INTRODUCTION TO JAVA SCRIPT:

• Web pages are two types

i. **Static web page:** there is no specific interaction with the client

ii. **Dynamic web page:** web page which is having interactions with client and as well as validations can be added.

• Script means small piece of Code.

• Scripting Language is a high-level programming language, whose programs are interpreted by another program at run time rather than compiled by the computer processor.

• BY using JavaScript we can create interactive web pages. It is designed to add interactivity to HTML pages.

• Previously JavaScript was known as LiveScript, but later it was changed to JavaScript. As Java was very popular at that time and introducing a new language with the similarity in names would be beneficial they thought.

• Scripting languages are of 2 types.

▪ **client-side** scripting languages

▪ **servers-side** scripting languages

• In general Client-side scripting is used for performing

simple validations at client-side;

Server-side scripting is used for database verifications.

• Examples:

Client-side scripting languages: VBScript, JavaScript and Jscript.

Server-side scripting languages: ASP, JSP, Servlets and PHP etc.

• Simple HTML code is called static web page, if you add script to HTML page it is called dynamic page.

• Netscape Navigator developed JavaScript and Microsoft' s version of JavaScript is Jscript.

Features of JavaScript:

• JavaScript is a lightweight, interpreted programming language means that scripts execute without preliminary compilation.

• It is an Object-based Scripting Language.

• Java script is case sensitive language

• Complementary to and integrated with Java.

• Open and cross-platform.

Advantages of JavaScript:

1. Less server interaction:

You can validate the user input before sending the page off to the server. This saves server traffic, which means less load on server.

2. Immediate feedback to the visitors or end-users:

If you submit a form if there is any error in form filling immediately visitor get the feedback, because validation performed at client side.

3. Can put dynamic text into an HTML page
4. Used to Validate form input data
5. Java script code can react to user events
6. Can be used to detect the visitor' s browser

Limitations of JavaScript:

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.

JAVA Vs JAVASCRIPT:

JAVA	JAVASCRIPT
1. Object Oriented Programming Language	Object based Scripting Language
2. Platform Independent	Browser Dependant
3. It is both compiled and interpreted	It is interpreted at runtime
4. It is used to create server side applications and standalone programming	It is used to make the web pages more interactive
5. Java is a strongly typed language	JavaScript is not strongly typed(Loosely Typed)
6. Developed by sun Microsystems	Developed by Netscape
7. Java Programs can be standalone	JavaScript must be placed inside an HTML document to function

Embedding JavaScript in an HTML Page:

Embed a JavaScript in an HTML document by using <script>
and </script> html tags.

Syntax:

```
<script ...>
JavaScript code
</script>
```

<script > tag has the following attributes.

Type	Refers to the MIME (Multipurpose Internet Mail Extensions) type of the script.
Language	This attribute specifies what scripting languageyou are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out theuse of this attribute.

Example:

```
<html>
<body>
<script language="javascript" type="text/javascript">
document.write ("Hello World!")
</script>
</body>
</html>
```

Alert Message Example:

```
<html>
<head>
<title>My First JavaScript code!!!</title>
<script type="text/javascript">
alert("Hello World!");
</script>
</head&gt;
<body>
</body>
</html>
```

```
Alert(“ Hello World” );
Var d= Confirm(“ Do you want to continue?” );
Var i=prompt(“ Enter your name:” ,” SRGEC” );
```

Comments in JavaScript:

JavaScript supports both C-style and C++-style comments.

Thus:

• Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.

• Any text between the characters /* and */ is treated as a comment. This may span multiple lines.

VARIABLES:

• Like any programming language JavaScript has variables.

• Stores data items used in the script.

• Strict rules governing how you name your variables (Much like other languages):

Naming Conventions for Variables:

• Variable names must begin with a alphabet([a-z]/[A-Z]) or underscore;

• You can' t use spaces in names

• Names are case sensitive so the variables fred, FRED and frEd all refer to different variables,

• It is not a good idea to name variables with similar names

• You can' t use a reserved word as a variable name,
e.g. var.

Creating Variables

• Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the var keyword as follows.

```
<script type="text/javascript">
var name;
var rollno;
</script>
```

• Storing a value in a variable is called variable initialization. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable.

```
<script type="text/javascript">
var name = " Aziz" ;
var rollno=501;
</script>
```

Scope of Variables in JavaScript:

The scope of a variable is the region of your program in which it is defined and is accessible. JavaScript variables have only two scopes.

• **Global Variables:** A global variable has global scope which means it can be defined and used anywhere in your JavaScript code.

• **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Automatically Global:

• If you assign a value to a variable that has not been declared, it will automatically become a GLOBAL variable.

• This code example will declare a global variable price, even if the value is assigned inside a function.

Example:

```
myFunction();
// code here can use price
function myFunction()
{
price = 250; //has Global scope
}
```

Example:

```
<script language="javascript" type="text/javascript">
var collegename="GEC college"; //global scope
function function1()
{
var studentname="Anand";//local scope
document.write("<center>"+studentname+"</center><br
>");
document.write("<center>"+collegename+"</center><br>
");//global scope
}
function function2()
{
var branchname="Information Technology";//local
scope
document.write("<center>"+branchname+"</center><br
>");
document.write("<center>"+collegename+"</center><br>
");//global scope
document.write("<center>"+studentname+"</center>");/
/not displayed because of local scope
}
function1();
function2();
</script>
```

DATA TYPES:

• JavaScript has only four types of data

- Numeric
- String
- Boolean
- Null

- **Numeric :**
  - Integers such as 108 or 1120 or 2016
  - Floating point values like 23.42, -56.01 and 2E45.
  - No need to differentiate between.
  - In fact variables can change type within program.
- **String:**
  - A String is a Collection of character.
  - All of the following are strings:

  "Computer", "Digital" , "12345.432".
  - Put quotes around the value to a assign a variable:

  name = "Uttam K.Roy";
- **Boolean:**
  - Variables can hold the values true and false.
  - Used a lot in conditional tests (later).
- **Null:**
  - Used when you don' t yet know something.
  - A null value means one that has not yet been decided.
  - It does not mean nil or zero and should NOT be used in that way.

FUNCTIONS:

- A function is a group of reusable code which can be called anywhere in your program.
- This eliminates the need of writing the same code again and again.
- It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.
- Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions.
- We were using these functions again and again, but they had been written in core JavaScript only once.
- JavaScript allows us to write our own functions as well.
- Function Definition
- Before we use a function, we need to define it.
- The most common way to define a function in JavaScript is
- By using keyword function, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax:

```
<script type="text/javascript">
function functionname(parameter-list)
{
statements
}
</script>
```

Example:

```
<html>
<head>
<title>My First JavaScript code!!!</title>
<script type="text/javascript">
function sayHello()
{
document.write("Hello Anand How
are you...?");
}
sayHello();//calling function
</script>
</head>;
<body>
</body>
</html>
```

Calling a Function:

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
<head>
<title>Calling a function</title>
<style type='text/css'>
{
text-align:center;
}
</style>
<script type="text/javascript">
function sayHello()
{
var name=form.name.value;
document.write("Hello "+name+" Good
Morning");
}
</script>
</head>
<body>

<p>Please enter you name and click
the button to get wishes
```

```
</p></br>
<form name='form'>
<input type='text' name='name'
placeholder='Enter Name'><br><br>
<input type='button' value='click here' onclick='sayHello();'>
</form>
</body>
</html>
```

OPERATORS:

JavaScript supports the following types of operators.

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical (or Relational) Operators
- Conditional (or ternary) Operators

Arithmetic Operators:

- JavaScript supports the following arithmetic operators:
- Assume variable A holds 10 and variable B holds 20,

Operator	Description	Example	Result
+	Adds two numbers or joins two strings	20 + 10	30
-	Subtracts right operand from left	20 - 10	10
*	Multiplies two numbers	20 * 10	200
/	Divides left by right operand (quotient)	20 / 10	2
%	Modulus (returns remainder)	20 % 10	0
++	Increments value by 1	int m = 20; ++m;	21
--	Decrements value by 1	int m = 20; --m;	19

Assignment Operators:

Operator	Description	Example	Result
=	Assigns value from right to left	int m = 20;	m = 20
+=	Adds and assigns	m += 10;	m = 30
-=	Subtracts and assigns	m -= 5;	m = 15
*=	Multiplies and assigns	m *= 10;	m = 200
/=	Divides and assigns (quotient)	m /= 10;	m = 2
%=	Modulus and assigns (remainder)	m %= 10;	m = 0

Comparison Operators:

Operator	Description	Example	Result
==	Checks if values are equal	20 == 10	false
!=	Checks if values are not equal	20 != 10	true
>	Checks if left > right	20 > 10	true
>=	Checks if left >= right	20 >= 10	true
<	Checks if left < right	20 < 10	false
<=	Checks if left <= right	20 <= 10	false

Logical (or Relational) Operators:

Operator	Description	Example	Result
&&	Logical AND – true if **both** conditions are true	true && false	false
`		`	Logical OR – true if **at least one** condition is true
!	Logical NOT – inverts the value	!true	false

Conditional (or ternary) Operators:

Operator	Description	Example	Result
?:	Returns one of two values based on a condition	(20 > 10) ? 20 : 10	20

CONTROL FLOW STATEMENTS:

These statements allow you to control the flow of your program' s execution based upon conditions known only during run time.

In JavaScript we have the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

The if Statement

Syntax

```
if (condition)
{
block of code to be executed if the condition is true
}
```

The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

```
if (condition)
{
block of code to be executed if the condition is true
}
else
{
block of code to be executed if the condition is false
}
```

Example:

```
<HTML>
<HEAD>
<script>
function check()
{
var age=form.age.value;
if(age>=18)
{
alert("You are eligible for vote");
}
else
```

```
{
alert("You are not eligible for vote");
}
}
</script>
</HEAD>
<BODY>
<form name='form'>
<p>Enter your age and check whether you are
eligible for vote or not?</p><br>
<input type='text' name='age'><br><br>
<input type='button' value='check eligibility' onclick='check();'>
</form>
</BODY>
</HTML>
```

The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

Syntax:

```
if (condition1)
{
block of code to be executed if condition1 is true
}
else if (condition2)
{
block of code to be executed if the condition1 is false and
condition2 is true
}
else
{
block of code to be executed if the condition1 is false and
condition2 is false
}
```

Example:

```
<HTML>
<HEAD>
<script>
function check()
{
var percentage=form.percentage.value;
if(percentage>=90&&percentage<=100)
{
alert("Your grade is A+");
```

```
}
else if(percentage>=75&&percentage<90)
{
alert("Your grade is A");
}
else if(percentage>=60&&percentage<75)
{
alert("Your grade is B");
}
else if(percentage>=40&&percentage<60)
{
alert("Your grade is C");
}
else if(percentage>100)
{
alert("Wrong details.....");
}
else
{
alert("You are failed");
}
}
</script>
</HEAD>
<BODY>
<form name='form'>
<p>Enter your marks to know your
grade</p><br>
<input type='text' name='percentage'
placeholder='EX:70.45/70'><br><br>
<input type='button' value='check grade' onclick='check();'>
</form>
</BODY>
</HTML>
```

Switch Statement:

Use the switch statement to select one of many blocks of code to be executed.

Syntax:

```
switch(expression) {
case 1:
code block
break;
case 2:
code block
break;
.
.
case n:
code block
break;
default:
default code block
}
```

This is how it works:

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

Example:

```
<HTML>
<HEAD>
<script>
function check()
{
var category=form.category.value;
switch(category)
{
case "SC":
alert("50 vanacies");
break;
case "OC":
alert("5 vacanices");
break;
case "BC":
alert("30 vanacies");
break;
case "ST":
alert("45 vanacies");
break;
case "OBC":
```

```
alert("20 vanacies");
break;
default:
alert("please enter valid category");
break;
}
}
</script>
</HEAD>
<BODY>
<form name='form'>
<p>Please enter your category to check no of
vacanices</p><br>
<input type='text' name='category'
placeholder='EX:OC/BC/OBC/SC/ST'><br><br>
<input type='button' value='Check Vacancies' onclick='check();'>
</form>
</BODY>
</HTML>
```

The While Loop

Syntax:

Example:

Write a JavaScript code to print 0 to n even numbers using while loop.

```
<HTML>
<HEAD>
<script>
function check()
{
while (condition)
{
code block to be executed
}
var number=form.number.value;
var i=1;
while(i<=number)
{
if(i%2==0)
document.write("<center>"+i+"</center><br>");
i++;
}
}
</script>
```

```
</HEAD>
<BODY>
<form name='form'>
<p>Find o to n even numbers</p><br>
<input type='text' name='number'><br><br>
<input type='button' value='Get Even Numbers' onclick='check();'>
</form>
</BODY>
</HTML>
```

The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:

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

The for Loop:

The for loop has the following syntax:

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

Statement 1 is executed before the loop (the code block) starts.
Statement 2 defines the condition for running the loop (the code block).
Statement 3 is executed each time after the loop (the code block) has been executed.

Write a JavaScript code to print 1 to 10 even numbers using

Example 2:

```
<HTML>
<HEAD>
<script>
function check()
{
var number=form.number.value;
var fact=1;
for(var i=1;i<=number;i++)
{
fact=fact*i;
}
```

```
alert("factorial of "+number+"is "+fact);
}
</script>
</HEAD>
<BODY>
<form name='form'>
<p>Enter a number to know the factorial</p><br>
<input type='text' name='number'><br><br>
<input type='button' value='Get Factorial' onclick='check();'>
</form>
</BODY>
</HTML>
```

OBJECTS IN JAVA SCRIPT: (BUILT-IN OBJECTS)

- ➢ An Object is a thing.
- ➢ There are pre defined objects and user defined objects in Javascript.
- ➢ Each object can have properties and methods:
  - A property tells you something about an object.
  - A method performs an action
- ➢ The following are some of the Pre defined objects/Built- in Objects.
  - Document
  - Window
  - Browser/Navigator
  - Form
  - String
  - Math
  - Array
  - Date

HTML DOM

The way document content is accessed and modified is called the Document Object Model, or DOM.

In the HTML DOM (Document Object Model), everything is a node:

- The document itself is a document node
- All HTML elements are element nodes
- All HTML attributes are attribute nodes
- Text inside HTML elements are text nodes
- Comments are comment nodes

The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

- **Window object** − Top of the hierarchy. It is the outmost element of the object hierarchy.
- **Document object** − Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.
- **Form object** − Everything enclosed in the <form>...</form> tags sets the form object.

- **Form control elements** − The form object contains all the elements defined for that object such as text fields,buttons, radio buttons, and checkboxes.

Here is a simple hierarchy of a few important objects −

THE DOCUMENT OBJECT

- When an HTML document is loaded into a web browser, it becomes a document object.
- The document object is the root node of the HTML document and the "owner" of all other nodes:

(element nodes, text nodes, attribute nodes, and comment nodes).

- The document object provides properties and methods to access all node objects, from within JavaScript.
- Tip: The document is a part of the Window object and can be accessed as window.document.

Properties

Property	Description
alinkColor	Sets the color of **active links** (i.e., links that are being clicked).
bgColor	Sets the **background color** of the webpage. Usually used inside <body>.
title	The **title** of the current document, as specified in the <title> tag.
URL	The **URL/location** of the current document.
vlinkColor	Sets the color of **visited links**, specified inside the <body> tag.
fgColor	Sets the **foreground (text) color** of the webpage. Usually used in <body>.

Methods

Method	Description
getElementById(id)	Finds and returns an element with the specified **ID**.
getElementsByTagName(name)	Returns a **collection** of all elements with the given **tag name**.
getElementsByClassName(name)	Returns a **collection** of elements with the specified **class name**.
write(text)	Writes the given **text** directly into the HTML output stream.
writeln(text)	Same as write() but adds a **newline** at the end of the output.

WINDOW OBJECT:

- The window object is supported by all browsers. It represents the browser's window.
- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object.
- Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object:

  window.document.getElementById("header");

  is the same as:

```
document.getElementById("header");
```

Properties

- **defaultStatus** - This is the default message that is loaded into the status bar when the window loads.
- **opener** The object that caused the window to open.
- **status** - The status bar is the bar on the lower left side of the browser and is used to display temporary messages
- **length** - The number of frames that the window contains.

Methods

- **alert("message")** - The string passed to the alert function is displayed in an alert dialog box.
- **open**("URLname","Windowname",["options"]) - A new window is opened with the name specified by the second parameter.
- **close()** - This function will close the current window or the named window.
- **confirm**("message") The string passed to the confirm function is displayed in the confirm dialog box.
- **prompt**("message","defaultmessage") - A prompt dialog box is displayed with the message passed as the prompt question or phrase.

Example:

```
<HTML>
<HEAD>
<script>
function funalert()
{
window.alert("Hello be alert....");
}
function funopen()
{
window.open("http://www.gmail.com");
}
function funprompt()
{
window.prompt("Do you want to exit?");
}
function funconfirm()
{
window.confirm("Do you want to exit?");
}
function funclose()
{
window.close();
}
```

```
</script>
</HEAD>
<BODY>
<form>
<input type='button' value='click here for alert()'
onclick='funalert()'>
<input type='button' value='click here for open()'
onclick='funopen()'>
<input type='button' value='click here for
prompt()' onclick='funprompt()'>
<input type='button' value='click here for
cofirm()' onclick='funconfirm()'>
<input type='button' value='click here for close()'
onclick='funclose()'>
</form>
</BODY>
</HTML>
```

FORM OBJECT:

Properties

- **action** - The action attribute of the Top of Form element
- **length** - Gives the number of form controls in the form
- **method**- The method attribute of the Top of Form element
- **name** - The name attribute of the Top of Form element
- **target** - The target attribute of the Top of Form element

Methods

- **reset()-** Resets all form elements to their default values
- **submit()-** Submits the form

Properties of Form Elements

The following table lists the properties of form elements

- checked - Returns true when checked or false when not
- form - Returns a reference to the form in which it is part of
- length - Number of options in the <select> element.
- name - Accesses the name attribute of the element
- selectedIndex - Returns the index number of the currently selected item
- value - the value attribute of the element or content of a text input

STRING OBJECT:

String The string object allows you to deal with strings of text.

Properties

- **length** - The number of characters in the string.

Methods:

- **charAt(index)** - Returns a string containing the character at the specified location.

- **indexOf(pattern)** - Returns -1 if the value is not found and returns the index of the first character of the first string matching the pattern in the string.
- **indexOf(pattern, index)** - Returns -1 if the value is not found and returns the index of the first character of the first string matching the pattern in the string. Searching begins at the index value in the string.
- **lastIndexOf(pattern)** - Returns -1 if the value is not found and returns the index of the first character of the last string matching the pattern in the string.
- **lastIndexOf(pattern, index)** - Returns -1 if the value is not found and returns the index of the first character of the last string matching the pattern in the string. Searching begins at the index value in the string.
- **split(separator)** - Splits a string into substrings based on the separator character.
- **substr(start, length)** - Returns the string starting at the "start" index of the string Continuing for the specified length of characters unless the end of the string is found first.
- **substring(start, end)** - Returns the string starting at the "start" index of the string and ending at "end" index location, less one.
- **toLowerCase()** - Returns a copy of the string with all characters in lower case.
- **toUpperCase()** - Returns a copy of the string with all characters in upper case.

Example:

```
<HTML>
<HEAD>
<script>
var txt = ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
document.writeln(sln);
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
document.write("<center>"+pos+"</center>");
document.write("<center>"+str.toUpperCase()+
"</center>");
document.write("<center>"+str.toLowerCase()+"</center
>");
document.write("<center>"+str.lastIndexOf("locate")+"</
center>");
document.write("<center>"+str.split(" ")+"</center>");
document.write("<center>"+str.substr(5,10)+"</center>"
);
</script>
</HEAD>
<BODY>
</BODY>
</HTML>
```

ARRAY OBJECT:

The Array object is used to store multiple values in a single variable.

Properties:

- **length** - Sets or returns the number of elements in an array

Methods:

- **concat()** - Joins two or more arrays, and returns a copy of the joined arrays
- **indexOf()** - Search the array for an element and returns its position
- **join()** - Joins all elements of an array into a string
- **lastIndexOf()** - Search the array for an element, starting at the end, and returns its position
- **pop()** - Removes the last element of an array, and returns that element
- **push()-** Adds new elements to the end of an array, and returns the new length
- **reverse()** - Reverses the order of the elements in an array
- **shift() -** Removes the first element of an array, and returns that element
- **slice()** - Selects a part of an array, and returns the new array
- **sort()** - Sorts the elements of an array
- **splice()** - Adds/Removes elements from an array
- **toString()** - Converts an array to a string, and returns the result.

Example:

```
<HTML>
<HEAD>
<script>
var cars = new Array("Saab", "Volvo", "BMW");
var bikes = new Array("Pulsar", "Honda", "FZ");
document.write("<center>"+cars.concat(bikes)+"</center>");
document.write("<center>"+cars.indexOf("Volvo")+"</center>");
bikes.push("Bajaj");
document.write("<center>"+bikes+"</center>");
bikes.reverse();
document.write("<center>"+bikes+"</center>");
cars.sort();
document.write("<center>"+cars+"</center>");
</script>
</HEAD>
<BODY>
</BODY>
</HTML>
```

BROWSER OBJECT/NAVIGATOR OBJECT:

It is used to obtain information about client browser.

Properties

- ➢ **appName**- Returns Browser Name
- ➢ **appVersion**- Returns Browser Version
- ➢ **appUserAgent**- It Returns User Agent
- ➢ **plugins-** It will display Plugins.
- ➢ **mimeTypes –** It will Return Mime type supported by browser

DATE OBJECT:

The Date object is used to work with dates and times

- getDate() - Get the day of the month. It is returned as a value between 1 and 31.
- getDay() - Get the day of the week as a value from 0 to 6
- getHours() - The value returned is 0 through 23.
- getMinutes() - The value returned is 0 through 59.
- getMonth() - Returns the month from the date object as a value from 0 through 11.
- getSeconds() - The value returned is 0 through 59.
- getTime() - The number of milliseconds since January 1, 1970.
- getYear() - Returns the numeric four digit value of the year.
- setDate(value) - Set the day of the month in the date object as a value from 1 to 31.
- setHours(value) - Set the hours in the date object with a value of 0 through 59.
- setMinutes(value) - Set the minutes in the date object with a value of 0 through 59.
- setMonth(value) - Set the month in the date object as a value of 0 through 11.
- setSeconds(value) - Set the seconds in the date object with a value of 0 through 59.
- setTime(value) - Sets time on the basis of number of milliseconds since January 1, 1970.
- setYear(value) - Set the year in the date instance as a 4 digit numeric value.

Example:

```
<HTML>
<HEAD>
<script>
var d=new Date();
document.write("<center>"+d.getDate()+"</center>");
document.write("<center>"+d.getDay()+"</center>");
document.write("<center>"+d.getHours()+"</center>");
document.write("<center>"+d.getMinutes()+"</center>");
document.write("<center>"+d.getMonth()+"</center>");
document.write("<center>"+d.getYear()+"</center>");
document.write("<center>"+d.getTime()+"</center>");
</script>
</HEAD>
<BODY>
```

</BODY>
</HTML>

EVENT HANDLING:

JavaScript is an Event Driven System

Event:

An Event is " any change that the user makes to the state of the browser"

There are 2 types of events that can be used to trigger script:

1. Window Events
2. User Events

1. Window Events, which occurs when
   - A page loads or unloads
   - Focus is being moved to or away from a window or frame
   - After a period of time has elapsed
2. User Events, which occur when the user interacts with elements in the page using mouse or a keyboard.

Event Handlers:

Event handlers are Javascript functions which you associate with an HTML element as part of its definition in the HTML source code.

Syntax:

```
<element attributes
eventAttribute=" handler" >
```

Attribute	**Description**
onblur	Triggered when an element **loses focus**.
onchange	Triggered when the **value of a form field** is changed.
onclick	Triggered when the user **clicks** on an element.
ondblclick	Triggered when the user **double-clicks** on an element.
onfocus	Triggered when an element **gains focus**.
onkeydown	Triggered when a **key is pressed down**.
onkeypress	Triggered when a **key is pressed and released**.
onkeyup	Triggered when a **key is released**.
onload	Triggered when the **page finishes loading**.
onmousedown	Triggered when the **mouse button is pressed down** over an element.
onmousemove	Triggered when the **mouse is moved** over an element.
onmouseout	Triggered when the **mouse leaves** an element.
onmouseover	Triggered when the **mouse hovers** over an element.
onmouseup	Triggered when the **mouse button is released**.
onmove	Triggered when a **window is moved** or restored. *(Mostly obsolete)*

Attribute	Description
onresize	Triggered when the **window is resized**.
onmousewheel	Triggered when the **mouse wheel is rotated**. *(Use with caution)*
onreset	Triggered when a **form is reset**.
onselect	Triggered when **text is selected** in a form field.
onsubmit	Triggered when a **form is submitted**.
onunload	Triggered when the user **leaves the page**.

Examples:

```
1. <html>
<head>
<script language="javascript">
function fun()
{
alert("Page is Loaded");
}
</script>
</head>
<body onload="fun()">
</body>
</html>

2. <html>
<head>
<script language="javascript">
function fun()
{
alert("You Clicked on Button");
}
</script>
</head>
<body>
<input type="button" value="Click Me" onClick="fun()">
</body>
</html>

3. <HTML>
<HEAD>
<script>
function check()
```

```
{
var number=form.number.value;
var i=1;
while(i<=number)
{
if(i%2==0)
document.write("<center>"+i+"</center><br>");
i++;
}
}
</script>
</HEAD>
<BODY>
<form name='form' onSubmit='check();'>
<p>Find 1 to n even numbers</p><br>
<input type='text' name='number'><br><br>
<input type='submit' value='Get Even Numbers'>
</form>
</BODY>
</HTML>
```

Math Object:

The math object provides you properties and methods for mathematical constants and functions. Unlike other global objects, Math is not a constructor. All the properties and methods of Math are static and can be called by using Math as an object without creating it. Thus, you refer to the constant pi as Math.PI and you call the sine function as Math.sin(x), where x is the method's argument.

Math Properties (Constants)

JavaScript provides 8 mathematical constants that can be accessed with the Math object:

Example

- Math.E // returns Euler's number
- Math.PI // returns PI
- Math.SQRT2 // returns the square root of 2
- Math.SQRT1_2 // returns the square root of 1/2
- Math.LN2 // returns the natural logarithm of 2
- Math.LN10 // returns the natural logarithm of 10
- Math.LOG2E // returns base 2 logarithm of E
- Math.LOG10E // returns base 10 logarithm of E

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x.
acos(x)	Returns the arccosine of x, in radians.
acosh(x)	Returns the hyperbolic arccosine of x.
asin(x)	Returns the arcsine of x, in radians.
asinh(x)	Returns the hyperbolic arcsine of x.
atan(x)	Returns the arctangent of x (between -π/2 and π/2 radians).
atan2(y, x)	Returns the arctangent of the quotient of its arguments y/x.
atanh(x)	Returns the hyperbolic arctangent of x.
cbrt(x)	Returns the cube root of x.
ceil(x)	Returns x, rounded **upwards** to the nearest integer.
cos(x)	Returns the cosine of x (in radians).
cosh(x)	Returns the hyperbolic cosine of x.
exp(x)	Returns the value of e^x.
floor(x)	Returns x, rounded **downwards** to the nearest integer.
log(x)	Returns the natural logarithm (base **e**) of x.
max(x, y, ...)	Returns the number with the **highest** value.
min(x, y, ...)	Returns the number with the **lowest** value.
pow(x, y)	Returns the value of x raised to the power of y (**x^y**).
random()	Returns a random number between **0** and **1**.
round(x)	Rounds x to the **nearest** integer.
sin(x)	Returns the sine of x (in radians).
sinh(x)	Returns the hyperbolic sine of x.
sqrt(x)	Returns the square root of x.
tan(x)	Returns the tangent of x (in radians).
tanh(x)	Returns the hyperbolic tangent of x.
trunc(x)	Returns the integer part of x, removing any fractional digits.

Example:

```
<HTML>
<HEAD>
<script>
document.write("<center>"+Math.PI+"</center><br>");
document.write("<center>"+Math.ceil(0.991)+"</center><br>");
document.write("<center>"+Math.floor(0.991)+"</center><br>");
document.write("<center>"+Math.min(12,3,42,55,75,1)+"</center><br>");
```

```
document.write("<center>"+Math.max(12,3,42,55,75,1)+"</center><br>");
document.write("<center>"+Math.pow(5,3)+"</center><br>");
document.write("<center>"+Math.sqrt(25)+"</center><br>");
document.write("<center>"+Math.random()+"</center><br>");
</script>
</HEAD>
<BODY>
</BODY>
</HTML>
```

DHTML WITH JAVASCRIPT:

- It refers to the technique of making web pages dynamic by client-side scripting to manipulate the document content and presentation
- Web pages can be made more lively, dynamic or interactive by DHTML techniques.
- DHTML is not a markup language or a software tool.
- DHTML involves the following aspects.
  - HTML - For designing static web pages
  - JAVASCRIPT - For browser scripting
  - CSS (Cascading Style Sheets) - For style and presentation control
  - DOM(Document Object Model) - An API for scripts to access and manipulate the web page as a document.

So**, DHTML = HTML + CSS + JAVASCRIPT + DOM**

HTML Vs DHTML

HTML	**DHTML**
1. It is used to create static web pages.	Used to create dynamic web pages.
2. Consists of simple HTML tags.	Made up of HTML tags+CSS+javascript+DOM
3. It is a markup language.	It is a technique to make web pages dynamic through client-side programming.
4. Do not allow to alter the text and graphics on the web page unless web page gets changed.	DHTML allows you to alter the text and graphics of the web page without changing the entire web page.
5. Creation of HTML web pages is simple	Creation of DHTML web pages is complex.
6. Web pages are less interactive.	Web pages are more interactive.
7. HTML sites will be slow upon client-side technologies.	DHTML sites will be fast enough upon client-side technologies.

Form Validation:

Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server.

JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally 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 data.
- **Data Format Validation** − Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

Example:

```
<html>
<head>
<script language="javascript" type="text/javascript">
function validate()
{
if(form.name.value==0)
{
alert("Username should not be empty");
form.name.focus();
return false;
}
if(form.password.value==0)
{
alert("password should not be empty");
form.password.focus();
return false;
}
if(form.password.value.length<6)
{
alert("password length should be greater than 6");
form.password.focus();
return false;
}
return true;
}
</script>
</head>
```

```
<body>
<center>
<form name="form" onsubmit="return validate(this);"
action="login.jsp">
<h1>Login Here</h1>
<table>
<tr><td>Enter Name</td><td> <input type="text"
name="name"></td></tr>
<tr><td>Enter Password</td><td> <input
type="password" name="password"></td></tr>
<tr><td colspan='2' align='center'><input type="submit"
value="Login"></td></tr>
</table>
</form>
</center>
</body>
</html>
```

Registration page validation Example:

```
<html>
<head>
<script language="javascript" type="text/javascript">
function validate()
{
if(form.name.value==0)
{
alert("Username should not be empty");
form.name.focus();
return false;
}
if(form.name.value.length<8)
{
alert("Username should be minimum 8
characters");
form.name.focus();
return false;
}
if(form.password.value==0)
{
alert("password should not be empty");
form.password.focus();
return false;
```

```
}
if(form.password.value.length<6)
{
alert("password length should be greater than 6");
form.password.focus();
return false;
}
if(form.gender.value==0)
{
alert("please select gender");
form.name.focus();
return false;
}
if(form.address.value==0)
{
alert("Address should not be empty");
form.name.focus();
return false;
}
if(form.mobile.value==0)
{
alert("Mobile num should not be empty");
form.name.focus();
return false;
}
if(form.mobile.value.length<10)
{
alert("Mobile num should be 10 digits");
form.name.focus();
return false;
}
return true;
}
</script>
</head>
<body>
<center>
<form name="form" onsubmit="return validate(this);"
action="login.jsp">
<h1>Register Here</h1>
<table>
<tr><td>Enter Name</td><td> <input type="text"
```

```
name="name"></td></tr>
<tr><td>Enter Password</td><td> <input
type="password" name="password"></td></tr>
<tr><td>Select Gender</td><td><input type="radio"
name="gender" value="male">Male<input type="radio"
name="gender" value="female">FeMale</td></tr>
<tr><td>Address</td><td><textarea
name="address"></textarea></td></tr>
<tr><td>Select State</td>
<td>
<select name="country">
<option value="Srilanka">Srilanka
<option value="India">India
<option value="Australia">Australia
</select>
</td>
</tr>
<tr><td>Enter Mobile</td><td> <input type="text"
name="mobile"></td></tr>
<tr><td colspan='2' align='center'><input type="submit"
value="Login"></td></tr>
</table>
</form>
</center>
</body>
</html>
```

Thank You

And I'll keep on updating this file as per the requirements of the organizations