ASSIGNMENT NO – 02

AIM: Implement a web page index.htm for any client website (e.g., a restaurant website project) using following:

a. HTML syntax: heading tags, basic tags and attributes, frames, tables, images, lists, linksfor text and images, forms etc.
b. Use of Internal CSS, Inline CSS, External CSS

LEARNING OBJECTIVES:

1. Understand Basic Tags of HTML
2. Understand CSS

THEORY:

Basic HTML

HTML stands for **Hyper Text Markup Language**. An HTML file is a text file containing markup tags. The markup tags tell the Web browser how to display the page. An HTML file must have an ‘htm’ or ‘html’ file extension. An HTML file can be created using a simple text editor. The rule-making body of the Web is **World Wide Web Consortium** (**W3C**). W3C puts together specifications for Web standards. The most essential Web standards are **HTML**, CSS and XML. The latest HTML standard is XHTML 1.0.

Example: Creating a simple web page

1. Start Notepad.
2. Type in the following text

```
<html>
<head>
<title>Title of page</title>
</head>
<body>
This is a very basic webpage. <b>This text will be displayed in bold</b>
</body>
</html>
```

3. Save the file as "firstpage.html".
4. Double click the saved file the browser will display the page.

EXAMPLE EXPLAINED:

The first tag in your HTML document is <html>. This tag tells your browser that this is the start of an HTML document. The last tag in your document is </html>. This tag tells your browser that this is the end of the HTML document.

The text between the <head> tag and the </head> tag is header information. Header information is not displayed in the browser window.

The text between the <title> tags is the title of your document. The title is displayed in your browser's caption.

The text between the <body> tags is the text that will be displayed in your browser.

The text between the <b> and </b> tags will be displayed in a bold font.

HTM OR HTML EXTENSION?

When you save an HTML file, you can use either the .htm or the .html extension. We have used **.html** in our example.

HTML TAGS

1. HTML tags are used to mark-up HTML elements
2. HTML tags are surrounded by the two characters < and >
3. The surrounding characters are called angle brackets
4. HTML tags normally come in pairs like <b> and </b>
5. The first tag in a pair is the start tag, the second tag is the end tag
6. The text between the start and end tags is the element content
7. HTML tags are not case sensitive, <b> means the same as <B>

USE LOWERCASE TAGS?

We have just said that HTML tags are not case sensitive: <B> means the same as <b>. It is recommended to always use because

IF YOU WANT TO PREPARE YOURSELF FOR THE NEXT GENERATIONS OF HTML, YOU SHOULD START USING LOWERCASE TAGS. THE WORLD WIDE WEB CONSORTIUM RECOMMENDS LOWERCASE TAGS IN THEIR HTML 4 RECOMMENDATION, AND XHTML (THE NEXT GENERATION HTML) DEMANDS LOWERCASE TAGS.

Tags can have attributes. Attributes can provide additional information about the HTML elements on your page.

This tag defines the body element of your HTML page: <body>. With an added bgcolor attribute, you can tell the browser that the background color of your page should be red, like this: <body bgcolor="red">.

Attributes always come in name/value pairs like this: name="value".

Attributes are always added to the start tag of an HTML element.

QUOTE STYLES, "RED" OR 'RED'?

Attribute values should always be enclosed in quotes. Double style quotes are the most common, but single style quotes are also allowed. In some rare situations, like when the attribute value itself contains quotes, it is necessary to use single quotes:

HEADINGS

Headings are defined with the <h1> to <h6> tags. <h1> defines the largest heading. <h6> defines the smallest heading.

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
<h4>This is a heading</h4>
<h5>This is a heading</h5>
<h6>This is a heading</h6>
```

HTML automatically adds an extra blank line before and after a heading.

PARAGRAPHS

Paragraphs are defined with the <p> tag.

```
<p>This is a paragraph</p>

<p>This is another paragraph</p>
```

HTML automatically adds an extra blank line before and after a paragraph.

LINE BREAKS

The <br> tag is used when you want to end a line, but don't want to start a new paragraph. The <br> tag forces a line break wherever you place it.
<p>This <br> is a para<br>graph with line breaks</p>
The <br> tag is an empty tag. It has no closing tag.

COMMENTS IN HTML

The comment tag is used to insert a comment in the HTML source code. A comment will be ignored by the browser. You can use comments to explain your code, which can help you when you edit the source code at a later date.
<!-- This is a comment -->
Note: that you need an exclamation point after the opening bracket, but not before the closing bracket.

TEXT FORMATTING TAGS

Tag	Description
<b>	Defines bold text
<big>	Defines big text
<em>	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
<strong>	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text

CHARACTER ENTITIES

Some characters have a special meaning in HTML, like the less than sign (<) that defines the start of an HTML tag. If we want the browser to actually display these characters we must insert character entities in the HTML source.
A character entity has three parts: an ampersand (&), an entity name or a # and an entity number, and finally a semicolon (;).
To display a less than sign in an HTML document we must write: **&lt;** or **&#60;**
The advantage of using a name instead of a number is that a name is easier to remember. The disadvantage is that not all browsers support the newest entity names, while the support for entity numbers is very good in almost all browsers.
Note that the entities are case sensitive.

N-BREAKING SPACE

THE MOST COMMON CHARACTER ENTITY IN HTML IS THE NON-BREAKING SPACE.
Normally HTML will truncate spaces in your text. If you write 10 spaces in your text HTML will remove 9 of them. To add spaces to your text, use the &nbsp; character entity.

MOST COMMON CHARACTER ENTITIES

Result	Description	Entity Name	Entity Number
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	quotation mark	&quot;	&#34;
'	apostrophe	&apos; (does not work in IE)	&#39;

ADDITIONAL COMMONLY USED CHARACTER ENTITIES

Result	Description	Entity Name	Entity Number
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
§	section	&sect;	&#167;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
×	multiplication	&times;	&#215;
÷	division	&divide;	&#247;

THE ANCHOR TAG AND THE HREF ATTRIBUTE

HTML uses the <a> (anchor) tag to create a link to another document.
An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc.
The syntax of creating an anchor:
<a href="url">Text to be displayed</a>
The <a> tag is used to create an anchor to link, the href attribute is used to address the document to link to, and the words between the open and close of the anchor tag will be displayed as a hyperlink.
This anchor defines a link to **EEE 111** webpage:
<a href="http://faraday.ee.emu.edu.tr/eee111">Visit EEE 111</a>

THE TARGET ATTRIBUTE

With the target attribute, you can define **where** the linked document will be opened.
The line below will open the document in a new browser window:
<a href="http://faraday.ee.emu.edu.tr/eee111" target="_blank"> Visit EEE 111</a>

THE ANCHOR TAG AND THE NAME ATTRIBUTE

The name attribute is used to create a named anchor. When using named anchors we can create links that can jump directly into a specific section on a page, instead of letting the user scroll around to find what he/she is looking for.
Below is the syntax of a named anchor:
<a name="label">Text to be displayed</a>
The name attribute is used to create a named anchor. The name of the anchor can be any text you care to use.
The line below defines a named anchor:
<a href="#down">Bottom of the page</a>
You should notice that a named anchor is not displayed in a special way.
To link directly to the "down" section, add a # sign and the name of the anchor to the end of a URL, like this:
<a href="http://faraday.ee.emu.edu.tr/eee111#down">Jump to down section</a>
A hyperlink to the Useful Tips Section from WITHIN the file "firstpage.html" will look like this:
<A NAME="DOWN">DOWN IS HERE</A>

TABLES

TABLES ARE DEFINED WITH THE <TABLE> TAG. A TABLE IS DIVIDED INTO ROWS (WITH THE <TR> TAG), AND EACH ROW IS DIVIDED INTO DATA CELLS (WITH THE <TD> TAG). THE LETTERS TD STANDS FOR "TABLE DATA," WHICH IS THE CONTENT OF A DATA CELL. A DATA CELL CAN CONTAIN TEXT, IMAGES, LISTS, PARAGRAPHS, FORMS, HORIZONTAL RULES, TABLES, ETC.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

TABLES AND THE BORDER ATTRIBUTE

If you do not specify a border attribute the table will be displayed without any borders. Sometimes this can be useful, but most of the time, you want the borders to show.
To display a table with borders, you will have to use the border attribute:

```
<table border="1">
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
```

HEADINGS IN A TABLE

Headings in a table are defined with the <th> tag.

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How it looks in a browser:

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

EMPTY CELLS IN A TABLE

Table cells with no content are not displayed very well in most browsers.

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td></td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

Note that the borders around the empty table cell are missing (NB! Mozilla Firefox displays the border).

To avoid this, add a non-breaking space (&nbsp;) to empty data cells, to make the borders visible:

```
<table border="1">

<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>&nbsp;</td>
</tr>
</table>
```

How it looks in a browser:

row 1, cell 1	row 1, cell 2
row 2, cell 1	

TABLE TAGS

Tag	**Description**
<table>	Defines a table
<th>	Defines a table header
<tr>	Defines a table row
<td>	Defines a table cell
<caption>	Defines a table caption
<colgroup>	Defines groups of table columns
<col>	Defines the attribute values for one or more columns in a table
<thead>	Defines a table head
<tbody>	Defines a table body
<tfoot>	Defines a table footer

HTML supports ordered, unordered and definition lists

UNORDERED LISTS

An unordered list is a list of items. The list items are marked with bullets (typically small black circles).
An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

Here is how it looks in a browser:

- Coffee
- Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

ORDERED LISTS

An ordered list is also a list of items. The list items are marked with numbers.
An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

Here is how it looks in a browser:

1. Coffee
2. Milk

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

DEFINITION LISTS

A definition list is **not** a list of items. This is a list of terms and explanation of the terms.
A definition list starts with the <dl> tag. Each definition-list term starts with the <dt> tag. Each definition-list definition starts with the <dd> tag.

```
<dl>
<dt>Coffee</dt>
<dd>Black hot drink</dd>
<dt>Milk</dt>
<dd>White cold drink</dd>
</dl>
```

Here is how it looks in a browser:

```
Coffee
        Black hot drink
Milk
        White cold drink
```

Inside a definition-list definition (the <dd> tag) you can put paragraphs, line breaks, images, links, other lists, etc.

LIST TAGS

Tag	**Description**
<ol>	Defines an ordered list
<ul>	Defines an unordered list
<li>	Defines a list item
<dl>	Defines a definition list
<dt>	Defines a definition term
<dd>	Defines a definition description

WHAT IS CSS?

Cascading **S**tyle **S**heets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs,variations in display for different devices and screen sizes as well as a variety of other effects.

ADVANTAGES OF CSS

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

WHO CREATES AND MAINTAINS CSS?

CSS was invented by **Håkon Wium Lie** on October 10, 1994 and maintained through a group of people within the W3C called the CSS Working Group. The CSS Working Group creates documents called **specifications**. When a specification has been discussed and officially ratified by W3C members, it becomes a recommendation.

These ratified specifications are called recommendations because the W3C has no control over the actual implementation of the language. Independent companies and organizations create that software.

NOTE − The World Wide Web Consortium, or W3C is a group that makes recommendations about how the Internet works and how it should evolve.

CSS VERSIONS

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

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

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

CSS3 Modules

CSS3 Modules are having old CSS specifications as well as extension features.

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

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

- **Selector** − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
- **Property** - A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be *color*, *border* etc.
- **Value** - Values are assigned to properties. For example, *color* property can have value either *red* or *#F1F1F1* etc.

You can put CSS Style Rule Syntax as follows −

```
selector { property: value }
```

Example: You can define a table border as follows −

```
table{ border :1px solid #C00; }
```

Here table is a selector and border is a property and given value *1px solid #C00* is the value of that property.

You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one.

The Type Selectors

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings:

```
h1 {
  color: #36CFFF;
}
```

The Universal Selectors

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

```
* {
  color: #000000;
}
```

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

The Descendant Selectors

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

```
ul em {
  color: #000000;
}
```

THE CLASS SELECTORS

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {
  color: #000000;
}
```

This rule renders the content in black for every element with class attribute set to *black* in our document. You can make it a bit more particular. For example:

```
h1.black {
  color: #000000;
}
```

This rule renders the content in black for only <h1> elements with class attribute set to *black*.
You can apply more than one class selectors to given element. Consider the following example:

```
<p class="center bold">
  This para will be styled by the classes center and bold.
</p>
```

THE ID SELECTORS

You can define style rules based on the *id* attribute of the elements. All the elements having that *id* will be formatted according to the defined rule.

```
#black {
  color: #000000;
}
```

This rule renders the content in black for every element with *id* attribute set to *black* in our document. You can make it a bit more particular. For example −

```
h1#black {
  color: #000000;
}
```

This rule renders the content in black for only <h1> elements with *id* attribute set to *black*.
The true power of *id* selectors is when they are used as the foundation for descendant selectors, For example:

```
#black h2 {
  color: #000000;
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie with in tags having *id* attribute set to *black*.

THE CHILD SELECTORS

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

```
body > p {
  color: #000000;
}
```

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

THE ATTRIBUTE SELECTORS

You can also apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of *text* −

```
input[type = "text"]{
  color: #000000;
}
```

The advantage to this method is that the <input type = "submit" /> element is unaffected, and the color applied only to the desired text fields.
There are following rules applied to attribute selector.

- **p[lang]** - Selects all paragraph elements with a *lang* attribute.
- **p[lang="fr"]** - Selects all paragraph elements whose *lang* attribute has a value of exactly "fr".
- **p[lang~="fr"]** - Selects all paragraph elements whose *lang* attribute contains the word "fr".
- **p[lang|="en"]** - Selects all paragraph elements whose *lang* attribute contains values that are exactly "en", or begin with "en-".

MULTIPLE STYLE RULES

You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example −

```
h1 {
  color: #36C;
  font-weight:  normal;
  letter-spacing:  .4em;
  margin-bottom: 1em;
  text-transform: lowercase;
}
```

Here all the property and value pairs are separated by a **semi colon (;)**. You can keep them in a single line or multiple lines. For better readability we keep them into separate lines.
For a while, don't bother about the properties mentioned in the above block..

GROUPING SELECTORS

You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example −

```
h1,  h2,  h3  {
  color: #36C;
  font-weight:  normal;
  letter-spacing:  .4em;
  margin-bottom: 1em;
  text-transform: lowercase;
}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.
You can combine the various *id* selectors together as shown below −

```
#content, #footer, #supplement {
  position: absolute;
  left:  510px;
  width: 200px;
}
```

Conclusion: *Hence we have created Restaurant website using HTML,CSS.*

ASSIGNMENT NO – 04

AIM:

Implement an application in Java Script using following:

a. Design UI of application using HTML, CSS etc.

b. Include Java script validation

c. Use of prompt and alert window using Java Script

e.g., Design and implement a simple calculator using Java Script for operations like addition, multiplication, subtraction, division, square of number etc.

a. Design calculator interface like text field for input and output, buttons for numbers and operators etc.

b. Validate input values

c. Prompt/alerts for invalid values etc.

LEARNING OBJECTIVES:

Understand important of validation.

Understand Basic Syntax of JavaScript.

Understand Basic Syntax Of JQuery.

THEORY:

JavaScript

JavaScript is a very powerful **client-side scripting language**. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage more lively and interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development.

How to Run JavaScript?

Being a scripting language, **JavaScript cannot run on its own. In fact, the browser is responsible for running JavaScript code**. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it. The main advantage of JavaScript is that **all modern web browsers support** JavaScript. So, you do not have to worry whether your site visitor uses Internet Explorer, Google Chrome, Firefox or any other browser. JavaScript will be supported. Also, JavaScript **runs on any operating system** including Windows, Linux or Mac. Thus, JavaScript overcomes the main disadvantages of VBScript (Now deprecated) which is limited to just IE and Windows.

Tools You Need

To start with, you need a text editor to write your code and a browser to display the web pages you develop. You can use text editor of your choice including Notepad++, Visual Studio Code, Sublime Text, Atom or any other text editor you are comfortable with. You can use any web browser including Google Chrome, Firefox, Microsoft Edge, Internet Explorer etc.

A Simple JavaScript Code

You should place all your JavaScript code within **<script> tags** (<script> and </script>) if you are keeping your JavaScript code within the HTML document itself. This helps your browser distinguish your JavaScript code from the rest of the code. As there are other client side scripting languages (Example: VBScript), it is highly recommended that you specify the scripting language you use. You have to use the type attribute within the <script> tag and set its value to text/javascript like this:

<script type="text/javascript">

Example:
This code is editable. Click Run to Execute

```
<html>
<head>
   <title>My First JavaScript code!!!</title>
   <script type="text/javascript">
      alert("Welcome!!! You are now learning JavaScript.");
   </script>
</head>
<body>
</body>
</html>
```

Note: type="text/javascript" is not necessary in HTML5. Following code will work.

```
<html>
<head>
   <title>My First JavaScript code!!!</title>
   <script>
      alert("Welcome!!! You are now learning JavaScript.");
   </script>
</head>
<body>
</body>
</html>
```

USING VARIABLES IN JAVASCRIPT

Variables are used to **store values** (name = "John") **or expressions** (sum = x + y). Before using a variable, you first need to declare it. You have to use the keyword **var** to declare a variable like this:

```
var name;
```

You can assign a value to the variable either while declaring the variable or after declaring the variable.

```
var name = "John";
```

OR

```
var  name;

name = "John";
```

Naming Variables

Though you can name the variables as you like, it is a good programming practice to give descriptive and meaningful names to the variables. Moreover, variable names should start with a letter and they are case sensitive. Hence the variables studentname and studentName are different because the letter n in name is different (n and N).

LEARN ARRAYS IN JAVASCRIPT

An array is an object that can store a **collection of items**. Arrays become really useful when you need to store large amounts of data of the same type. Suppose you want to store details of 500 employees. If you are using variables, you will have to create 500 variables whereas you can do the same with a single array. You can access the items in an array by referring to its **indexnumber** and the index of the first element of any array is zero.

Creating an Array

You can create an array in JavaScript as given below.

```
var students = ["John", "Ann", "Kevin"];
```

Here, you are initializing your array as and when it is created with values “John”, “Ann” and “Kevin”.The index of “John”, “Ann” and “Kevin” is 0, 1 and 2 respectively. If you want to add more elements to the students array, you can do it like this:

```
students[3] = "Emma";
students[4] = "Rose";
```

You can also create an array using Array constructor like this:

```
var students = new Array("John", "Ann", "Kevin");
```

OR

```
var students = new Array();

students[0] = "John";

students[1] = "Ann";

students[2] = "Kevin";
```

HOW TO USE LOOPS IN JAVASCRIPT

Loops are useful when you have to execute the same lines of code repeatedly, for a specific number of times or as long as a specific condition is true. Suppose you want to type a ‘Hello’ message 100 times in your webpage. Of course, you will have to copy and paste the same line 100 times. Instead if you use loops, you can complete this task in just 3 or 4 lines.

Different Types of Loops

There are mainly four types of loops in JavaScript.

1. for loop
2. for/in loop (explained later)
3. while loop
4. do…while loop

FOR LOOP

Syntax:

```
for(statement1; statement2; statment3)

{

lines of code to be executed

}
```

1. The statement1 is executed first even before executing the looping code. So, this statement is normally used to assign values to variables that will be used inside the loop.
2. The statement2 is the condition to execute the loop.
3. The statement3 is executed every time after the looping code is executed.

WHILE LOOP

Syntax:

```
while(condition)

{

lines of code to be executed

}
```

The “while loop” is executed as long as the specified condition is true. Inside the while loop, you should include the statement that will end the loop at some point of time. Otherwise, your loop will never end and your browser may crash.

DO…WHILE LOOP

Syntax:

```
do

{

block of code to be executed

} while (condition)
```

The do…while loop is very similar to while loop. The only difference is that in do…while loop, the block of code gets executed once even before checking the condition.

Validation:

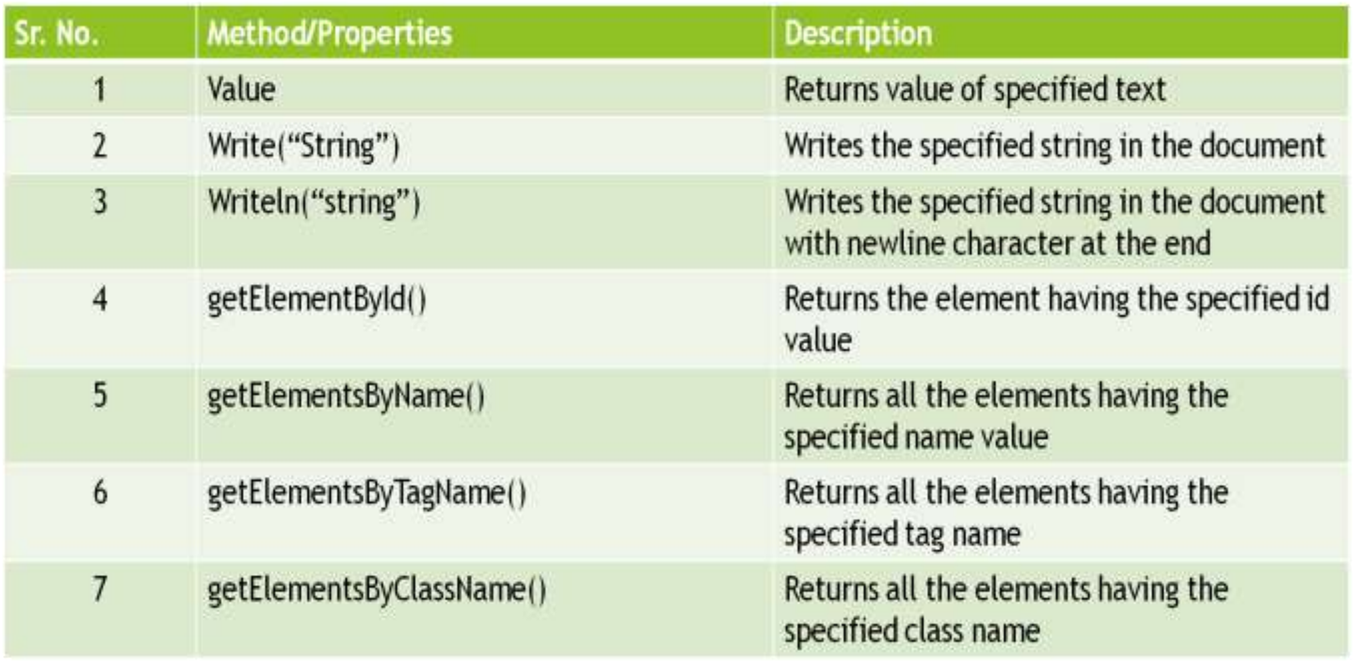
When client enters the all necessary data and press the submit button form validation is done at server side If data entered by a client is incorrect or missing,the server needs to send all data back to the client and request for resubmission of form with correct information. This is really a lengthy process which puts a lot of load(burden) on the server.

So, JavaScript provides a way to validate form's data on the client's side itself before sending it to the web server. Form validation performs two functions-

- Basic Validation –First of all the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for the data

- Data Format Validation − Secondly, the data that is entered must be checked for correct format and its value. The code must include appropriate logic to test correctness of data.

Properties and Methods of Document Object:



Sr. No.	Method/Properties	Description
1	Value	Returns value of specified text
2	Write("String")	Writes the specified string in the document
3	Writeln("string")	Writes the specified string in the document with newline character at the end
4	getElementById()	Returns the element having the specified id value
5	getElementsByName()	Returns all the elements having the specified name value
6	getElementsByTagName()	Returns all the elements having the specified tag name
7	getElementsByClassName()	Returns all the elements having the specified class name

JQuery

What is jQuery?

jQuery is a lightweight, "write less, do more", JavaScript library.
The purpose of jQuery is to make it much easier to use JavaScript on your website.
jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.
The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Advantages of JQuery

- ts light weight when compared to other javascript frameworks.

- it has a wide **range** of **plugins** available for various specific needs.
- it is easier for a designer to learn jQuery as it uses familiar CSS syntax. jQuery is Javascript for Designers

Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google

Downloading jQuery
There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

Both versions can be downloaded from jQuery.com.
The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section):

```
<head>
<script src="jquery-3.3.1.min.js"></script>
</head>
```

Tip: Place the downloaded file in the same directory as the pages where you wish to use it.

Do you wonder why we do not have type="text/javascript" inside the <script> tag?

This is not required in HTML5. JavaScript is the default scripting language in HTML5 and in all modern browsers!

jQuery CDN
If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).
Both Google and Microsoft host jQuery.
To use jQuery from Google or Microsoft, use one of the following:

Google CDN:

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
</head>
```

Microsoft CDN:

```
<head>
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>
</head>
```

CONCLUSIONS:
Hence, we have studied validation using JavaScript and JQuery.

ASSIGNMENT NO - 09

AIM:

Design an application using Angular JS.
e.g., Design registration (first name, last name, username, password) and login page using Angular JS.

LEARNING OBJECTIVES:

To understand implementation of AngularJS.

THEORY:

ANGULARJS INTRODUCTION

AngularJS is a **JavaScript framework**. It can be added to an HTML page with a <script> tag.
AngularJS extends HTML attributes with **Directives**, and binds data to HTML with **Expressions**.

ANGULARJS IS A JAVASCRIPT FRAMEWORK

AngularJS is a JavaScript framework. It is a library written in JavaScript.
AngularJS is distributed as a JavaScript file, and can be added to a web page with a script tag:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
```

ANGULARJS EXTENDS HTML

AngularJS extends HTML with **ng-directives**.
The **ng-app** directive defines an AngularJS application.
The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.
The **ng-bind** directive binds application data to the HTML view.

ANGULARJS EXAMPLE

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>

</body>
</html>
```

Example explained:

AngularJS starts automatically when the web page has loaded.
The **ng-app** directive tells AngularJS that the <div> element is the "owner" of an AngularJS **application**.
The **ng-model** directive binds the value of the input field to the application variable **name**.
The **ng-bind** directive binds the **innerHTML** of the <p> element to the application variable **name**.

ANGULARJS DIRECTIVES

As you have already seen, AngularJS directives are HTML attributes with an **ng** prefix.
The **ng-init** directive initializes AngularJS application variables.

ANGULARJS EXAMPLE

```
<div ng-app="" ng-init="firstName='John'">

<p>The name is <span ng-bind="firstName"></span></p>

</div>
```

Alternatively with valid HTML:

ANGULARJS EXAMPLE

```
<div data-ng-app="" data-ng-init="firstName='John'">

<p>The name is <span data-ng-bind="firstName"></span></p>

</div>
```

You can use **data-ng-**, instead of **ng-**, if you want to make your page HTML valid.
You will learn a lot more about directives later in this tutorial.

ANGULARJS EXPRESSIONS

AngularJS expressions are written inside double braces: **{{ expression }}**.
AngularJS will "output" data exactly where the expression is written:

ANGULARJS EXAMPLE

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="">
  <p>My first expression: {{ 5 + 5 }}</p>
</div>

</body>
</html>
```

AngularJS expressions bind AngularJS data to HTML the same way as the **ng-bind** directive.

ANGULARJS EXAMPLE

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="">
  <p>Name: <input type="text" ng-model="name"></p>
  <p>{{name}}</p>
</div>
```

```
</body>
</html>
```
You will learn more about expressions later in this tutorial.

ANGULARJS APPLICATIONS

AngularJS **modules** define AngularJS applications.
AngularJS **controllers** control AngularJS applications.
The **ng-app** directive defines the application, the **ng-controller** directive defines the controller.

ANGULARJS EXAMPLE

```
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
   $scope.firstName= "John";
   $scope.lastName= "Doe";
});
</script>
```
AngularJS modules define applications:

ANGULARJS MODULE

```
var app = angular.module('myApp', []);
```
AngularJS controllers control applications:

ANGULARJS CONTROLLER

```
app.controller('myCtrl', function($scope) {
   $scope.firstName= "John";
   $scope.lastName= "Doe";
});
```

ANGULARJS EXPRESSIONS

AngularJS binds data to HTML using **Expressions**.

ANGULARJS EXPRESSIONS

AngularJS expressions can be written inside double braces: {{ ***expression*** }}.

AngularJS expressions can also be written inside a directive: ng-bind="***expression***".

AngularJS will resolve the expression, and return the result exactly where the expression is written.

AngularJS expressions are much like **JavaScript expressions:** They can contain literals, operators, and variables.

Example {{ 5 + 5 }} or {{ firstName + " " + lastName }}

EXAMPLE

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div ng-app="">
  <p>My first expression: {{ 5 + 5 }}</p>
</div>

</body>
</html>
```

If you remove the ng-app directive, HTML will display the expression as it is, without solving it:

EXAMPLE

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>

<div>
  <p>My first expression: {{ 5 + 5 }}</p>
</div>

</body>
</html>
```

You can write expressions wherever you like, AngularJS will simply resolve the expression and return the result.

Example: Let AngularJS change the value of CSS properties.

Change the color of the input box below, by changing its value:

EXAMPLE

```
<div ng-app="" ng-init="myCol='lightblue'">

<input style="background-color:{{myCol}}" ng-model="myCol">

</div>
```

ANGULARJS NUMBERS

AngularJS numbers are like JavaScript numbers:

EXAMPLE

```
<div ng-app="" ng-init="quantity=1;cost=5">

<p>Total in dollar: {{ quantity * cost }}</p>

</div>
```

Same example using ng-bind:

EXAMPLE

```
<div ng-app="" ng-init="quantity=1;cost=5">
<p>Total in dollar: <span ng-bind="quantity * cost"></span></p>
</div>
```

CONCLUSION: Hence we created Application using AngularJS.