerCase Example";
var s2=s1.toLowerCase();
document.write(s2);
</script>
```

Test it Now

Output:

```
javascript tolowercase example
```

6) JavaScript String toUpperCase() Method

The JavaScript String toUpperCase() method returns the given string in uppercase letters.

```
<script>
var s1="JavaScript toUpperCase Example";
var s2=s1.toUpperCase();
document.write(s2);
</script>
```

Test it Now

Output:

```
JAVASCRIPT TOUPPERCASE EXAMPLE
```

7) JavaScript String slice(beginIndex, endIndex) Method

The JavaScript String slice(beginIndex, endIndex) method returns the parts of string from given beginIndex to endIndex. In slice() method, beginIndex is inclusive and endIndex is exclusive.

```
<script>
var s1="abcdefgh";
var s2=s1.slice(2,5);
document.write(s2);
</script>
```

Test it Now

Output:

```
cde
```

8) JavaScript String trim() Method

The JavaScript String trim() method removes leading and trailing whitespaces from the string.

```
<script>
var s1="     javascript trim     ";
var s2=s1.trim();
document.write(s2);
</script>
```

Test it Now

Output:

```
javascript trim
```

9) JavaScript String split() Method

```
<script>
```

```
var str="This is JavaTpoint website";
document.write(str.split(" ")); //splits the given string.
</script>
```

JavaScript Date Object

The **JavaScript date** object can be used to get year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

You can use different Date constructors to create date object. It provides methods to get and set day, month, year, hour, minute and seconds.

Constructor

You can use 4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

JavaScript Date Methods

Let's see the list of JavaScript date methods with their description.

Methods	Description
getDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of local time.
getDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of local time.
getFullYears()	It returns the integer value that represents the year on the basis of local time.
getHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of local time.

getMilliseconds()	It returns the integer value between 0 and 999 that represents the milliseconds on the basis of local time.
getMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of local time.
getMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of local time.
getSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of local time.
getUTCDate()	It returns the integer value between 1 and 31 that represents the day for the specified date on the basis of universal time.
getUTCDay()	It returns the integer value between 0 and 6 that represents the day of the week on the basis of universal time.
getUTCFullYears()	It returns the integer value that represents the year on the basis of universal time.
getUTCHours()	It returns the integer value between 0 and 23 that represents the hours on the basis of universal time.
getUTCMinutes()	It returns the integer value between 0 and 59 that represents the minutes on the basis of universal time.
getUTCMonth()	It returns the integer value between 0 and 11 that represents the month on the basis of universal time.
getUTCSeconds()	It returns the integer value between 0 and 60 that represents the seconds on the basis of universal time.
setDate()	It sets the day value for the specified date on the basis of local time.
setDay()	It sets the particular day of the week on the basis of local time.

setFullYears()	It sets the year value for the specified date on the basis of local time.
setHours()	It sets the hour value for the specified date on the basis of local time.
setMilliseconds()	It sets the millisecond value for the specified date on the basis of local time.
setMinutes()	It sets the minute value for the specified date on the basis of local time.
setMonth()	It sets the month value for the specified date on the basis of local time.
setSeconds()	It sets the second value for the specified date on the basis of local time.
setUTCDate()	It sets the day value for the specified date on the basis of universal time.
setUTCDay()	It sets the particular day of the week on the basis of universal time.
setUTCFullYears()	It sets the year value for the specified date on the basis of universal time.
setUTCHours()	It sets the hour value for the specified date on the basis of universal time.
setUTCMilliseconds()	It sets the millisecond value for the specified date on the basis of universal time.
setUTCMinutes()	It sets the minute value for the specified date on the basis of universal time.
setUTCMonth()	It sets the month value for the specified date on the basis of universal time.

setUTCSeconds()	It sets the second value for the specified date on the basis of universal time.
toDateString()	It returns the date portion of a Date object.
toISOString()	It returns the date in the form ISO format string.
toJSON()	It returns a string representing the Date object. It also serializes the Date object during JSON serialization.
toString()	It returns the date in the form of string.
toTimeString()	It returns the time portion of a Date object.
toUTCString()	It converts the specified date in the form of string using UTC time zone.
valueOf()	It returns the primitive value of a Date object.

JavaScript Date Example

Let's see the simple example to print date object. It prints date and time both.

```
Current Date and Time: <span id="txt"></span>
<script>
var today=new Date();
document.getElementById('txt').innerHTML=today;
</script>
```

Test it Now

Output:

```
Current Date and Time: Sun Apr 12 2020 10:32:55 GMT+0530 (India Standard
Time)
```

Let's see another code to print date/month/year.

```
<script>
var date=new Date();
var day=date.getDate();
var month=date.getMonth()+1;
var year=date.getFullYear();
document.write("<br>Date is: "+day+"/"+month+"/"+year);
</script>
```

Output:

```
Date is: 12/4/2020
```

JavaScript Current Time Example

Let's see the simple example to print current time of system.

```
Current Time: <span id="txt"></span>
<script>
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
</script>
```

Test it Now

Output:

```
Current Time: 10:32:55
```

JavaScript Digital Clock Example

Let's see the simple example to display digital clock using JavaScript date object.

There are two ways to set interval in JavaScript: by setTimeout() or setInterval() method.

```
Current Time: <span id="txt"></span>
<script>
window.onload=function(){getTime();}
function getTime(){
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
setTimeout(function(){getTime()},1000);
}
//setInterval("getTime()",1000);//another way
function checkTime(i){
if (i<10){
  i="0" + i;
 }
return i;
}
</script>
```

Test it Now

Output:

```
Current Time: 10:33:11
```

JavaScript Math

The **JavaScript math** object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors.

JavaScript Math Methods

Let's see the list of JavaScript Math methods with description.

Methods	Description
abs()	It returns the absolute value of the given number.
acos()	It returns the arccosine of the given number in radians.
asin()	It returns the arcsine of the given number in radians.
atan()	It returns the arc-tangent of the given number in radians.
cbrt()	It returns the cube root of the given number.
ceil()	It returns a smallest integer value, greater than or equal to the given number.

cos()	It returns the cosine of the given number.
cosh()	It returns the hyperbolic cosine of the given number.
exp()	It returns the exponential form of the given number.
floor()	It returns largest integer value, lower than or equal to the given number.
hypot()	It returns square root of sum of the squares of given numbers.
log()	It returns natural logarithm of a number.
max()	It returns maximum value of the given numbers.
min()	It returns minimum value of the given numbers.
pow()	It returns value of base to the power of exponent.
random()	It returns random number between 0 (inclusive) and 1 (exclusive).
round()	It returns closest integer value of the given number.
sign()	It returns the sign of the given number
sin()	It returns the sine of the given number.
sinh()	It returns the hyperbolic sine of the given number.
sqrt()	It returns the square root of the given number
tan()	It returns the tangent of the given number.
tanh()	It returns the hyperbolic tangent of the given number.

trunc()	It returns an integer part of the given number.

Math.sqrt(n)

The JavaScript math.sqrt(n) method returns the square root of the given number.

```
Square Root of 17 is: <span id="p1"></span>
<script>
document.getElementById('p1').innerHTML=Math.sqrt(17);
</script>
```

Test it Now

Output:

```
Square Root of 17 is: 4.123105625617661
```

Math.random()

The JavaScript math.random() method returns the random number between 0 to 1.

```
Random Number is: <span id="p2"></span>
<script>
document.getElementById('p2').innerHTML=Math.random();
</script>
```

Test it Now

Output:

```
Random Number is: 0.6673551449777442
```

Math.pow(m,n)

The JavaScript math.pow(m,n) method returns the m to the power of n that is m^n.

```
3 to the power of 4 is: <span id="p3"></span>
<script>
document.getElementById('p3').innerHTML=Math.pow(3,4);
</script>
```

Test it Now

Output:

```
3 to the power of 4 is: 81
```

Math.floor(n)

The JavaScript math.floor(n) method returns the lowest integer for the given number. For example 3 for 3.7, 5 for 5.9 etc.

```
Floor of 4.6 is: <span id="p4"></span>
<script>
document.getElementById('p4').innerHTML=Math.floor(4.6);
</script>
```

Test it Now

Output:

```
Floor of 4.6 is: 4
```

Math.ceil(n)

The JavaScript math.ceil(n) method returns the largest integer for the given number. For example 4 for 3.7, 6 for 5.9 etc.

```
Ceil of 4.6 is: <span id="p5"></span>
<script>
document.getElementById('p5').innerHTML=Math.ceil(4.6);
</script>
```

Test it Now

Output:

```
Ceil of 4.6 is: 5
```

Math.round(n)

The JavaScript math.round(n) method returns the rounded integer nearest for the given number. If fractional part is equal or greater than 0.5, it goes to upper value 1 otherwise lower value 0. For example 4 for 3.7, 3 for 3.3, 6 for 5.9 etc.

```
Round of 4.3 is: <span id="p6"></span><br>
Round of 4.7 is: <span id="p7"></span>
<script>
document.getElementById('p6').innerHTML=Math.round(4.3);
document.getElementById('p7').innerHTML=Math.round(4.7);
</script>
```

Test it Now

Output:

```
Round of 4.3 is: 4
Round of 4.7 is: 5
```

Math.abs(n)

The JavaScript math.abs(n) method returns the absolute value for the given number. For example 4 for -4, 6.6 for -6.6 etc.

```
Absolute value of -4 is: <span id="p8"></span>
<script>
document.getElementById('p8').innerHTML=Math.abs(-4);
</script>
```

Test it Now

Output:

```
Absolute value of -4 is: 4
```

JavaScript Number Object

The **JavaScript number** object *enables you to represent a numeric value*. It may be integer or floating-point. JavaScript number object follows IEEE standard to represent the floating-point numbers.

By the help of Number() constructor, you can create number object in JavaScript. For example:

```
var n=new Number(value);
```

If value can't be converted to number, it returns NaN(Not a Number) that can be checked by isNaN() method.

You can direct assign a number to a variable also. For example:

```
var x=102;//integer value
var y=102.7;//floating point value
var z=13e4;//exponent value, output: 130000
var n=new Number(16);//integer value by number object
```

Test it Now

Output:

```
102 102.7 130000 16
```

JavaScript Number Constants

Let's see the list of JavaScript number constants with description.

Constant	Description
MIN_VALUE	returns the largest minimum value.

MAX_VALUE	returns the largest maximum value.
POSITIVE_INFINITY	returns positive infinity, overflow value.
NEGATIVE_INFINITY	returns negative infinity, overflow value.
NaN	represents "Not a Number" value.

JavaScript Number Methods

Let's see the list of JavaScript number methods with their description.

Methods	Description
isFinite()	It determines whether the given value is a finite number.
isInteger()	It determines whether the given value is an integer.
parseFloat()	It converts the given string into a floating point number.
parseInt()	It converts the given string into an integer number.
toExponential()	It returns the string that represents exponential notation of the given number.
toFixed()	It returns the string that represents a number with exact digits after a decimal point.
toPrecision()	It returns the string representing a number of specified precision.
toString()	It returns the given number in the form of string.

JavaScript Boolean

JavaScript Boolean is an object that represents value in two states: *true* or *false*. You can create the JavaScript Boolean object by Boolean() constructor as given below.

```
Boolean b=new Boolean(value);
```

The default value of JavaScript Boolean object is *false*.

JavaScript Boolean Example

```
<script>
document.write(10<20);//true
document.write(10<5);//false
</script>
```

JavaScript Boolean Properties

Property	Description
constructor	returns the reference of Boolean function that created Boolean object.
prototype	enables you to add properties and methods in Boolean prototype.

JavaScript Boolean Methods

Method	Description
toSource()	returns the source of Boolean object as a string.
toString()	converts Boolean into String.
valueOf()	converts other type into Boolean.

Browser Object Model

1. Browser Object Model (BOM)

The **Browser Object Model** (BOM) is used to interact with the browser.

The default object of browser is window means you can call all the functions of window by specifying window or directly. For example:

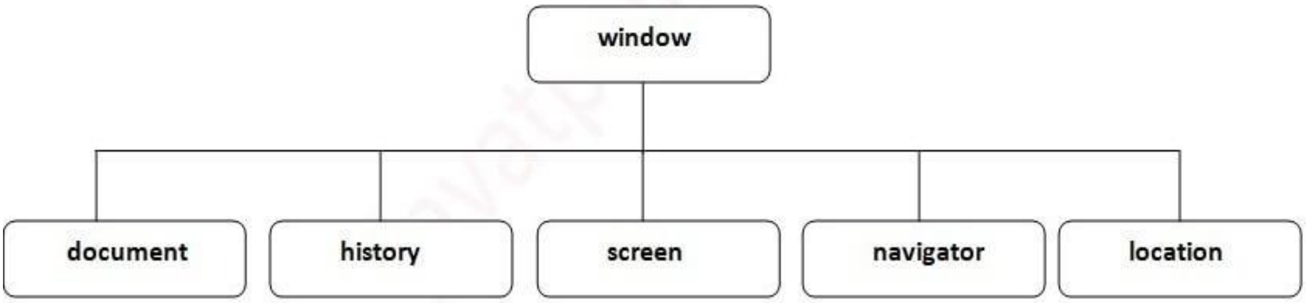
```
1. window.alert("hello javatpoint");
```

is same as:

```
1. alert("hello javatpoint");
```

You can use a lot of properties (other objects) defined underneath the window object like document, history, screen, navigator, location, innerHeight, innerWidth,

> ***Note: The document object represents an html document. It forms DOM (Document Object Model).***



Visit the next page to learn about window object fully with example.

Window Object

1. Window Object
2. Properties of Window Object
3. Methods of Window Object
4. Example of Window Object

The **window object** represents a window in browser. An object of window is created automatically by the browser.

Window is the object of browser, **it is not the object of javascript**. The javascript objects are string, array, date etc.

Note: if html document contains frame or iframe, browser creates additional window objects for each frame.

Methods of window object

The important methods of window object are as follows:

Method	Description
alert()	displays the alert box containing message with ok button.
confirm()	displays the confirm dialog box containing message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs action after specified time like calling function, evaluating expressions etc.

Example of alert() in javascript

It displays alert dialog box. It has message and ok button.

```
<script type="text/javascript">
function msg(){
 alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

Output of the above example

Example of confirm() in javascript

It displays the confirm dialog box. It has message with ok and cancel buttons.

```
<script type="text/javascript">
function msg(){
var v= confirm("Are u sure?");
if(v==true){
alert("ok");
}
else{
alert("cancel");
}

}
</script>

<input type="button" value="delete record" onclick="msg()"/>
```

Output of the above example

Example of prompt() in javascript

It displays prompt dialog box for input. It has message and textfield.

```
<script type="text/javascript">
function msg(){
var v= prompt("Who are you?");
alert("I am "+v);

}
</script>

<input type="button" value="click" onclick="msg()"/>
```

Output of the above example

Example of open() in javascript

It displays the content in a new window.

```
<script type="text/javascript">
function msg(){
open("http://www.javatpoint.com");
}
</script>
<input type="button" value="javatpoint" onclick="msg()"/>
```

Output of the above example

Example of setTimeout() in javascript

It performs its task after the given milliseconds.

```
<script type="text/javascript">
function msg(){
setTimeout(
function(){
alert("Welcome to Javatpoint after 2 seconds")
},2000);

}
</script>

<input type="button" value="click" onclick="msg()"/>
```

Output of the above example

JavaScript History Object

The **JavaScript history object** represents an array of URLs visited by the user. By using this object, you can load previous, forward or any particular page.

The history object is the window property, so it can be accessed by:

```
window.history
```

Or,

```
history
```

Property of JavaScript history object

There are only 1 property of history object.

No.	Property	Description
1	length	returns the length of the history URLs.

Methods of JavaScript history object

There are only 3 methods of history object.

No.	Method	Description
1	forward()	loads the next page.
2	back()	loads the previous page.
3	go()	loads the given page number.

Example of history object

Let’s see the different usage of history object.

1. history.back();//for previous page
2. history.forward();//for next page
3. history.go(2);//for next 2nd page
4. history.go(-2);//for previous 2nd page

JavaScript Navigator Object

1. Navigator Object
2. Properties of Navigator Object
3. Methods of Navigator Object
4. Example of Navigator Object

The **JavaScript navigator object** is used for browser detection. It can be used to get browser information such as appName, appCodeName, userAgent etc.

The navigator object is the window property, so it can be accessed by:

1. window.navigator

Or,

1. navigator

Property of JavaScript navigator object

There are many properties of navigator object that returns information of the browser.

No.	Property	Description
1	appName	returns the name
2	appVersion	returns the version
3	appCodeName	returns the code name
4	cookieEnabled	returns true if cookie is enabled otherwise false
5	userAgent	returns the user agent
6	language	returns the language. It is supported in Netscape and Firefox only.
7	userLanguage	returns the user language. It is supported in IE only.
8	plugins	returns the plugins. It is supported in Netscape and Firefox only.
9	systemLanguage	returns the system language. It is supported in IE only.
10	mimeTypes[]	returns the array of mime type. It is supported in Netscape and Firefox only.

11	platform	returns the platform e.g. Win32.
12	online	returns true if browser is online otherwise false.

Methods of JavaScript navigator object

The methods of navigator object are given below.

No.	Method	Description
1	javaEnabled()	checks if java is enabled.
2	taintEnabled()	checks if taint is enabled. It is deprecated since JavaScript 1.2.

Example of navigator object

Let’s see the different usage of history object.

```
<script>
document.writeln("<br/>navigator.appCodeName: "+navigator.appCodeName);
document.writeln("<br/>navigator.appName: "+navigator.appName);
document.writeln("<br/>navigator.appVersion: "+navigator.appVersion);
document.writeln("<br/>navigator.cookieEnabled: "+navigator.cookieEnabled);
document.writeln("<br/>navigator.language: "+navigator.language);
document.writeln("<br/>navigator.userAgent: "+navigator.userAgent);
document.writeln("<br/>navigator.platform: "+navigator.platform);
document.writeln("<br/>navigator.onLine: "+navigator.onLine);
</script>
```

Test it Now

```
navigator.appCodeName: Mozilla
navigator.appName: Netscape
navigator.appVersion: 5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
navigator.cookieEnabled: true
navigator.language: en-US
navigator.userAgent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
navigator.platform: Win32
navigator.onLine: true
```

JavaScript Screen Object

The **JavaScript screen object** holds information of browser screen. It can be used to display screen width, height, colorDepth, pixelDepth etc.

The navigator object is the window property, so it can be accessed by:

```
window.screen
```

Or,

```
screen
```

Property of JavaScript Screen Object

There are many properties of screen object that returns information of the browser.

No.	Property	Description
1	width	returns the width of the screen
2	height	returns the height of the screen
3	availWidth	returns the available width
4	availHeight	returns the available height
5	colorDepth	returns the color depth
6	pixelDepth	returns the pixel depth.

Example of JavaScript Screen Object

Let’s see the different usage of screen object.

```
<script>
document.writeln("<br/>screen.width: "+screen.width);
document.writeln("<br/>screen.height: "+screen.height);
document.writeln("<br/>screen.availWidth: "+screen.availWidth);
document.writeln("<br/>screen.availHeight: "+screen.availHeight);
document.writeln("<br/>screen.colorDepth: "+screen.colorDepth);
document.writeln("<br/>screen.pixelDepth: "+screen.pixelDepth);
</script>
```

Test it Now

```
screen.width: 1366
screen.height: 768
screen.availWidth: 1366
screen.availHeight: 728
screen.colorDepth: 24
screen.pixelDepth: 24
```

Document Object Model

The **document object** represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

As mentioned earlier, it is the object of window. So

```
window.document
```

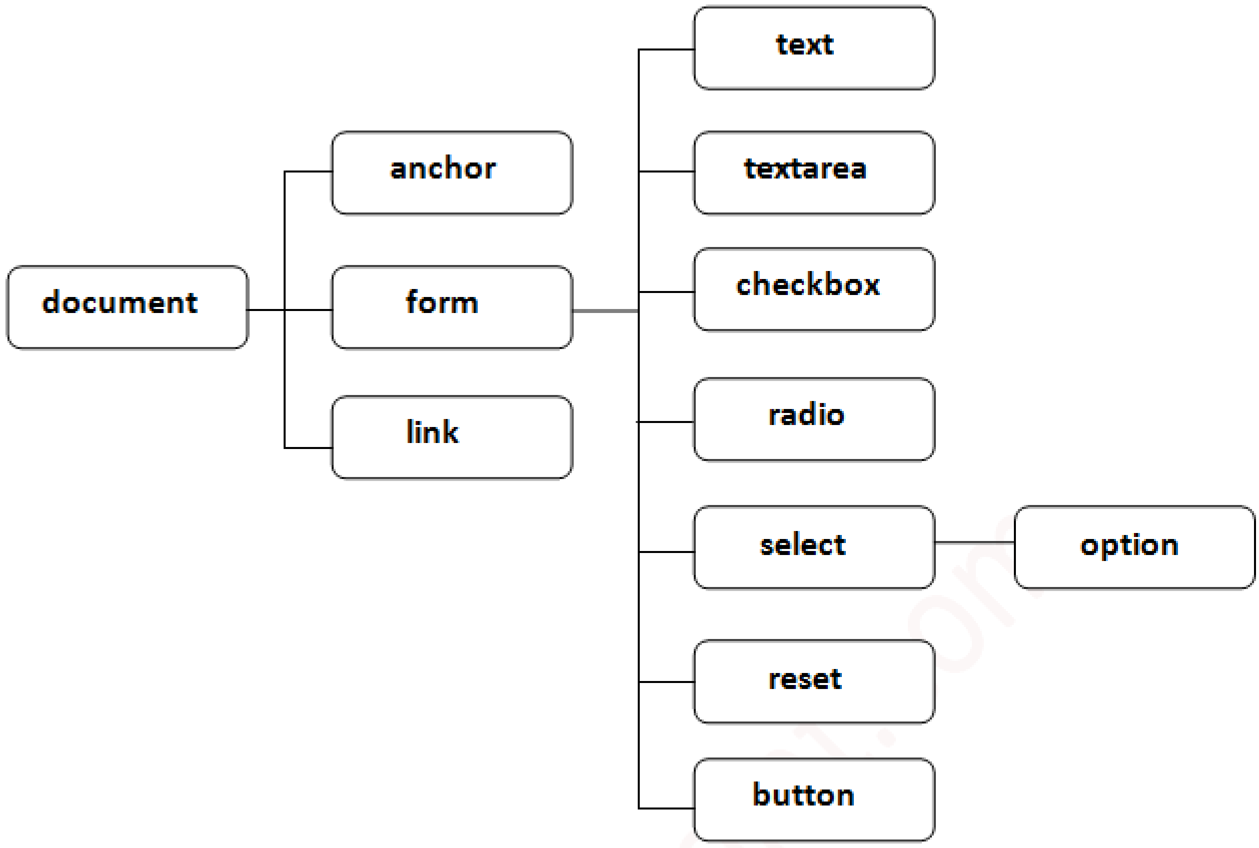
Is same as

```
document
```

According to W3C - *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

Properties of document object

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



Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.

getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
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>
```

Output of the above example

Enter Name:

printName

Javascript - document.getElementById() method

1. getElementById() method

2. Example of getElementById()

The **document.getElementById()** method returns the element of specified id.

In the previous page, we have used **document.form1.name.value** to get the value of the input value. Instead of this, we can use document.getElementById() method to get value of the input text. But we need to define id for the input field.

Let's see the simple example of document.getElementById() method that prints cube of the given number.

```
<script type="text/javascript">
function getcube(){
var number=document.getElementById("number").value;
alert(number*number*number);
}
</script>
<form>
Enter No:<input type="text" id="number" name="number"/><br/>
<input type="button" value="cube" onclick="getcube()"/>
</form>
```

Output of the above example

Enter No:

Cube

Javascript - document.getElementsByName() method

1. getElementsByName() method
2. Example of getElementsByName()

The **document.getElementsByName()** method returns all the element of specified name.

The syntax of the getElementsByName() method is given below:

```
document.getElementsByName("name")
```

Here, name is required.

Example of document.getElementsByName() method

In this example, we going to count total number of genders. Here, we are using getElementsByName() method to get all the genders.

```
<script type="text/javascript">
function totalelements()
{
var allgenders=document.getElementsByName("gender");
alert("Total Genders:"+allgenders.length);
}
</script>
<form>
Male:<input type="radio" name="gender" value="male">
Female:<input type="radio" name="gender" value="female">

<input type="button" onclick="totalelements()" value="Total Genders">
</form>
```

Output of the above example

Male:☒ Female:☒

Total Genders

Javascript - document.getElementsByTagName() method

1. getElementsByTagName() method
2. Example of getElementsByTagName()

The **document.getElementsByTagName()** method returns all the element of specified tag name.

The syntax of the getElementsByTagName() method is given below:

```
document.getElementsByTagName("name")
```

Here, name is required.

Example of document.getElementsByTagName() method

In this example, we going to count total number of paragraphs used in the document. To do this, we have called the document.getElementsByTagName("p") method that returns the total paragraphs.

```
<script type="text/javascript">
function countpara(){
var totalpara=document.getElementsByTagName("p");
alert("total p tags are: "+totalpara.length);

}
</script>
<p>This is a pragraph</p>
<p>Here we are going to count total number of paragraphs by getElementByTagName() method.</p>
<p>Let's see the simple example</p>
<button onclick="countpara()">count paragraph</button>
```

Output of the above example

```
This is a pragraph

Here we are going to count total number of paragraphs by getElementByTagName()
method.

Let's see the simple example

count paragraph
```

Another example of document.getElementsByTagName() method

In this example, we going to count total number of h2 and h3 tags used in the document.

```
<script type="text/javascript">
function counth2(){
var totalh2=document.getElementsByTagName("h2");
alert("total h2 tags are: "+totalh2.length);
}
function counth3(){
var totalh3=document.getElementsByTagName("h3");
alert("total h3 tags are: "+totalh3.length);
}
</script>
<h2>This is h2 tag</h2>
<h2>This is h2 tag</h2>
<h3>This is h3 tag</h3>
<h3>This is h3 tag</h3>
<h3>This is h3 tag</h3>
<button onclick="counth2()">count h2</button>
<button onclick="counth3()">count h3</button>
```

Output of the above example

This is h2 tag

This is h2 tag

This is h3 tag
This is h3 tag
This is h3 tag
count h2 count h3

Note: Output of the given examples may differ on this page because it will count the total number of para , total number of h2 and total number of h3 tags used in this document.

Javascript - innerHTML

The **innerHTML** property can be used to write the dynamic html on the html document.

It is used mostly in the web pages to generate the dynamic html such as registration form, comment form, links etc.

Example of innerHTML property

In this example, we are going to create the html form when user clicks on the button.

In this example, we are dynamically writing the html form inside the div name having the id mylocation. We are identifing this position by calling the document.getElementById() method.

```
<script type="text/javascript" >
function showcommentform() {
var data="Name:<input type='text' name='name'><br>Comment:<br><textarea rows='5' cols='80'></textarea>
<br><input type='submit' value='Post Comment'>";
document.getElementById('mylocation').innerHTML=data;
}
</script>
<form name="myForm">
<input type="button" value="comment" onclick="showcommentform()">
<div id="mylocation"></div>
```

```
</form>
```

Test it Now

Output of the above example

Show/Hide Comment Form Example using innerHTML

```
<!DOCTYPE html>
<html>
<head>
<title>First JS</title>
<script>
var flag=true;
function commentform(){
var cform="<form action='Comment'>Enter Name:<br><input type='text' name='name'/><br/>
Enter Email:<br><input type='email' name='email'/><br>Enter Comment:<br/>
<textarea rows='5' cols='70'></textarea><br><input type='submit' value='Post Comment'/></form>";
if(flag){
document.getElementById("mylocation").innerHTML=cform;
flag=false;
}else{
document.getElementById("mylocation").innerHTML="";
flag=true;
}
}
</script>
</head>
<body>
<button onclick="commentform()">Comment</button>
<div id="mylocation"></div>
</body>
</html>
```

Output of the above example

Comment

Integrating HTML with Javascript:-

Q)How to catch Html Element values at Runtime?

we can catch Html Element values at runtime by using 2 ways :-

1. form name
2. document object

form name:- Every HTml Element can be identified by using some name

syn to catch Html Element values at Runtime by using formname:-

```
            formname.elementname.value;
```

```
<html>
  <head>
      <script type="text/javascript">
            function GetFullName()
            {
                  var fname=f1.t1.value;
                  var lname=f1.t2.value;
                  var fullname=fname+lname;
                  document.write(fullname);
            }
     </script>
  </head>
 <body>
  <form name="f1">
    Enter FirstName
    <input type="text" name="t1">
    <br>
    Enter LastName
    <input type="text" name="t2">
    <br>
    <input type="button" name="b1" value="Display" onclick="GetFullName()">
  </form>
 </body>
</html>
```

TextBox:- TextBox is DataEntry control which is used to accept the i/p from the user at Runtime

By Default TextBox will accept the data in the form of string

+ operator is called as overloaded operator

it will perform addition operation between numbers and concadination operation between strings

2+3=5 number+number=number

2+3.5=5.5 number+number=number

3.4+4.3=7.7 number+number=number

"abc"+"def"="abcdef" string+string=string

note:- we cannot perform Arthimetic operations n string

"2"+"3"="23"

By default Textbox will accept the data in the form of string

"2"+"3"="23"

"2"*"3"=Error

"3"-"2"=Error

==

eval() :- eval() is used to convert string into number

===========

```
<html>
<head>
  <script type="text/javascript">
        function Add()
        {
          var x=eval(f1.t1.value);
          var y=eval(f1.t2.value);
           var sum=x+y;
           document.write(sum);
        }
        function Sub()
        {
          var x=eval(f1.t1.value);
```

```
            var y=eval(f1.t2.value);
             var diff=x-y;
             document.write(diff);
         }
   </script>
</head>
<body>
   <form name="f1">
       Enter Firstno
       <input type="text" name="t1">
       <br>
       Enter Secondno
       <input type="text" name="t2">
       <br>
       <input type="button" value="Add" onclick="Add()">
       <input type="button" value="Sub" onclick="Sub()">
   </form
</body>
</html>
```

Javascript - innerText

1. javascript innerText
2. Example of innerText property

The **innerText** property can be used to write the dynamic text on the html document. Here, text will not be interpreted as html text but a normal text.

It is used mostly in the web pages to generate the dynamic content such as writing the validation message, password strength etc.

Javascript innerText Example

In this example, we are going to display the password strength when releases the key after press.

```
<script type="text/javascript" >
function validate() {
```

```
var msg;
if(document.myForm.userPass.value.length>5){
msg="good";
}
else{
msg="poor";
}
document.getElementById('mylocation').innerText=msg;
 }

</script>
<form name="myForm">
<input type="password" value="" name="userPass" onkeyup="validate()">
Strength:<span id="mylocation">no strength</span>
</form>
```

Test it Now

Output of the above example

☒ Strength:no strength

JavaScript Form Validation

It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.

JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation.

Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

JavaScript Form Validation Example

In this example, we are going to validate the name and password. The name can’t be empty and password can’t be less than 6 characters long.

Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```

Test it Now

JavaScript Retype Password Validation

```
<script type="text/javascript">
function matchpass(){
var firstpassword=document.f1.password.value;
var secondpassword=document.f1.password2.value;

if(firstpassword==secondpassword){
return true;
}
else{
alert("password must be same!");
return false;
}
}
</script>

<form name="f1" action="register.jsp" onsubmit="return matchpass()">
Password:<input type="password" name="password" /><br/>
```

```
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form>
```

Test it Now

JavaScript Number Validation

Let's validate the textfield for numeric value only. Here, we are using isNaN() function.

```
<script>
function validate(){
var num=document.myform.num.value;
if (isNaN(num)){
  document.getElementById("numloc").innerHTML="Enter Numeric value only";
  return false;
}else{
  return true;
  }
}
</script>
<form name="myform" onsubmit="return validate()" >
Number: <input type="text" name="num"><span id="numloc"></span><br/>
<input type="submit" value="submit">
</form>
```

Test it Now

JavaScript validation with image

Let’s see an interactive JavaScript form validation example that displays correct and incorrect image if input is correct or incorrect.

```
<script>
function validate(){
var name=document.f1.name.value;
var password=document.f1.password.value;
var status=false;

if(name.length<1){
document.getElementById("nameloc").innerHTML=
" <img src='unchecked.gif'/> Please enter your name";
status=false;
}else{
```

```
document.getElementById("nameloc").innerHTML=" <img src='checked.gif'/>";
status=true;
}
if(password.length<6){
document.getElementById("passwordloc").innerHTML=
" <img src='unchecked.gif'/> Password must be at least 6 char long";
status=false;
}else{
document.getElementById("passwordloc").innerHTML=" <img src='checked.gif'/>";
}
return status;
}
</script>

<form name="f1" action="#" onsubmit="return validate()">
<table>
<tr><td>Enter Name:</td><td><input type="text" name="name"/>
<span id="nameloc"></span></td></tr>
<tr><td>Enter Password:</td><td><input type="password" name="password"/>
<span id="passwordloc"></span></td></tr>
<tr><td colspan="2"><input type="submit" value="register"/></td></tr>
</table>
</form>
```

Test it Now

Output:

Enter Name:

Enter Password:

☒

JavaScript email validation

We can validate the email by the help of JavaScript.

There are many criteria that need to be follow to validate the email id such as:

- email id must contain the @ and . character
- There must be at least one character before and after the @.
- There must be at least two characters after . (dot).

Let's see the simple example to validate the email field.

```
<script>
function validateemail()
{
var x=document.myform.email.value;
var atposition=x.indexOf("@");
var dotposition=x.lastIndexOf(".");
if (atposition<1 || dotposition<atposition+2 || dotposition+2>=x.length){
  alert("Please enter a valid e-mail address \n atpostion:"+atposition+"\n dotposition:"+dotposition);
  return false;
  }
}
</script>
<body>
<form name="myform"  method="post" action="#" onsubmit="return validateemail();" >
Email: <input type="text" name="email"><br/>

<input type="submit" value="register">
</form>
```

==

Javascript by Kannababu(sathyaTechnologies)   11:30 AM Batch

www.sathyatech.com

https://www.youtube.com/user/kannababubanna

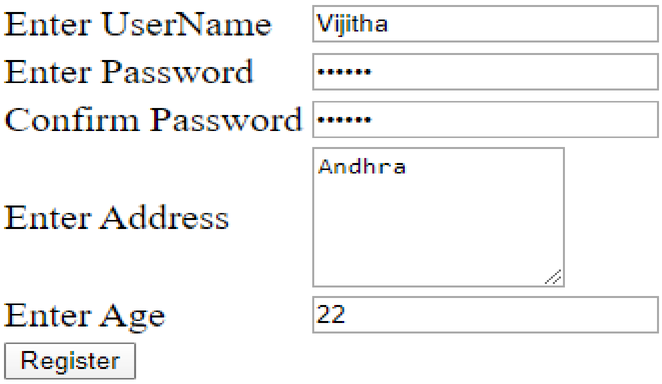
==

```
<html>
<head>
  <style type="text/css">
      #s1
      {
        color:red;
       }
 </style>
  <script type="text/javascript">
    function validate()
    {
        var uname=document.getElementById("t1").value;
        var pwd=document.getElementById("t2").value;
        var cpwd=document.getElementById("t3").value;
        var address=document.getElementById("t4").value;
        var r="";
       if(uname=="")
      {
          r="username must not be Empty"+"<br>";
      }
     if(pwd=="")
```

```
      {
          r=r+"password must not be Empty"+"<br>";
      }
      if(cpwd=="")
      {
          r=r+"confirm password must not be Empty"+"<br>";
      }
      if(address=="")
      {
          r=r+"Address must not be Empty"+"<br>";
      }
      if(uname==pwd)
      {
          r=r+"uname and password must not be same"+"<br>";
      }
     if(pwd!=cpwd)
      {
          r=r+"password mismatch"+"<br>";
      }
      if(document.getElementById("t5").value=="")
      {
           r=r+"Age must not be empty"+"<br>";
      }
      document.getElementById("s1").innerHTML=r;
    }
 </script>
</head>
<body>
 <form id="f1">
  <table>
   <tr>
   <td>
    Enter UserName
   </td>
    <td>
    <input type="text" id="t1">
    </td>
    </tr>
   <tr>
   <td>
    Enter Password
   </td>
    <td>
    <input type="password" id="t2">
    </td>
    </tr>
   <tr>
   <td>
   Confirm Password
   </td>
```

```
    <td>
    <input type="password" id="t3">
    </td>
    </tr>
<tr>
   <td>
    Enter Address
   </td>
    <td>
     <textarea rows="5" cols="15" id="t4">
     </textarea>
    </td>
    </tr>
<tr>
   <td>
     Enter Age
  </td>
    <td>
    <input type="text" id="t5">
    </td>
    </tr>
   <tr>
       <td>
     <input type="button" value="Register" onclick="validate()">
       </td>
       </tr>
  <tr>
      <td>
   <span id="s1"></span>
      </td>
  </tr>
  </table>
 </form>
</body>
</html>
```

Output: When data is entered Correctly.



Enter UserName Vijitha

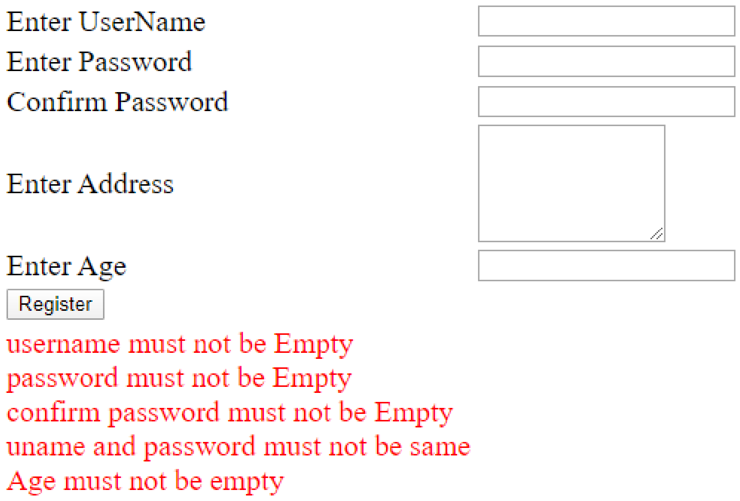
Enter Password ••••••

Confirm Password ••••••

Enter Address Andhra

Enter Age 22

Output: When data is not entered .



Enter UserName
Enter Password
Confirm Password
Enter Address
Enter Age

username must not be Empty
password must not be Empty
confirm password must not be Empty
uname and password must not be same
Age must not be empty

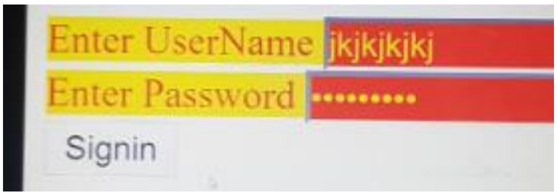
Registration Page 1:

```
<html>

<head>

<style type="text/css">

.x

{

color:yellow;

background-color:red;

}

.y
```

```
{
color:red;
background-color:yellow;
}
</style>
<script type="text/javascript">
function validate()
{
var uname=document.getElementById("t1").value;
var pwd=document.getElementById("t2").value;
if(uname=="")
{
alert("username must not be empty");
}
else if(pwd=="")
{
alert("password must not be empty");
}
}
</script>
</head>
<body>
<form id="f1">
<span class="y"> Enter UserName </span>
<input type="text" id="t1" class="x">
<br>
<span class="y"> Enter Password </span>
<input type="password" id="t2" class="x">
<br>
<input type="button" id="b1" value="Signin" onclick="validate()">
```

```
</form>

</body>

</html>
```

Output:



Enter UserName jkjkjkjkj

Enter Password •••••••••

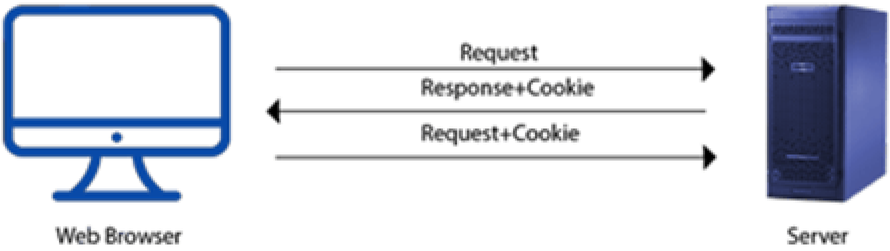
avaScript Cookies

A cookie is an amount of information that persists between a server-side and a client-side. A web browser stores this information at the time of browsing.

A cookie contains the information as a string generally in the form of a name-value pair separated by semi-colons. It maintains the state of a user and remembers the user's information among all the web pages.

How Cookies Works?

- When a user sends a request to the server, then each of that request is treated as a new request sent by the different user.
- So, to recognize the old user, we need to add the cookie with the response from the server.
- browser at the client-side.
- Now, whenever a user sends a request to the server, the cookie is added with that request automatically. Due to the cookie, the server recognizes the users.



How to create a Cookie in JavaScript?

In JavaScript, we can create, read, update and delete a cookie by using **document.cookie** property.

The following syntax is used to create a cookie:

```
document.cookie="name=value";
```

JavaScript Cookie Example

Example 1

Let's see an example to set and get a cookie.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
    <script>
    function setCookie()
    {
        document.cookie="username=Duke Martin";
    }
    function getCookie()
    {
        if(document.cookie.length!=0)
        {
        alert(document.cookie);
        }
        else
        {
        alert("Cookie not available");
        }
    }
    </script>

</body>
</html>
```

Example 2

Here, we display the cookie's name-value pair separately.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
    <script>
    function setCookie()
    {
        document.cookie="username=Duke Martin";
    }
    function getCookie()
    {
        if(document.cookie.length!=0)
        {
            var array=document.cookie.split("=");
        alert("Name="+array[0]+" "+"Value="+array[1]);
        }
        else
        {
        alert("Cookie not available");
        }
    }
    </script>

</body>
</html>
```

Example 3

In this example, we provide choices of color and pass the selected color value to the cookie. Now, cookie stores the last choice of a user in a browser. So, on reloading the web page, the user's last choice will be shown on the screen.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
        <select id="color" onchange="display()">
                <option value="Select Color">Select Color</option>
                <option value="yellow">Yellow</option>
                <option value="green">Green</option>
                <option value="red">Red</option>
            </select>
            <script type="text/javascript">
                function display()
                {
                    var value = document.getElementById("color").value;
                    if (value != "Select Color")
                    {
                        document.bgColor = value;
                        document.cookie = "color=" + value;
                    }
                }
                window.onload = function ()
                {
                    if (document.cookie.length != 0)
                    {
                        var array = document.cookie.split("=");
                        document.getElementById("color").value = array[1];
                        document.bgColor = array[1];
                    }
                }


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

Cookie Attributes

JavaScript provides some optional attributes that enhance the functionality of cookies. Here, is the list of some attributes with their description.

Attributes	Description
expires	It maintains the state of a cookie up to the specified date and time.
max-age	It maintains the state of a cookie up to the specified time. Here, time is given
path	It expands the scope of the cookie to all the pages of a website.
domain	It is used to specify the domain for which the cookie is valid.

Cookie expires attribute

The cookie expires attribute provides one of the ways to create a persistent cookie. Here, a date and time are declared that represents the active period of a cookie. Once the declared time is passed, a cookie is deleted automatically.

Let's see an example of cookie expires attribute.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
    <script>
    function setCookie()
    {
        document.cookie="username=Duke Martin;expires=Sun, 20 Aug 2030 12:00:00 UTC";
    }
    function getCookie()
    {
        if(document.cookie.length!=0)
        {
            var array=document.cookie.split("=");
        alert("Name="+array[0]+" "+"Value="+array[1]);
        }
        else
        {
        alert("Cookie not available");
        }
    }
    </script>
</body>
</html>
```

Cookie max-age attribute

The cookie max-age attribute provides another way to create a persistent cookie. Here, time is declared in seconds. A cookie is valid up to the declared time only.

Let's see an example of cookie max-age attribute.

```
<!DOCTYPE html>
```

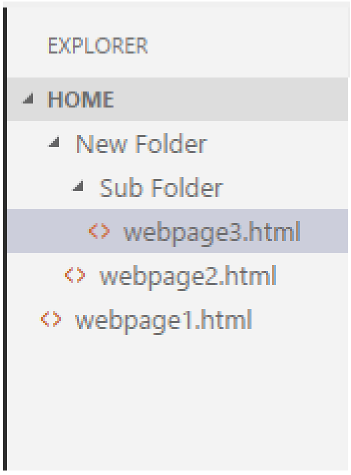
```
<html>
<head>
</head>
<body>
<input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
    <script>
    function setCookie()
    {
        document.cookie="username=Duke Martin;max-age=" + (60 * 60 * 24 * 365) + ";"
    }
    function getCookie()
    {
        if(document.cookie.length!=0)
        {
            var array=document.cookie.split("=");
        alert("Name="+array[0]+" "+"Value="+array[1]);
        }
        else
        {
        alert("Cookie not available");
        }
    }
    </script>
</body>
</html>
```

Cookie path attribute

If a cookie is created for a webpage, by default, it is valid only for the current directory and sub-directory. JavaScript provides a path attribute to expand the scope of cookie up to all the pages of a website.

Cookie path attribute Example

Let's understand the path attribute with the help of an example.



Here, if we create a cookie for webpage2.html, it is valid only for itself and its sub-directory (i.e., webpage3.html). It is not valid for webpage1.html file.

In this example, we use path attribute to enhance the visibility of cookies up to all the pages. Here, you all just need to do is to maintain the above directory structure and put the below program in all three web pages. Now, the cookie is valid for each web page.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
    <script>
    function setCookie()
    {
        document.cookie="username=Duke Martin;max-age=" + (60 * 60 * 24 * 365) + ";path=/;"
    }
    function getCookie()
    {
        if(document.cookie.length!=0)
        {
            var array=document.cookie.split("=");
        alert("Name="+array[0]+" "+"Value="+array[1]);
        }
        else
        {
        alert("Cookie not available");
        }
    }
    </script>
```

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

Cookie domain attribute

A JavaScript domain attribute specifies the domain for which the cookie is valid. Let's suppose if we provide any domain name to the attribute such like:

```
domain=javatpoint.com
```

Here, the cookie is valid for the given domain and all its sub-domains.

However, if we provide any sub-domain to the attribute such like:

```
omain=training.javatpoint.com
```

Here, the cookie is valid only for the given sub-domain. So, it's a better approach to provide domain name instead of sub-domain.

Cookie with multiple Name-Value pairs

In JavaScript, a cookie can contain only a single name-value pair. However, to store more than one name-value pair, we can use the following approach: -

- Serialize the custom object in a JSON string, parse it and then store in a cookie.
- For each name-value pair, use a separate cookie.

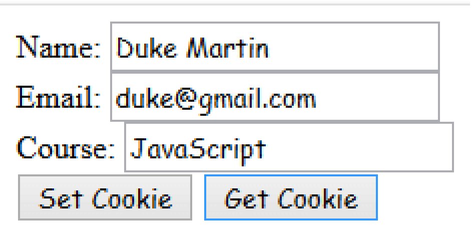
Examples to Store Name-Value pair in a Cookie

Example 1

Let's see an example to check whether a cookie contains more than one name-value pair.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    Name: <input type="text" id="name"><br>
    Email: <input type="email" id="email"><br>
    Course: <input type="text" id="course"><br>
<input type="button" value="Set Cookie" onclick="setCookie()">
<input type="button" value="Get Cookie" onclick="getCookie()">
<script>
    function setCookie()
    {
//Declaring 3 key-value pairs
        var info="Name="+ document.getElementById("name").value+";Email="+document.getElementById("email").value+";Course="+document.getElementById("course").value;
//Providing all 3 key-value pairs to a single cookie
        document.cookie=info;
    }

    function getCookie()
    {
        if(document.cookie.length!=0)
        {
       //Invoking key-value pair stored in a cookie
        alert(document.cookie);
        }
        else
        {
        alert("Cookie not available")
        }
    }
</script>
</body>
</html>
```

Output:



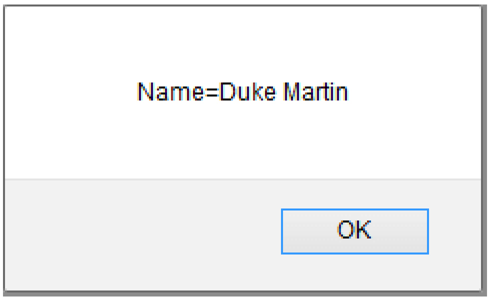
Name: Duke Martin

Email: duke@gmail.com

Course: JavaScript

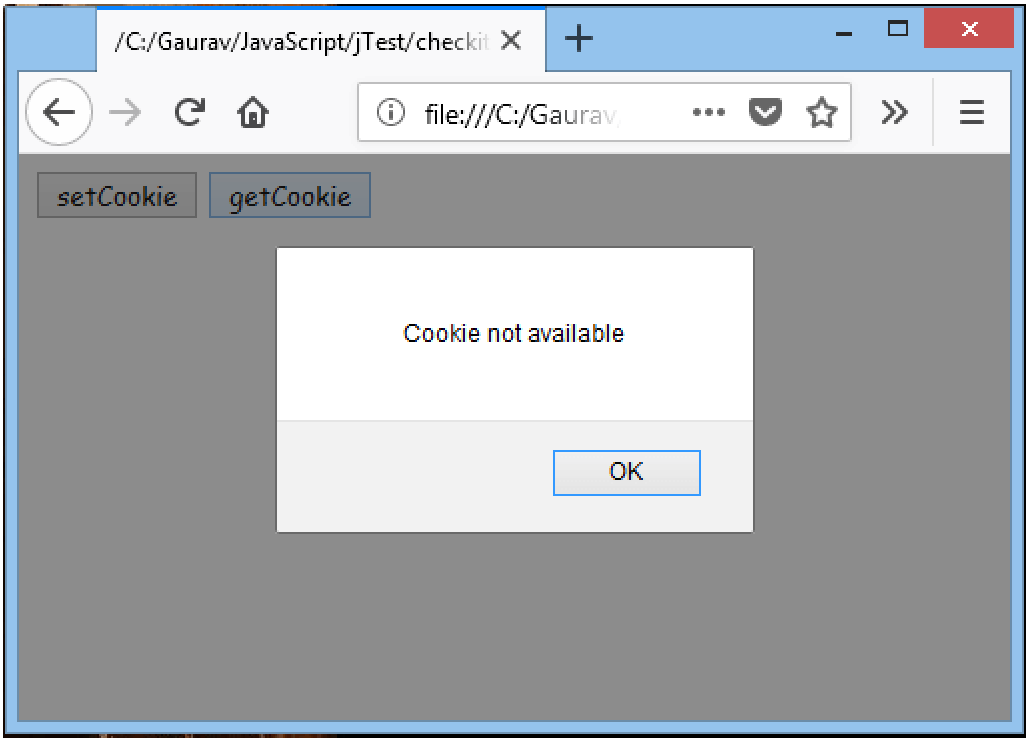
Test it Now

On clicking **Get Cookie** button, the below dialog box appears.


Name=Duke Martin

Here, we can see that only a single name-value is displayed.

However, if you click, **Get Cookie** without filling the form, the below dialog box appears.



Cookie not available

Example 2

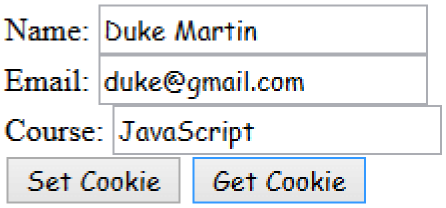
Let's see an example to store different name-value pairs in a cookie using JSON.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    Name: <input type="text" id="name"><br>
    Email: <input type="email" id="email"><br>
    Course: <input type="text" id="course"><br>
<input type="button" value="Set Cookie" onclick="setCookie()">
<input type="button" value="Get Cookie" onclick="getCookie()">
```

```

<script>
    function setCookie()
{
    var obj = {};//Creating custom object
    obj.name = document.getElementById("name").value;
    obj.email = document.getElementById("email").value;
    obj.course = document.getElementById("course").value;

//Converting JavaScript object to JSON string
var jsonString = JSON.stringify(obj);

    document.cookie = jsonString;
}
        function getCookie()
{
    if( document.cookie.length!=0)
    {
//Parsing JSON string to JSON object
    var obj = JSON.parse(document.cookie);

       alert("Name="+obj.name+" "+"Email="+obj.email+" "+"Course="+obj.course);
    }
    else
    {
       alert("Cookie not available");
    }
}
    </script>
</body>
</html>
```

Test it Now

Output:

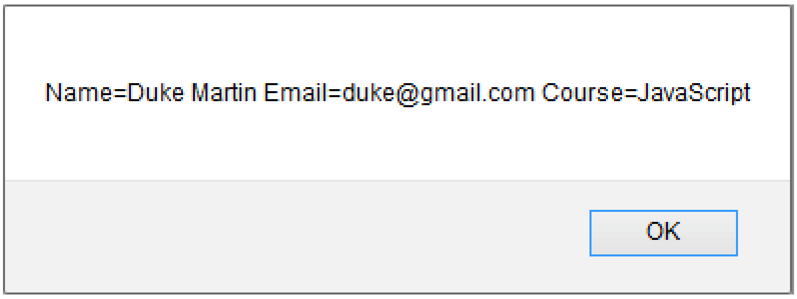


Name: Duke Martin

Email: duke@gmail.com

Course: JavaScript

On clicking **Get Cookie** button, the below dialog box appears.



Name=Duke Martin Email=duke@gmail.com Course=JavaScript

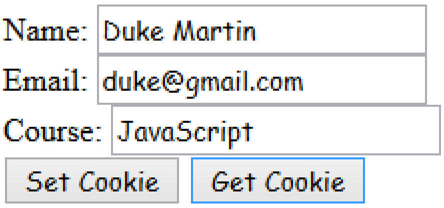
Here, we can see that all the stored name-value pairs are displayed.

Example 3

Let's see an example to store each name-value pair in a different cookie.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    Name: <input type="text" id="name"><br>
    Email: <input type="email" id="email"><br>
    Course: <input type="text" id="course"><br>
<input type="button" value="Set Cookie" onclick="setCookie()">
<input type="button" value="Get Cookie" onclick="getCookie()">

<script>
 function setCookie()
{
    document.cookie = "name=" + document.getElementById("name").value;
    document.cookie = "email=" + document.getElementById("email").value;
    document.cookie = "course=" + document.getElementById("course").value;
}
function getCookie()
{
    if (document.cookie.length != 0)
    {
        alert("Name="+document.getElementById("name").value+" Email="+document.getElementById("email").value+" Course="+document.getElementById("course").value);
    }
    else
    {
        alert("Cookie not available");
    }
}
 </script>
</body>
</html>
```

Output:



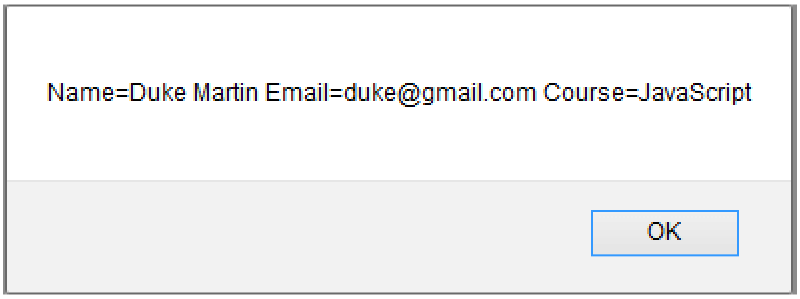
Name: Duke Martin

Email: duke@gmail.com

Course: JavaScript

Test it Now

On clicking **Get Cookie** button, the below dialog box appears.



Name=Duke Martin Email=duke@gmail.com Course=JavaScript

Here, also we can see that all the stored name-value pairs are displayed.

Deleting a Cookie in JavaScript

In the previous section, we learned the different ways to set and update a cookie in JavaScript. Apart from that, JavaScript also allows us to delete a cookie. Here, we see all the possible ways to delete a cookie.

Different ways to delete a Cookie

These are the following ways to delete a cookie:

- A cookie can be deleted by using expire attribute.
- A cookie can also be deleted by using max-age attribute.
- We can delete a cookie explicitly, by using a web browser.

Examples to delete a Cookie

Example 1

In this example, we use expire attribute to delete a cookie by providing expiry date (i.e. any past date) to it.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<input type="button" value="Set Cookie" onclick="setCookie()">
<input type="button" value="Get Cookie" onclick="getCookie()">
<script>
function setCookie()
{
    document.cookie="name=Martin Roy; expires=Sun, 20 Aug 2000 12:00:00 UTC";

}
function getCookie()
{
    if(document.cookie.length!=0)
    {
    alert(document.cookie);
    }
    else
    {
        alert("Cookie not avaliable");
    }
}
</script>
</body>
</html>
```

Example 2

In this example, we use **max-age** attribute to delete a cookie by providing zero or negative number (that represents seconds) to it.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<input type="button" value="Set Cookie" onclick="setCookie()">
```

```
<input type="button" value="Get Cookie" onclick="getCookie()">
<script>
function setCookie()
{
    document.cookie="name=Martin Roy;max-age=0";
}
function getCookie()
{
    if(document.cookie.length!=0)
    {
    alert(document.cookie);
    }
    else
    {
        alert("Cookie not avaliable");
    }
}

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

Example 3

Let's see an example to set, get and delete multiple cookies.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<input type="button" value="Set Cookie1" onclick="setCookie1()">
<input type="button" value="Get Cookie1" onclick="getCookie1()">
<input type="button" value="Delete Cookie1" onclick="deleteCookie1()">
<br>
<input type="button" value="Set Cookie2" onclick="setCookie2()">
<input type="button" value="Get Cookie2" onclick="getCookie2()">
<input type="button" value="Delete Cookie2" onclick="deleteCookie2()">
<br>
<input type="button" value="Display all cookies" onclick="displayCookie()">

<script>
function setCookie1()
{
    document.cookie="name=Martin Roy";
     cookie1=  document.cookie;
}
function setCookie2()
{
    document.cookie="name=Duke William";
     cookie2=  document.cookie;
}

function getCookie1()
{
    if(cookie1.length!=0)
    {
    alert(cookie1);
    }
    else
    {
        alert("Cookie not available");
    }
}

function getCookie2()
{
    if(cookie2.length!=0)
    {
    alert(cookie2);
    }
    else
    {
        alert("Cookie not available");
    }
}

function deleteCookie1()
{
    document.cookie=cookie1+";max-age=0";
    cookie1=document.cookie;
    alert("Cookie1 is deleted");
}

function deleteCookie2()
{
    document.cookie=cookie2+";max-age=0";
    cookie2=document.cookie;
```

```
   alert("Cookie2 is deleted");
}

function displayCookie()
{
if(cookie1!=0&&cookie2!=0)
{
    alert(cookie1+" "+cookie2);
}
else if(cookie1!=0)
{
    alert(cookie1);
}
else if(cookie2!=0)
{
    alert(cookie2);
}
else{
    alert("Cookie not available");
}

}

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

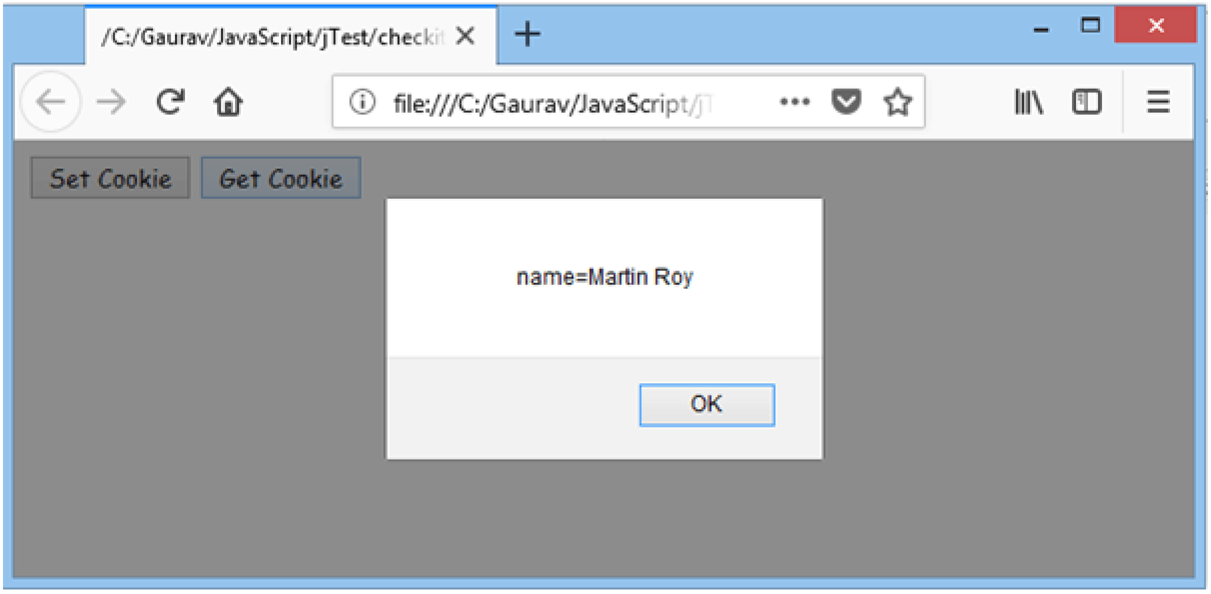
Example 4

Let's see an example to delete a cookie explicitly.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<input type="button" value="Set Cookie" onclick="setCookie()">
<input type="button" value="Get Cookie" onclick="getCookie()">
<script>
function setCookie()
{
    document.cookie="name=Martin Roy";

}
```

```
function getCookie()
{
    if(document.cookie.length!=0)
    {
    alert(document.cookie);
    }
    else
    {
        alert("Cookie not avaliable");
    }
}
</script>
</body>
</html>
```

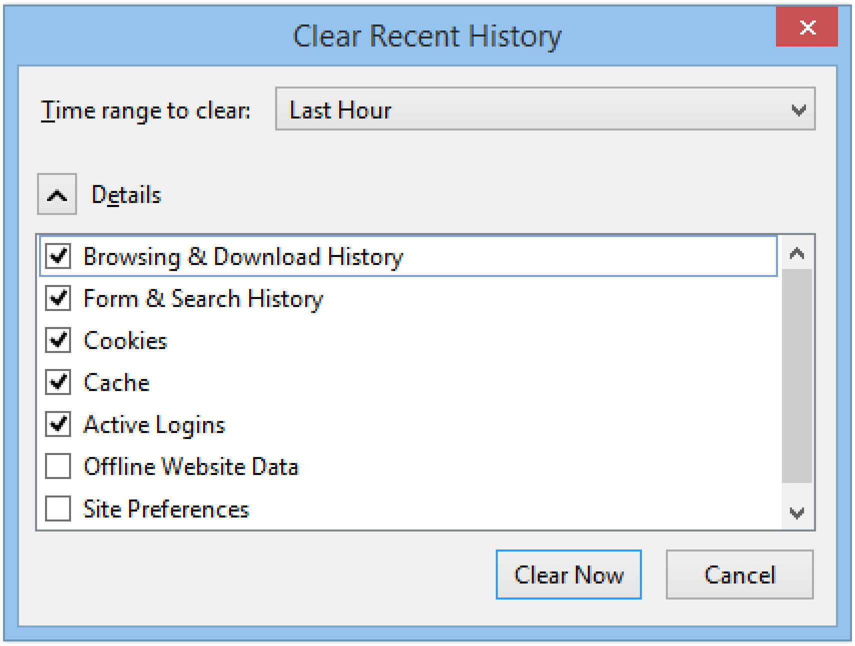
After clicking **Set Cookie** once, whenever we click **Get Cookie**, the cookies key and value is displayed on the screen.


/C:/Gaurav/JavaScript/jTest/checki
file:///C:/Gaurav/JavaScript/j
Set Cookie
Get Cookie
OK

name=Martin Roy

To delete a cookie explicitly, follow the following steps:

- Open Mozilla Firefox.
- Click **Open menu - Library - History - Clear Recent History - Details**.



Clear Recent History

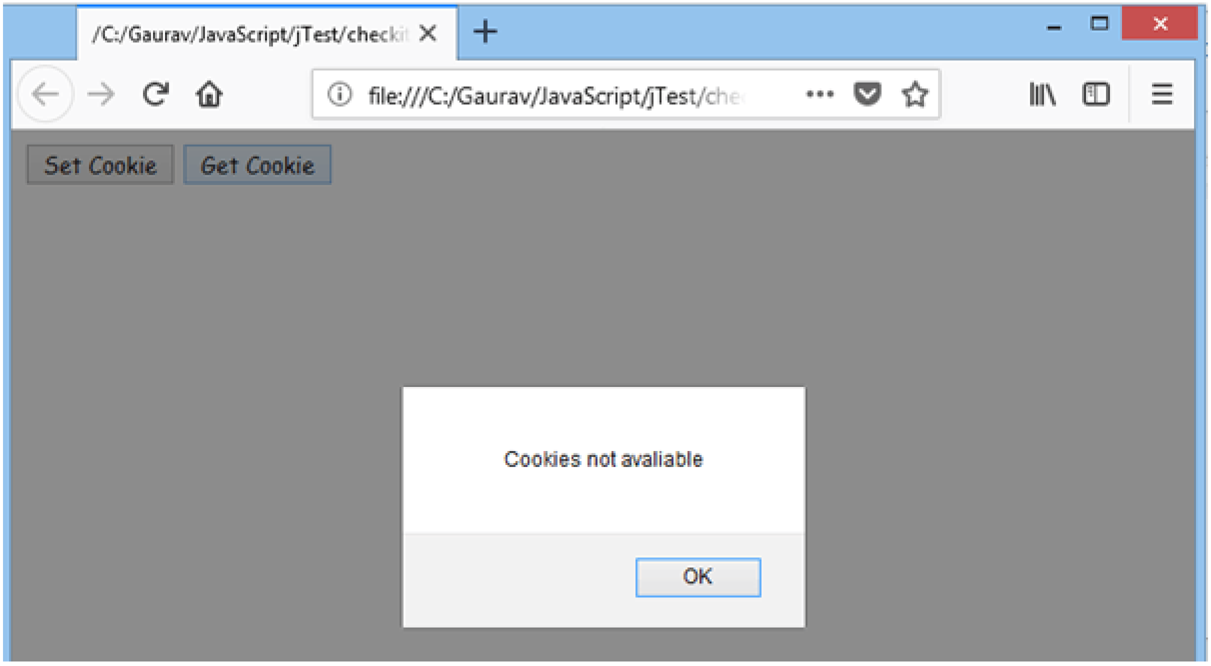
Time range to clear: Last Hour

Details

☑ Browsing & Download History
☑ Form & Search History
☑ Cookies
☑ Cache
☑ Active Logins
□ Offline Website Data
□ Site Preferences

- Here we can see a **Cookies** checkbox which is already marked. Now, click **Clear Now** to delete the cookies explicitly.

Now, on clicking **Get Cookie**, the below dialog box appears.



/C:/Gaurav/JavaScript/jTest/checki

file:///C:/Gaurav/JavaScript/jTest/che

Cookies not avaliable

Here, we can see that the cookies are deleted.

JavaScript Events

The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.

For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

Some of the HTML events and their event handlers are:

Mouse events:

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse co
mouseout	onmouseout	When the cursor of the mouse lea
mousedown	onmousedown	When the mouse button is presse
mouseup	onmouseup	When the mouse button is release
mousemove	onmousemove	When the mouse movement takes

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user pre

Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element

submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form ele
change	onchange	When the user modifies or changes the

Window/Document events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of t
unload	onunload	When the visitor leaves the current webpa
resize	onresize	When the visitor resizes the window of the

Let's discuss some examples over events and their handlers.

Click Event

```
<html>
<head> Javascript Events </head>
<body>
<script language="Javascript" type="text/Javascript">
    <!--
    function clickevent()
    {
        document.write("This is JavaTpoint");
    }
    //-->
</script>
<form>
<input type="button" onclick="clickevent()" value="Who's this?"/>
</form>
</body>
</html>
```

Test it Now

MouseOver Event

```
<html>
<head>
<h1> Javascript Events </h1>
</head>
<body>
<script language="Javascript" type="text/Javascript">
    <!--
    function mouseoverevent()
    {
        alert("This is JavaTpoint");
    }
    //-->
</script>
<p onmouseover="mouseoverevent()"> Keep cursor over me</p>
</body>
</html>
```

Test it Now

Focus Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onfocus="focusevent()"/>
<script>
<!--
    function focusevent()
    {
        document.getElementById("input1").style.background=" aqua";
    }
//-->
</script>
</body>
</html>
```

Test it Now

Keydown Event

```
<html>
<head> Javascript Events</head>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onkeydown="keydownevent()"/>
```

```
<script>
<!--
    function keydownevent()
    {
        document.getElementById("input1");
        alert("Pressed a key");
    }
//-->
</script>
</body>
</html>
```

Test it Now

Load event

```
<html>
<head>Javascript Events</head>
</br>
<body onload="window.alert('Page successfully loaded');">
<script>
<!--
document.write("The page is loaded successfully");
//-->
</script>
</body>
</html>
```

Test it Now

JavaScript Interview Questions

JavaScript interview questions and answers for provides a list of top 20 interview questions. The frequently asked JavaScript interview questions with answers for beginners and professionals are given below.

1) What is JavaScript?

JavaScript is *a scripting language*. It is different from Java language. It is object-based, lightweight, cross-platform translated language. It is widely used for client-side validation. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser. More details.

2) List some features of JavaScript.

Some of the features of JavaScript are:

- Lightweight
- Interpreted programming language
- Good for the applications which are network-centric
- Complementary to Java
- Complementary to HTML
- Open source

Cross-platform

3) List some of the advantages of JavaScript.

Some of the advantages of JavaScript are:

- Server interaction is less
- Feedback to the visitors is immediate
- Interactivity is high
- Interfaces are richer

4) List some of the disadvantages of JavaScript.

Some of the disadvantages of JavaScript are:

- No support for multithreading
- No support for multiprocessing
- Reading and writing of files is not allowed
- No support for networking applications.

5) Define a named function in JavaScript.

The function which has named at the time of definition is called a named function. For example

```
function msg()
{
  document.writeln("Named Function");
}
msg();
```

6) Name the types of functions

The types of function are:

- Named - These type of functions contains name at the time of definition. For Example:

```
function display()
{
  document.writeln("Named Function");
}
display();
```

- Anonymous - These type of functions doesn't contain any name. They are declared dynamically at runtime.

```
var display=function()
{
  document.writeln("Anonymous Function");
}
display();
```

7) Define anonymous function

It is a function that has no name. These functions are declared dynamically at runtime using the function operator instead of the function declaration. The function operator is more flexible than a function declaration. It can be easily used in the place of an expression. For example:

```
var display=function()
{
  alert("Anonymous Function is invoked");
}
display();
```

8) Can an anonymous function be assigned to a variable?

Yes, you can assign an anonymous function to a variable.

9) In JavaScript what is an argument object?

The variables of JavaScript represent the arguments that are passed to a function.

10) Define closure.

In JavaScript, we need closures when a variable which is defined outside the scope in reference is accessed from some inner scope.

```
var num = 10;
function sum()
{
document.writeln(num+num);
}
sum();
```

11) If we want to return the character from a specific index which method is used?

The JavaScript string charAt() method is used to find out a char value present at the specified index. The index number starts from 0 and goes to n-1, where n is the length of the string. The index value can't be a negative, greater than or equal to the length of the string. For example:

```
var str="Javatpoint";
document.writeln(str.charAt(4));
```

12) What is the difference between JavaScript and JScript?

Netscape provided the JavaScript language. Microsoft changed the name and called it JScript to avoid the trademark issue. In other words, you can say JScript is the same as JavaScript, but Microsoft provides it.

13) How to write a hello world example of JavaScript?

A simple example of JavaScript hello world is given below. You need to place it inside the body tag of HTML.

```
<script type="text/javascript">
document.write("JavaScript Hello World!");
</script>
```

More details.

14) How to use external JavaScript file?

I am assuming that js file name is message.js, place the following script tag inside the head tag.

```
1. <script type="text/javascript" src="message.js"></script>
```

More details.

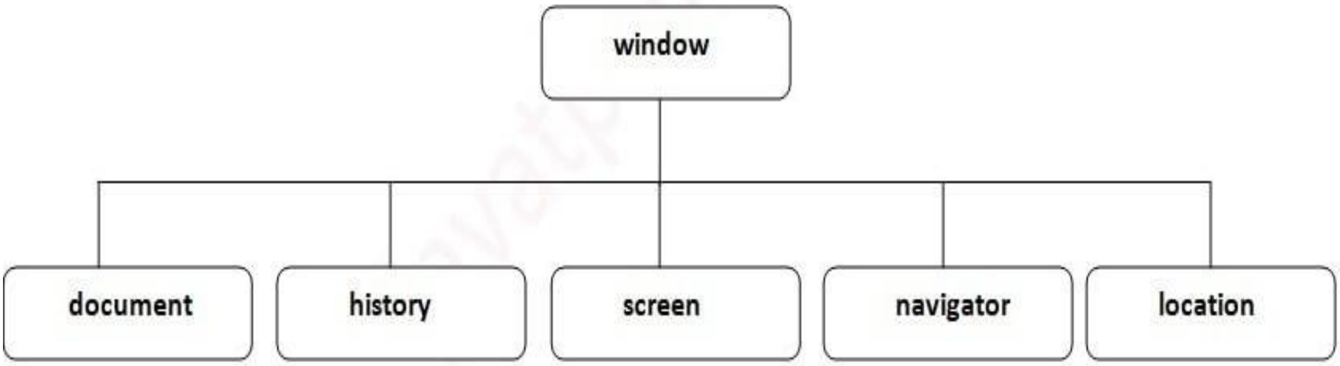
15) Is JavaScript case sensitive language?

Yes, JavaScript is a case sensitive language. For example:

```
1. Var msg = "JavaScript is a case-sensitive language"; //Here, var should be used to declare a variable
2. function display()
3. {
4. document.writeln(msg); // It will not display the result.
5. }
6. display();
```

16) What is BOM?

BOM stands for *Browser Object Model*. It provides interaction with the browser. The default object of a browser is a window. So, you can call all the functions of the window by specifying the window or directly. The window object provides various properties like document, history, screen, navigator, location, innerHeight, innerWidth,



More Details: Browser Object Model

17) What is DOM? What is the use of document object?

DOM stands for *Document Object Model*. A document object represents the HTML document. It can be used to access and change the content of HTML.

More Details: Document Object Model

18) What is the use of window object?

The window object is created automatically by the browser that represents a window of a browser. It is not an object of JavaScript. It is a browser object.

The window object is used to display the popup dialog box. Let's see with description.

Method	Description
alert()	displays the alert box containing the message with ok button.
confirm()	displays the confirm dialog box containing the message with ok and cancel button.
prompt()	displays a dialog box to get input from the user.
open()	opens the new window.
close()	closes the current window.
setTimeout()	performs the action after specified time like calling function, evaluating expressions.

More details.

19) What is the use of history object?

The history object of a browser can be used to switch to history pages such as back and forward from the current page or another page. There are three methods of history object.

1. history.back() - It loads the previous page.
2. history.forward() - It loads the next page.
3. history.go(number) - The number may be positive for forward, negative for backward. It loads the given page number.

More details.

20) How to write a comment in JavaScript?

There are two types of comments in JavaScript.

1. Single Line Comment: It is represented by // (double forward slash)
2. Multi-Line Comment: Slash represents it with asterisk symbol as /* write comment here */

More details.

21) How to create a function in JavaScript?

To create a function in JavaScript, follow the following syntax.

```
function function_name(){
//function body
}
```

More details.

22) What are the JavaScript data types?

There are two types of data types in JavaScript:

1. Primitive Data Types - The primitive data types are as follows:

Data Type	Description
String	represents a sequence of characters, e.g., "hello"
Number	represents numeric values, e.g., 100
Boolean	represents boolean value either false or true
Undefined	represents an undefined value
Null	represents null, i.e., no value at all

2. Non-primitive Data Types - The non-primitive data types are as follows:

Data Type	Description
Object	represents an instance through which we can access members
Array	represents a group of similar values
RegExp	represents regular expression

More details.

23) What is the difference between == and ===?

The == operator checks equality only whereas === checks equality, and data type, i.e., a value must be of the same type.

24) How to write HTML code dynamically using JavaScript?

The innerHTML property is used to write the HTML code using JavaScript dynamically. Let's see a simple example:

```
document.getElementById('mylocation').innerHTML="<h2>This is heading using JavaScript</h2>";
```

More details.

25) How to write normal text code using JavaScript dynamically?

The innerText property is used to write the simple text using JavaScript dynamically. Let's see a simple example:

```
document.getElementById('mylocation').innerText="This is text using JavaScript";
```

More details.

26) How to create objects in JavaScript?

There are 3 ways to create an object in JavaScript.

1. By object literal
2. By creating an instance of Object
3. By Object Constructor

Let's see a simple code to create an object using object literal.

```
emp={id:102,name:"Rahul Kumar",salary:50000}
```

More details.

27) How to create an array in JavaScript?

There are 3 ways to create an array in JavaScript.

1. By array literal
2. By creating an instance of Array
3. By using an Array constructor

Let's see a simple code to create an array using object literal.

```
var emp=["Shyam","Vimal","Ratan"];
```

More details.

28) What does the isNaN() function?

The isNan() function returns true if the variable value is not a number. For example:

```
function number(num) {
  if (isNaN(num)) {
    return "Not a Number";
  }
  return "Number";
}
console.log(number('1000F'));
// expected output: "Not a Number"

console.log(number('1000'));
// expected output: "Number"
```

29) What is the output of 10+20+"30" in JavaScript?

3030 because 10+20 will be 30. If there is numeric value before and after +, it treats as binary + (arithmetic operator).

```
function display()
{
  document.writeln(10+20+"30");
}
display();
```

30) What is the output of "10"+20+30 in JavaScript?

102030 because after a string all the + will be treated as string concatenation operator (not binary +).

```
function display()
{
  document.writeln("10"+20+30);
}
display();
```

31) Difference between Client side JavaScript and Server side JavaScript?

Client-side JavaScript comprises the basic language and predefined objects which are relevant to running JavaScript in a browser. The client-side JavaScript is embedded directly by in the HTML pages. The browser interprets this script at runtime.

Server-side JavaScript also resembles client-side JavaScript. It has a relevant JavaScript which is to run in a server. The server-side JavaScript are deployed only after compilation.

32) In which location cookies are stored on the hard disk?

The storage of cookies on the hard disk depends on the OS and the browser.

The Netscape Navigator on Windows uses a cookies.txt file that contains all the cookies. The path is c:\Program Files\Netscape\Users\username\cookies.txt

The Internet Explorer stores the cookies on a file username@website.txt. The path is: c:\Windows\Cookies\username@Website.txt.

33) What is the real name of JavaScript?

The original name was **Mocha**, a name chosen by Marc Andreessen, founder of Netscape. In September of 1995, the name was changed to LiveScript. In December 1995, after receiving a trademark license from Sun, the name JavaScript was adopted.

34) What is the difference between undefined value and null value?

Undefined value: A value that is not defined and has no keyword is known as undefined value. For example:

```
int number;//Here, a number has an undefined value.
```

Null value: A value that is explicitly specified by the keyword "null" is known as a null value. For example:

```
String str=null;//Here, str has a null value.
```

35) How to set the cursor to wait in JavaScript?

The cursor can be set to wait in JavaScript by using the property "cursor". The following example illustrates the usage:

```
<script>
window.document.body.style.cursor = "wait";
</script>
```

36) What is this [[[]]]?

This is a three-dimensional array.

```
var myArray = [[[]]];
```

37) Are Java and JavaScript same?

No, Java and JavaScript are the two different languages. Java is a robust, secured and object-oriented programming language whereas JavaScript is a client-side scripting language with some limitations.

38) What is negative infinity?

Negative Infinity is a number in JavaScript which can be derived by dividing the negative number by zero. For example:

```
var num=-5;
function display()
{
  document.writeln(num/0);
}
display();
//expected output: -Infinity
```

39) What is the difference between View state and Session state?

"View state" is specific to a page in a session whereas "Session state" is specific to a user or browser that can be accessed across all pages in the web application.

40) What are the pop-up boxes available in JavaScript?

- Alert Box
- Confirm Box
- Prompt Box

Example of alert() in JavaScript

```
<script type="text/javascript">
function msg(){
 alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

Example of confirm() in JavaScript

```
<script type="text/javascript">
function msg(){
var v= confirm("Are u sure?");
if(v==true){
alert("ok");
}
else{
alert("cancel");
}

}
</script>

<input type="button" value="delete record" onclick="msg()"/>
```

Example of prompt() in JavaScript

```
<script type="text/javascript">
function msg(){
var v= prompt("Who are you?");
alert("I am "+v);

}
</script>

<input type="button" value="click" onclick="msg()"/>
```

41) How can we detect OS of the client machine using JavaScript?

The **navigator.appVersion** string can be used to detect the operating system on the client machine.

42) How to submit a form using JavaScript by clicking a link?

Let's see the JavaScript code to submit the form by clicking the link.

```
<form name="myform" action="index.php">
Search: <input type='text' name='query' />
<a href="javascript: submitform()">Search</a>
</form>
<script type="text/javascript">
function submitform()
{
  document.myform.submit();
}
</script>
```

43) Is JavaScript faster than ASP script?

Yes, because it doesn't require web server's support for execution.

44) How to change the background color of HTML document using JavaScript?

```
<script type="text/javascript">
document.body.bgColor="pink";
```

```
</script>
```

45) How to handle exceptions in JavaScript?

By the help of try/catch block, we can handle exceptions in JavaScript. JavaScript supports try, catch, finally and throw keywords for exception handling.

46) How to validate a form in JavaScript?

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```

Test it Now

Visit here: JavaScript form validation.

47) How to validate email in JavaScript?

```
<script>
function validateemail()
{
var x=document.myform.email.value;
var atposition=x.indexOf("@");
var dotposition=x.lastIndexOf(".");
if (atposition<1 || dotposition<atposition+2 || dotposition+2>=x.length){
  alert("Please enter a valid e-mail address \n atpostion:"+atposition+"\n dotposition:"+dotposition);
  return false;
  }
}
</script>
<body>
<form name="myform"  method="post" action="#" onsubmit="return validateemail();">
Email: <input type="text" name="email"><br/>

<input type="submit" value="register">
</form>
```

Test it Now

Visit here: JavaScript Email validation.

48) What is this keyword in JavaScript?

The this keyword is a reference variable that refers to the current object. For example:

```
var address=
{
company:"Javatpoint",
city:"Noida",
state:"UP",
fullAddress:function()
{
return this.company+" "+this.city+" "+this.state;
}
};
var fetch=address.fullAddress();
document.writeln(fetch);
```

49) What is the requirement of debugging in JavaScript?

JavaScript didn't show any error message in a browser. However, these mistakes can affect the output. The best practice to find out the error is to debug the code. The code can be debugged easily by using web browsers like Google Chrome, Mozilla Firebox.

To perform debugging, we can use any of the following approaches:

- Using console.log() method
- Using debugger keyword

50) What is the use of debugger keyword in JavaScript?

JavaScript debugger keyword sets the breakpoint through the code itself. The debugger stops the execution of the program at the position it is applied. Now, we can start the flow of execution manually. If an exception occurs, the execution will stop again on that particular line.. For example:

```
function display()
{
x = 10;
y = 15;
z = x + y;
debugger;
document.write(z);
document.write(a);
}
display();
```

51) What is the role of a strict mode in JavaScript?

The JavaScript strict mode is used to generates silent errors. It provides "use strict"; expression to enable the strict mode. This expression can only be placed as the first statement in a script or a function. For example:

```
"use strict";
x=10;
console.log(x);
```

52) What is the use of Math object in JavaScript?

The JavaScript math object provides several constants and methods to perform a mathematical operation. Unlike date object, it doesn't have constructors. For example:

```
function display()
{
  document.writeln(Math.random());
}
display();
```

53) What is the use of a Date object in JavaScript?

The JavaScript date object can be used to get a year, month and day. You can display a timer on the webpage by the help of JavaScript date object.

```
function display()
{
  var date=new Date();
var day=date.getDate();
var month=date.getMonth()+1;
var year=date.getFullYear();
document.write("<br>Date is: "+day+"/"+month+"/"+year);
}
display();
```

54) What is the use of a Number object in JavaScript?

The JavaScript number object enables you to represent a numeric value. It may be integer or floating-point. JavaScript number object follows the IEEE standard to represent the floating-point numbers.

```
function display()
{
var x=102;//integer value
var y=102.7;//floating point value
var z=13e4;//exponent value, output: 130000
var n=new Number(16);//integer value by number object
document.write(x+" "+y+" "+z+" "+n);
}
display();
```

55) What is the use of a Boolean object in JavaScript?

The JavaScript Boolean is an object that represents value in two states: true or false. You can create the JavaScript Boolean object by Boolean() constructor.

```
function display()
{
document.writeln(10<20);//true
document.writeln(10<5);//false
}
display();
```

56) What is the use of a TypedArray object in JavaScript?

The JavaScript TypedArray object illustrates an array like a view of an underlying binary data buffer. There is any number of different global properties, whose values are TypedArray constructors for specific element types.

```
function display()
```

```
{
var arr1= [1,2,3,4,5,6,7,8,9,10];
       arr1.copyWithin(2) ;
       document.write(arr1);
}
display();
```

57) What is the use of a Set object in JavaScript?

The JavaScript Set object is used to store the elements with unique values. The values can be of any type i.e. whether primitive values or object references. For example:

```
function display()
{
var set = new Set();
set.add("jQuery");
set.add("AngularJS");
set.add("Bootstrap");
for (let elements of set) {
 document.writeln(elements+"<br>");
}
}
display();
```

58) What is the use of a WeakSet object in JavaScript?

The JavaScript WeakSet object is the type of collection that allows us to store weakly held objects. Unlike Set, the WeakSet are the collections of objects only. It doesn't contain the arbitrary values. For example:

```
function display()
{
var ws = new WeakSet();
var obj1={};
var obj2={};
ws.add(obj1);
ws.add(obj2);
//Let's check whether the WeakSet object contains the added object
document.writeln(ws.has(obj1)+"<br>");
document.writeln(ws.has(obj2));
}
display()
```

59) What is the use of a Map object in JavaScript?

The JavaScript Map object is used to map keys to values. It stores each element as key-value pair. It operates the elements such as search, update and delete on the basis of specified key. For example:

```
function display()
{
var map=new Map();
map.set(1,"jQuery");
map.set(2,"AngularJS");
map.set(3,"Bootstrap");
document.writeln(map.get(1)+"<br>");
document.writeln(map.get(2)+"<br>");
document.writeln(map.get(3));
}
display();
```

60) What is the use of a WeakMap object in JavaScript?

The JavaScript WeakMap object is a type of collection which is almost similar to Map. It stores each element as a key-value pair where keys are weakly referenced. Here, the keys are objects and the values are arbitrary values. For example:

```
function display()
{
var wm = new WeakMap();
var obj1 = {};
var obj2 = {};
var obj3= {};
wm.set(obj1, "jQuery");
wm.set(obj2, "AngularJS");
wm.set(obj3,"Bootstrap");
document.writeln(wm.has(obj2));
}
display();
```