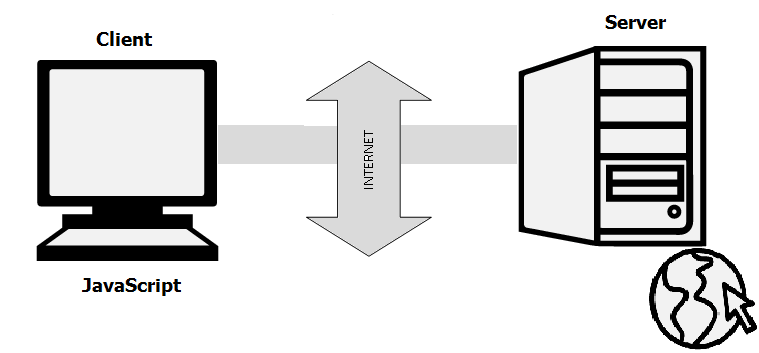
<https://www.guru99.com/interactive-javascript-tutorials.html>

Day 1: **What is 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](https://www.guru99.com/mobile-testing.html)application development.



**A Simple JavaScript Program**

### Hello World Example:

<html>

<head>

<title>My First JavaScript code!!!</title>

<script> alert("Hello World!"); </script>

</head>

<body>

</body>

</html>

### Summary

* JavaScript is a **client-side scripting language** developed by Brendan Eich.
* JavaScript can be **run on any operating systems** and almost all web browsers.
* You need a text editor to write JavaScript code and a browser to display your web page.

Day 1.1: **JavaScript Variable: Declare, Assign a Value with Example**

Variables are used to **store values** (name = "John") **or expressions** (sum = x + y).

### Declare Variables in JavaScript

var name;

### Assign a Value to the Variable

var name = "John";

OR

var name;

name = "John";

### Naming Variables

<html>

<head>

<title>Variables!!!</title>

<script type="text/javascript">

var one = 22;

var two = 3;

var add = one + two;

var minus = one - two;

var multiply = one \* two;

var divide = one/two;

document.write("First No: = " + one + "<br />Second No: = " + two + " <br />");

document.write(one + " + " + two + " = " + add + "<br/>");

document.write(one + " - " + two + " = " + minus + "<br/>");

document.write(one + " \* " + two + " = " + multiply + "<br/>");

document.write(one + " / " + two + " = " + divide + "<br/>");

</script>

</head>

<body>

</body>

</html>

Day 1.2: **JavaScript Array Methods: Create with Example**

### JavaScript Create Array

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

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";

### JavaScript Array Methods

The Array object has many properties and methods which help developers to handle arrays easily and efficiently. You can get the value of a property by specifying arrayname.property and the output of a method by specifying arrayname.method().

1. **length property** --> If you want to know the number of elements in an array, you can use the length property.
2. **prototype property** --> If you want to add new properties and methods, you can use the prototype property.
3. **reverse method** --> You can reverse the order of items in an array using a reverse method.
4. **sort method -->** You can sort the items in an array using sort method.
5. **pop method** --> You can remove the last item of an array using a pop method.
6. **shift method** --> You can remove the first item of an array using shift method.
7. **push method** --> You can add a value as the last item of the array.

<head>

<title>Arrays!!!</title>

<script type="text/javascript">

var students = new Array("John", "Ann", "Aaron", "Edwin", "Elizabeth");

Array.prototype.displayItems=function(){

for (i=0;i<this.length;i++){

document.write(this[i] + "<br />");

} }

document.write("students array<br />"); students.displayItems();

document.write("<br />The number of items in students array is " + students.length + "<br />");

document.write("<br />The SORTED students array<br />");

students.sort(); students.displayItems();

document.write("<br />The REVERSED students array<br />");

students.reverse(); students.displayItems();

document.write("<br />THE students array after REMOVING the LAST item<br />");

students.pop(); students.displayItems();

document.write("<br />THE students array after PUSH<br />");

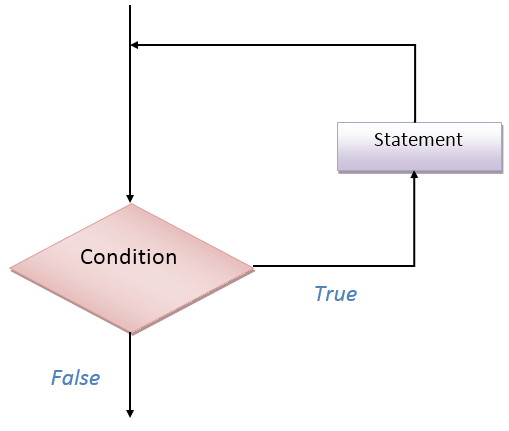
students.push("New Stuff"); students.displayItems();

</script>

</head>

Day 1.3:

# **For, While and Do While LOOP in JavaScript (with Example)**



### Different Types of Loops

There are mainly four types of loops in JavaScript.

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

## for loop 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.

<head>

<script type="text/javascript">

var students = new Array("John", "Ann", "Aaron", "Edwin", "Elizabeth");

document.write("<b>Using for loops </b><br />");

for (i=0;i<students.length;i++)

{

document.write(students[i] + "<br />");

}

</script>

</head>

## while loop

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.

<head>

<script type="text/javascript">

document.write("<b>Using while loops </b><br />");

var i = 0, j = 1, k;

document.write("Fibonacci series less than 40<br />");

while(i<40)

{

document.write(i + "<br />");

k = i+j;

i = j;

j = k;

}

</script>

</head>

## do…while loop

do

{

block of code to be executed

} while (condition)

<html>

<head>

<script type="text/javascript">

document.write("<b>Using do...while loops </b><br />");

var i = 2;

document.write("Even numbers less than 20<br />");

do

{

document.write(i + "<br />");

i = i + 2;

}while(i<20)

</script>

</head>

<body>

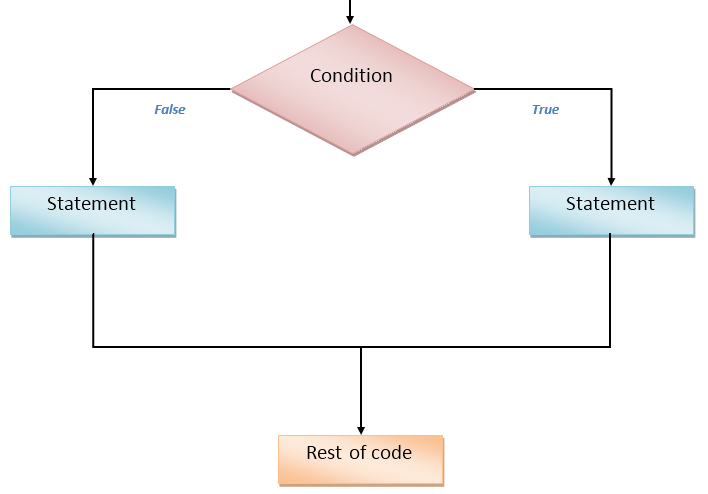
</body>

</html>

Day 1.4:

# **JavaScript Conditional Statements: IF, Else, Else IF (Example)**

Conditional statements are used to decide the flow of execution based on different conditions. If a condition is true, you can perform one action and if the condition is false, you can perform



### Different Types of Conditional Statements

There are mainly three types of conditional statements in JavaScript.

1. If statement
2. If…Else statement
3. If…Else If…Else statement

## **If statement**

Syntax:

if (condition)

{ lines of code to be executed if condition is true }

You can use If statement if you want to check only a specific condition.

<head>

<title>IF Statments!!!</title>

<script type="text/javascript">

var age = prompt("Please enter your age");

if(age>=18)

document.write("You are an adult <br />");

if(age<18)

document.write("You are NOT an adult <br />");

</script>

</head>

## **If…Else statement**

Syntax:

if (condition)

{ lines of code to be executed if the condition is true }

else

{ lines of code to be executed if the condition is false }

You can use If….Else statement if you have to check two conditions and execute a different set of codes.

<head>

<title>If...Else Statments!!!</title>

<script type="text/javascript">

// Get the current hours

var hours = new Date().getHours();

if(hours<12)

document.write("Good Morning!!!<br />");

else

document.write("Good Afternoon!!!<br />");

</script>

</head>

## **If…Else If…Else statement**

Syntax:

if (condition1)

{ lines of code to be executed if condition1 is true }

else if(condition2)

{ lines of code to be executed if condition2 is true}

else

{ lines of code to be executed if condition1 is false and condition2 is false }

You can use If….Else If….Else statement if you want to check more than two conditions.

<head>

<script type="text/javascript">

var one = prompt("Enter the first number");

var two = prompt("Enter the second number");

one = parseInt(one);

two = parseInt(two);

if (one == two)

document.write(one + " is equal to " + two + ".");

else if (one<two)

document.write(one + " is less than " + two + ".");

else

document.write(one + " is greater than " + two + ".");

</script>

</head>

Day.1.5:

# **JavaScript Define & Call Functions with Example**

Syntax:

function functionname() ->parenthesis

{ ->curly braces

lines of code to be executed

}

Try this yourself:

<head>

<title>Functions!!!</title>

<script type="text/javascript">

function myFunction()

{

document.write("This is a simple function.<br />");

}

myFunction();

</script>

</head>

### Function with Arguments

function functionname(arg1, arg2)

{ lines of code to be executed }

Try this yourself:

<head>

<script type="text/javascript">

var count = 0;

function countVowels(name)

{

for (var i=0;i<name.length;i++)

{

if(name[i] == "a" || name[i] == "e" || name[i] == "i" || name[i] == "o" || name[i] == "u")

count = count + 1;

}

document.write("Hello " + name + "!!! Your name has " + count + " vowels.");

}

var myName = prompt("Please enter your name");

countVowels(myName);

</script>

</head>

### JavaScript Return Value

function functionname(arg1, arg2)

{ lines of code to be executed

return val1;

}

Try this yourself:

<head>

<script type="text/javascript">

function returnSum(first, second)

{

var sum = first + second;

return sum;

}

var firstNo = 78;

var secondNo = 22;

document.write(firstNo + " + " + secondNo + " = " + returnSum(firstNo,secondNo));

</script>

</head>

Day 2: **Cookies in JavaScript: Set, Get & Delete Example**

### What are Cookies?

A cookie is a piece of data that is stored on your computer to be accessed by your browser. You also might have enjoyed the benefits of cookies knowingly or unknowingly. Have you ever saved your Facebook password so that you do not have to type it each and every time you try to login? If yes, then you are using cookies. Cookies are saved as key/value pairs.

### Why do you need a Cookie?

The communication between a web browser and server happens using a stateless protocol named HTTP. Stateless protocol treats each request independent. So, the server does not keep the data after sending it to the browser. But in many situations, the data will be required again. Here come cookies into a picture. With cookies, the web browser will not have to communicate with the server each time the data is required. Instead, it can be fetched directly from the computer.

## **Javascript Set Cookie**

You can create cookies using document. cookie property like this.

document.cookie = "cookiename=cookievalue"

You can even add expiry date to your cookie so that the particular cookie will be removed from the computer on the specified date. The expiry date should be set in the UTC/GMT format. If you do not set the expiry date, the cookie will be removed when the user closes the browser.

document.cookie = "cookiename=cookievalue; expires= Thu, 21 Aug 2014 20:00:00 UTC"

You can also set the domain and path to specify to which domain and to which directories in the specific domain the cookie belongs to. By default, a cookie belongs to the page that sets the cookie.

document.cookie = "cookiename=cookievalue; expires= Thu, 21 Aug 2014 20:00:00 UTC; path=/ "

//create a cookie with a domain to the current page and path to the entire domain.

## **JavaScript get Cookie**

You can access the cookie like this which will return all the cookies saved for the current domain.

var x = document.cookie

## **JavaScript Delete Cookie**

To delete a cookie, you just need to set the value of the cookie to empty and set the value of expires to a passed date.

document.cookie = "cookiename= ; expires = Thu, 01 Jan 1970 00:00:00 GMT"

### Try this Example yourself: (RUN TWICE)

<html>

<head>

<title>Cookie!!!</title>

<script type="text/javascript">

function createCookie(cookieName,cookieValue,daysToExpire)

{

var date = new Date();

date.setTime(date.getTime()+(daysToExpire\*24\*60\*60\*1000));

document.cookie = cookieName + "=" + cookieValue + "; expires=" + date.toGMTString();

}

function accessCookie(cookieName)

{

var name = cookieName + "=";

var allCookieArray = document.cookie.split(';');

for(var i=0; i<allCookieArray.length; i++)

{

var temp = allCookieArray[i].trim();

if (temp.indexOf(name)==0)

return temp.substring(name.length,temp.length);

}

return "";

}

function checkCookie()

{

var user = accessCookie("testCookie");

if (user!="")

alert("Welcome Back " + user + "!!!");

else

{

user = prompt("Please enter your name");

num = prompt("How many days you want to store your name on your computer?");

if (user!="" && user!=null)

{

createCookie("testCookie", user, num);

}

}

}

</script>

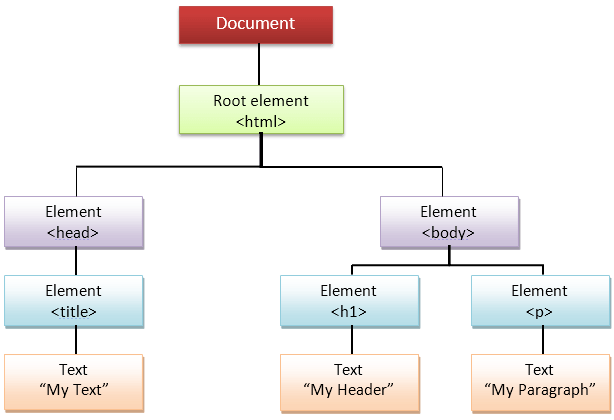
</head>

<body onload="checkCookie()"></body>

</html>

Day 2.1:

JavaScript DOM Tutorial with Example



### How to use DOM and Events

Using DOM, JavaScript can perform multiple tasks. It can create new elements and attributes, change the existing elements and attributes and even remove existing elements and attributes. JavaScript can also react to existing events and create new events in the page.

### getElementById, innerHTML Example

1. getElementById:  To access elements and attributes whose id is set.
2. innerHTML: To access the content of an element.

Try this Example yourself:

<html>

<head>

<title>DOM!!!</title>

</head>

<body>

<h1 id="one">Welcome</h1>

<p>This is the welcome message.</p>

<h2>Technology</h2>

<p>This is the technology section.</p>

<script type="text/javascript">

var text = document.getElementById("one").innerHTML;

alert("The first heading is " + text);

</script>

</body>

</html>

### getElementsByTagName Example

getElementsByTagName: To access elements and attributes using tag name. This method will return an array of all the items with the same tag name.

Try this Example yourself:

<html>

<head>

<title>DOM!!!</title>

</head>

<body>

<h1>Welcome</h1>

<p>This is the welcome message.</p>

<h2>Technology</h2>

<p id="second">This is the technology section.</p>

<script type="text/javascript">

var paragraphs = document.getElementsByTagName("p");

alert("Content in the second paragraph is " + paragraphs[1].innerHTML);

document.getElementById("second").innerHTML = "The orginal message is changed.";

</script>

</body>

</html>

### Event handler Example

1. createElement:  To create new element
2. removeChild: Remove an element
3. You can add an **event handler** to a particular element like this:

document.getElementById(id).onclick=function()

{ lines of code to be executed }

OR

document.getElementById(id).addEventListener("click", functionname)

Try this Example yourself:

<html>

<head>

<title>DOM!!!</title>

</head>

<body>

<input type="button" id="btnClick" value="Click Me!!" />

<script type="text/javascript">

document.getElementById("btnClick").addEventListener("click", clicked);

function clicked()

{ alert("You clicked me!!!"); }

</script>

</body>

</html>

Day 2.2: **Object Oriented JavaScript (OOJS) Tutorial with Example**

Many times, variables or arrays are not sufficient to simulate real-life situations. JavaScript allows you to create objects that act like real life objects. A student or a home can be an object that have many unique characteristics of their own. You can create properties and methods to your objects to make programming easier. If your object is a student, it will have properties like first name, last name, id etc and methods like calculateRank, changeAddress etc. If your object is a home, it will have properties like a number of rooms, paint color, location etc and methods like calculateArea, changeOwner etc.

### How to Create an Object

You can create an object like this:

var objName = new Object();

objName.property1 = value1;

objName.property2 = value2;

objName.method1 = function(){ line of code }

OR

var objName= {property1:value1, property2:value2, method1: function(){ lines of code} };

**Access Object Properties and Methods**

You can access properties of an object like this:

objectname.propertyname;

You can access methods of an object like this:

objectname.methodname();

Try this Example yourself:

<head>

<title>Objects!!!</title>

<script type="text/javascript">

var student = new Object();

student.fName = "John";

student.lName = "Smith";

student.id = 5;

student.markE = 76;

student.markM = 99;

student.markS = 87;

student.calculateAverage = function()

{

return (student.markE + student.markM + student.markS)/3;

};

student.displayDetails = function()

{

document.write("Student Id: " + student.id + "<br />");

document.write("Name: " + student.fName + " " + student.lName + "<br />");

var avg = student.calculateAverage();

document.write("Average Marks: " + avg);

};

student.displayDetails();

</script>

</head>

### OOPS Constructor

But creating objects of this kind is not that useful because here also, you will have to create different objects for different students. Here comes object constructor into picture. Object constructor helps you create an object type which can be reused to meet the need of individual instance.

Try this Example yourself:

<head>

<script type="text/javascript">

function Student(first, last, id, english, maths, science)

{

this.fName = first;

this.lName = last;

this.id = id;

this.markE = english;

this.markM = maths;

this.markS = science;

this.calculateAverage = function()

{

return (this.markE + this.markM + this.markS)/3;

}

this.displayDetails = function()

{

document.write("Student Id: " + this.id + "<br />");

document.write("Name: " + this.fName + " " + this.lName + "<br />");

var avg = this.calculateAverage();

document.write("Average Marks: " + avg + "<br /><br />");

}

}

var st1 = new Student("John", "Smith", 15, 85, 79, 90);

var st2 = new Student("Hannah", "Turner", 23, 75, 80, 82);

var st3 = new Student("Kevin", "White", 4, 93, 89, 90);

var st4 = new Student("Rose", "Taylor", 11, 55, 63, 45);

st1.displayDetails();

st2.displayDetails();

st3.displayDetails();

st4.displayDetails();

</script>

</head>

## **Loop Through the Properties of an Object**

for (variablename in objectname)

{ lines of code to be executed }

The for/in a loop is usually used to loop through the properties of an object. You can give any name for the variable, but the name of the object should be the same as that of an already existing object which you need to loop through.

Try this Example yourself:

<html>

<head>

<script type="text/javascript">

var employee={first:"John", last:"Doe", department:"Accounts"};

var details = "";

document.write("<b>Using for/in loops </b><br />");

for (var x in employee)

{

details = x + ": " + employee[x];

document.write(details + "<br />");

}

</script>

</head>

<body>

</body>

</html>

Day: 2.3

**Internal & External JavaScript: Learn with Example**

You can use JavaScript code in two ways.

1. You can either include the JavaScript code **internally within your HTML document**itself
2. You can keep the JavaScript code in **a separate external file** and then point to that file from your HTML document.

## **What is Internal JavaScript?**

<html>

<head>

<title>My First JavaScript code!!!</title>

<script type="text/javascript">

// Create a Date Object

var day = new Date();

// Use getDay function to obtain todays Day.

// getDay() method returns the day of the week as a number like 0 for Sunday, 1 for Monday,….., 5

// This value is stored in today variable

var today = day.getDay();

// To get the name of the day as Sunday, Monday or Saturday, we have created an array named weekday and stored the values

var weekday = new Array(7);

weekday[0]="Sunday";

weekday[1]="Monday";

\*\*\*\*\*\*\*\*\*\*

weekday[5]="Friday";

weekday[6]="Saturday";

// weekday[today] will return the day of the week as we want

document.write("Today is " + weekday[today] + ".");

</script>

</head>

<body>

</body>

</html>

## **What is External JavaScript?**

You plan to display the current date and time in all your web pages. Suppose you wrote the code and copied into all your web pages (say 100). But later, you want to change the format in which the date or time is displayed. In this case, you will have to make changes in all the 100 web pages. This will be a very time consuming and difficult task.

So, save the JavaScript code in a new file with the extension .js. Then, add a line of code in all your web pages to point to your .js file like this:

<script type="text/javascript" src="currentdetails.js">

**Note**: It is assumed that the .js file and all your web pages are in the same folder. If the external.js file is in a different folder, you need to specify the full path to your file in the src attribute.

How to link external JavaScript

var currentDate = new Date();

var day = currentDate.getDate();

var month = currentDate.getMonth() + 1;

var monthName;

var hours = currentDate.getHours();

var mins = currentDate.getMinutes();

var secs = currentDate.getSeconds();

var strToAppend;

if (hours >12 ){

hours1 = "0" + (hours - 12);

strToAppend = "PM";

}

else if (hours <12){

hours1 = "0" + hours;

strToAppend = "AM";

}

else{

hours1 = hours;

strToAppend = "PM";

}

if(mins<10)

mins = "0" + mins;

if (secs<10)

secs = "0" + secs;

switch (month) {

case 1:

monthName = "January"; break;

case 2:

monthName = "February"; break;

\*\*\*\*\*

case 11:

monthName = "November"; break;

case 12:

monthName = "December"; break;

}

var year = currentDate.getFullYear();

var myString;

myString = "Today is " + day + " - " + monthName + " - " + year + ".<br />Current time is " + hours1 + ":" + mins + ":" + secs + " " + strToAppend + ".";

document.write(myString);

This is your currentdetails.js file. Don’t worry seeing long lines of code. You will learn to code soon. Make changes to your HTML document like this:

<head>

<title>My External JavaScript Code!!!</title>

<script type="text/javascript" src="currentdetails.js">

</script>

</head>

### When to Use Internal and External JavaScript Code?

If you have only a few lines of code that is specific to a particular webpage, then it is better to keep your JavaScript code internally within your HTML document.

On the other hand, if your JavaScript code is used in many web pages, then you should consider keeping your code in a separate file. In that case, if you wish to make some changes to your code, you just have to change only one file which makes code maintenance easy. If your code is too long, then also it is better to keep it in a separate file. This helps in easy debugging.

Day:3

**Practical Code Examples using JavaScript**

## **Example#1: JavaScript Multiplication Table**

Create a simple multiplication table asking the user the number of rows and columns he wants.

<html>

<head>

<title>Multiplication Table</title>

<script type="text/javascript">

var rows = prompt("How many rows for your multiplication table?");

var cols = prompt("How many columns for your multiplication table?");

if(rows == "" || rows == null)

rows = 10;

if(cols== "" || cols== null)

cols = 10;

createTable(rows, cols);

function createTable(rows, cols)

{

var j=1;

var output = "<table border='1' width='500' cellspacing='0'cellpadding='5'>";

for(i=1;i<=rows;i++)

{

output = output + "<tr>";

while(j<=cols)

{

output = output + "<td>" + i\*j + "</td>";

j = j+1;

}

output = output + "</tr>";

j = 1;

}

output = output + "</table>";

document.write(output);

}

</script>

</head>

<body>

</body>

</html>

## **Example#2: JS Forms Example:**

Create a sample form program that collects the first name, last name, email, user id, password and confirms password from the user. All the inputs are mandatory and email address entered should be in correct format. Also, the values entered in the password and confirm password textboxes should be the same. After validating using JavaScript, In output display proper error messages in red color just next to the textbox where there is an error.

Solution with Source Code:

<html>

<head>

<title>Form Validation</title>

<script type="text/javascript">

var divs = new Array();

divs[0] = "errFirst";

divs[1] = "errLast";

divs[2] = "errEmail";

divs[3] = "errUid";

divs[4] = "errPassword";

divs[5] = "errConfirm";

function validate()

{

var inputs = new Array();

inputs[0] = document.getElementById('first').value;

inputs[1] = document.getElementById('last').value;

inputs[2] = document.getElementById('email').value;

inputs[3] = document.getElementById('uid').value;

inputs[4] = document.getElementById('password').value;

inputs[5] = document.getElementById('confirm').value;

var errors = new Array();

errors[0] = "<span style='color:red'>Please enter your first name!</span>";

errors[1] = "<span style='color:red'>Please enter your last name!</span>";

errors[2] = "<span style='color:red'>Please enter your email!</span>";

errors[3] = "<span style='color:red'>Please enter your user id!</span>";

errors[4] = "<span style='color:red'>Please enter your password!</span>";

errors[5] = "<span style='color:red'>Please confirm your password!</span>";

for (i in inputs)

{

var errMessage = errors[i];

var div = divs[i];

if (inputs[i] == "")

document.getElementById(div).innerHTML = errMessage;

else if (i==2)

{

var atpos=inputs[i].indexOf("@");

var dotpos=inputs[i].lastIndexOf(".");

if (atpos<1 || dotpos<atpos+2 || dotpos+2>=inputs[i].length)

document.getElementById('errEmail').innerHTML = "<span style='color: red'>Enter a valid email address!</span>";

else

document.getElementById(div).innerHTML = "OK!";

}

else if (i==5)

{

var first = document.getElementById('password').value;

var second = document.getElementById('confirm').value;

if (second != first)

document.getElementById('errConfirm').innerHTML = "<span style='color: red'>Your passwords don't match!</span>";

else

document.getElementById(div).innerHTML = "OK!";

}

else

document.getElementById(div).innerHTML = "OK!";

}

}

function finalValidate()

{

var count = 0;

for(i=0;i<6;i++)

{

var div = divs[i];

if(document.getElementById(div).innerHTML == "OK!")

count = count + 1;

}

if(count == 6)

document.getElementById("errFinal").innerHTML = "All the data you entered is correct!!!";

}

</script>

</head>

<body>

<table id="table1">

<tr>

<td>First Name:</td>

<td><input type="text" id="first" onkeyup="validate();" /></td>

<td><div id="errFirst"></div></td>

</tr>

<tr>

<td>Last Name:</td>

<td><input type="text" id="last" onkeyup="validate();"/></td>

<td><div id="errLast"></div></td>

</tr>

<tr>

<td>Email:</td>

<td><input type="text" id="email" onkeyup="validate();"/></td>

<td><div id="errEmail"></div></td>

</tr>

<tr>

<td>User Id:</td>

<td><input type="text" id="uid" onkeyup="validate();"/></td>

<td><div id="errUid"></div></td>

</tr>

<tr>

<td>Password:</td>

<td><input type="password" id="password" onkeyup="validate();"/></td>

<td><div id="errPassword"></div></td>

</tr>

<tr>

<td>Confirm Password:</td>

<td><input type="password" id="confirm" onkeyup="validate();"/></td>

<td><div id="errConfirm"></div></td>

</tr>

<tr>

<td><input type="button" id="create" value="Create" onclick="validate();finalValidate();"/></td>

<td><div id="errFinal"></div></td>

</tr>

</table>

</body>

</html>

## **Example#3: POPUP Message using Event:**

Display a simple message "Welcome!!!" on your demo webpage and when the user hovers over the message, a popup should be displayed with a message "Welcome to my WebPage!!!".

Solution:

<html>

<head>

<title>Event!!!</title>

<script type="text/javascript">

function trigger() {

document.getElementById("hover").addEventListener("mouseover", popup);

function popup()

{

alert("Welcome to my WebPage!!!");

}

}

</script>

<style>

p{

     font-size:50px;

     position: fixed;

     left: 550px;

     top: 300px;

}

</style>

</head>

<body  onload="trigger();">

<p id="hover">Welcome!!!</p>

</body>

</html>

Day 3.1:

JavaScript Interview Questions & Answers

**1. What is JavaScript?**

JavaScript is a client-side as well as server side scripting language that can be inserted into HTML pages and is understood by web browsers. JavaScript is also an Object based Programming language

**2. Enumerate the differences between Java and JavaScript?**

Java is a complete programming language. In contrast, JavaScript is a coded program that can be introduced to HTML pages. These two languages are not at all inter-dependent and are designed for the different intent. Java is an object - oriented programming (OOPS) or structured programming language like C++ or C whereas JavaScript is a client-side scripting language.

**3. What are JavaScript Data Types?**

Following are the JavaScript Data types:

* Number
* String
* Boolean
* Object
* Undefined

**4. What is the use of isNaN function?**

isNan function returns true if the argument is not a number otherwise it is false.

**5. Between JavaScript and an ASP script, which is faster?**

JavaScript is faster. JavaScript is a client-side language and thus it does not need the assistance of the web server to execute. On the other hand, ASP is a server-side language and hence is always slower than JavaScript. Javascript now is also a server side language (nodejs).

**6. What is negative infinity?**

Negative Infinity is a number in JavaScript which can be derived by dividing negative number by zero.

**7. Is it possible to break JavaScript Code into several lines?**

Breaking within a string statement can be done by the use of a backslash, '\', at the end of the first line

Example:

document.write("This is \a program");

And if you change to a new line when not within a string statement, then javaScript ignores break in line.

Example:

var x=1, y=2,

z= x+y;

The above code is perfectly fine, though not advisable as it hampers debugging.

**8. Which company developed JavaScript?**

Netscape is the software company who developed JavaScript.

**9. What are undeclared and undefined variables?**

Undeclared variables are those that do not exist in a program and are not declared. If the program tries to read the value of an undeclared variable, then a runtime error is encountered.

Undefined variables are those that are declared in the program but have not been given any value. If the program tries to read the value of an undefined variable, an undefined value is returned.

**10. Write the code for adding new elements dynamically?**

<html>

<head>

<title>t1</title>

<script type="text/javascript">

function addNode() { var newP = document.createElement("p");

var textNode = document.createTextNode(" This is a new text node");

newP.appendChild(textNode); document.getElementById("firstP").appendChild(newP); }

</script> </head>

<body> <p id="firstP">firstP<p> </body>

</html>

**11. What are global variables? How are these variable declared and what are the problems associated with using them?**

Global variables are those that are available throughout the length of the code, that is, these have no scope. The var keyword is used to declare a local variable or object. If the var keyword is omitted, a global variable is declared.

Example:

// Declare a global globalVariable = "Test";

The problems that are faced by using global variables are the clash of variable names of local and global scope. Also, it is difficult to debug and test the code that relies on global variables.

**12. What is a prompt box?**

A prompt box is a box which allows the user to enter input by providing a text box. Label and box will be provided to enter the text or number.

**13. What is 'this' keyword in JavaScript?**

'This' keyword refers to the object from where it was called.

**14. Explain the working of timers in JavaScript? Also elucidate the drawbacks of using the timer, if any?**

Timers are used to execute a piece of code at a set time or also to repeat the code in a given interval of time. This is done by using the functions **setTimeout, setInterval**and**clearInterval**.

The **setTimeout(function, delay)** function is used to start a timer that calls a particular function after the mentioned delay. The **setInterval(function, delay)** function is used to repeatedly execute the given function in the mentioned delay and only halts when cancelled. The **clearInterval(id)** function instructs the timer to stop.

Timers are operated within a single thread, and thus events might queue up, waiting to be executed.

**15. Which symbol is used for comments in Javascript?**

// for Single line comments and

/\* Multi Line

Comment

\*/

**16. What is the difference between ViewState and SessionState?**

'ViewState' is specific to a page in a session.

'SessionState' is specific to user specific data that can be accessed across all pages in the web application.

**17. What is === operator?**

=== is called as strict equality operator which returns true when the two operands are having the same value without any type conversion.

**18. Explain how can you submit a form using JavaScript?**

To submit a form using JavaScript use document.form[0].submit();

document.form[0].submit();

**19. Does JavaScript support automatic type conversion?**

Yes JavaScript does support automatic type conversion, it is the common way of type conversion used by JavaScript developers

**20. How can the style/class of an element be changed?**

It can be done in the following way:

document.getElementById("myText").style.fontSize = "20?;

or

document.getElementById("myText").className = "anyclass";

**21. Explain how to read and write a file using JavaScript?**

There are two ways to read and write a file using JavaScript

* Using JavaScript extensions
* Using a web page and Active X objects

**22. What are all the looping structures in JavaScript?**

Following are looping structures in Javascript:

* For
* While
* do-while loops

**23. What is called Variable typing in Javascript?**

Variable typing is used to assign a number to a variable and the same variable can be assigned to a string.

Example

i = 10;

i = "string";

This is called variable typing.

**24. How can you convert the string of any base to integer in JavaScript?**

The parseInt() function is used to convert numbers between different bases. parseInt() takes the string to be converted as its first parameter, and the second parameter is the base of the given string.

In order to convert 4F (of base 16) to integer, the code used will be -

parseInt ("4F", 16);

**25. Explain the difference between "==" and "==="?**

"==" checks only for equality in value whereas "===" is a stricter equality test and returns false if either the value or the type of the two variables are different.

**26. What would be the result of 3+2+"7"?**

Since 3 and 2 are integers, they will be added numerically. And since 7 is a string, its concatenation will be done. So the result would be 57.

**27. Explain how to detect the operating system on the client machine?**

In order to detect the operating system on the client machine, the navigator.platform string (property) should be used.

**28. What do mean by NULL in Javascript?**

The NULL value is used to represent no value or no object. It implies no object or null string, no valid boolean value, no number and no array object.

**29. What is the function of delete operator?**

The delete keyword is used to delete the property as well as its value.

Example

var student= {age:20, batch:"ABC"};

delete student.age;

**30. What is an undefined value in JavaScript?**

Undefined value means the

* Variable used in the code doesn't exist
* Variable is not assigned to any value
* Property doesn't exist

**31. What are all the types of Pop up boxes available in JavaScript?**

* Alert
* Confirm and
* Prompt

**32. What is the use of Void(0)?**

Void(0) is used to prevent the page from refreshing and parameter "zero" is passed while calling.

Void(0) is used to call another method without refreshing the page.

**33. How can a page be forced to load another page in JavaScript?**

The following code has to be inserted to achieve the desired effect:

<script language="JavaScript" type="text/javascript" >

<!-- location.href="http://newhost/newpath/newfile.html"; //--></script>

**34. What is the data type of variables of in JavaScript?**

All variables in the JavaScript are object data types.

**35. What is the difference between an alert box and a confirmation box?**

An alert box displays only one button which is the OK button.

But a Confirmation box displays two buttons namely OK and cancel.

**36. What are escape characters?**

Escape characters (Backslash) is used when working with special characters like single quotes, double quotes, apostrophes and ampersands. Place backslash before the characters to make it display.

Example:

document.write "I m a "good" boy"

document.write "I m a \"good\" boy"

**37. What are JavaScript Cookies?**

Cookies are the small test files stored in a computer and it gets created when the user visits the websites to store information that they need. Example could be User Name details and shopping cart information from the previous visits.

**38. Explain what is pop()method in JavaScript?**

The pop() method is similar as the shift() method but the difference is that the Shift method works at the start of the array. Also the pop() method take the last element off of the given array and returns it. The array on which is called is then altered.

Example:

var cloths = ["Shirt", "Pant", "TShirt"];

cloths.pop();

//Now cloth becomes Shirt,Pant

**39. Whether JavaScript has concept level scope?**

No. JavaScript does not have concept level scope. The variable declared inside the function has scope inside the function.

**40. Mention what is the disadvantage of using innerHTML in JavaScript?**

If you use innerHTML in JavaScript the disadvantage is

* Content is replaced everywhere
* We cannot use like "appending to innerHTML"
* Even if you use +=like "innerHTML = innerHTML + 'html'" still the old content is replaced by html
* The entire innerHTML content is re-parsed and build into elements, therefore its much slower
* The innerHTML does not provide validation and therefore we can potentially insert valid and broken HTML in the document and break it

**41. What is break and continue statements?**

Break statement exits from the current loop.

Continue statement continues with next statement of the loop.

**42. What are the two basic groups of dataypes in JavaScript?**

They are as –

* Primitive
* Reference types.

Primitive types are number and Boolean data types. Reference types are more complex types like strings and dates.

**43. How generic objects can be created?**

Generic objects can be created as:

var I = new object();

**44. What is the use of type of operator?**

'Typeof' is an operator which is used to return a string description of the type of a variable.

**45. Which keywords are used to handle exceptions?**

Try… Catch---finally is used to handle exceptions in the JavaScript

Try{

Code

}

Catch(exp){

Code to throw an exception

}

Finally{

Code runs either it finishes successfully or after catch

}

**46. Which keyword is used to print the text in the screen?**

document.write("Welcome") is used to print the text – Welcome in the screen.

**47. What is the use of blur function?**

Blur function is used to remove the focus from the specified object.

**48. What is variable typing?**

Variable typing is used to assign a number to a variable and then assign string to the same variable. Example is as follows:

i= 8;

i="john";

**49. How to find operating system in the client machine using JavaScript?**

The '**Navigator.appversion'** is used to find the name of the operating system in the client machine.

**50. What are the different types of errors in JavaScript?**

There are three types of errors:

* **Load time errors**: Errors which come up when loading a web page like improper syntax errors are known as Load time errors and it generates the errors dynamically.
* **Run time errors**: Errors that come due to misuse of the command inside the HTML language.
* **Logical Errors**: These are the errors that occur due to the bad logic performed on a function which is having different operation.

**51. What is the use of Push method in JavaScript?**

The push method is used to add or append one or more elements to the end of an Array. Using this method, we can append multiple elements by passing multiple arguments

**52. What is unshift method in JavaScript?**

Unshift method is like push method which works at the beginning of the array. This method is used to prepend one or more elements to the beginning of the array.

**53. What is the difference between JavaScript and Jscript?**

Both are almost similar. JavaScript is developed by Netscape and Jscript was developed by Microsoft .

**54. How are object properties assigned?**

Properties are assigned to objects in the following way -

obj["class"] = 12;

or

obj.class = 12;

**55. What is the 'Strict' mode in JavaScript and how can it be enabled?**

Strict Mode adds certain compulsions to JavaScript. Under the strict mode, JavaScript shows errors for a piece of codes, which did not show an error before, but might be problematic and potentially unsafe. Strict mode also solves some mistakes that hamper the JavaScript engines to work efficiently.

Strict mode can be enabled by adding the string literal "use strict" above the file. This can be illustrated by the given example:

function myfunction() {

"use strict";

var v = "This is a strict mode function";

}

**56. What is the way to get the status of a CheckBox?**

The status can be acquired as follows -

alert(document.getElementById('checkbox1').checked);

If the CheckBox will be checked, this alert will return TRUE.

**57. How can the OS of the client machine be detected?**

The navigator.appVersion string can be used to detect the operating system on the client machine.

**58. Explain window.onload and onDocumentReady?**

The onload function is not run until all the information on the page is loaded. This leads to a substantial delay before any code is executed.

onDocumentReady loads the code just after the DOM is loaded. This allows early manipulation of the code.

**59. How will you explain closures in JavaScript? When are they used?**

Closure is a locally declared variable related to a function which stays in memory when the function has returned.

For example:

function greet(message) {

console.log(message);

}

function greeter(name, age) {

return name + " says howdy!! He is " + age + " years old";

}

// Generate the message

var message = greeter("James", 23);

// Pass it explicitly to greet

greet(message);

This function can be better represented by using closures

function greeter(name, age) {

var message = name + " says howdy!! He is " + age + " years old";

return function greet() {

console.log(message);

};

}

// Generate the closure

var JamesGreeter = greeter("James", 23);

// Use the closure

JamesGreeter();

**60. How can a value be appended to an array?**

A value can be appended to an array in the given manner -

arr[arr.length] = value;

**61. Explain the for-in loop?**

The for-in loop is used to loop through the properties of an object.

The syntax for the for-in loop is -

for (variable name in object){

statement or block to execute

}

In each repetition, one property from the object is associated to the variable name, and the loop is continued till all the properties of the object are depleted.

**62. Describe the properties of an anonymous function in JavaScript?**

A function that is declared without any named identifier is known as an anonymous function. In general, an anonymous function is inaccessible after its declaration.

Anonymous function declaration -

var anon = function() {

alert('I am anonymous');

};

anon();

**63. What is the difference between .call() and .apply()?**

The function .call() and .apply() are very similar in their usage except a little difference. .call() is used when the number of the function's arguments are known to the programmer, as they have to be mentioned as arguments in the call statement. On the other hand, .apply() is used when the number is not known. The function .apply() expects the argument to be an array.

The basic difference between .call() and .apply() is in the way arguments are passed to the function. Their usage can be illustrated by the given example.

var someObject = {

myProperty : 'Foo',

myMethod : function(prefix, postfix) {

alert(prefix + this.myProperty + postfix);

}

};

someObject.myMethod('<', '>'); // alerts '<Foo>'

var someOtherObject = {

myProperty : 'Bar'

};

someObject.myMethod.call(someOtherObject, '<', '>'); // alerts '<Bar>'

someObject.myMethod.apply(someOtherObject, ['<', '>']); // alerts '<Bar>'

**64. Define event bubbling?**

JavaScript allows DOM elements to be nested inside each other. In such a case, if the handler of the child is clicked, the handler of parent will also work as if it were clicked too.

**65. Is JavaScript case sensitive? Give an example?**

Yes, JavaScript is case sensitive. For example, a function parseInt is not same as the function Parseint.

**66. What boolean operators can be used in JavaScript?**

The 'And' Operator (&&), 'Or' Operator (||) and the 'Not' Operator (!) can be used in JavaScript.

\*Operators are without the parenthesis.

**67. How can a particular frame be targeted, from a hyperlink, in JavaScript?**

This can be done by including the name of the required frame in the hyperlink using the 'target' attribute.

<a href="/newpage.htm" target="newframe">>New Page</a>

**68. What is the role of break and continue statements?**

Break statement is used to come out of the current loop while the continue statement continues the current loop with a new recurrence.

**69. Write the point of difference between web-garden and a web-farm?**

Both web-garden and web-farm are web hosting systems. The only difference is that web-garden is a setup that includes many processors in a single server while web-farm is a larger setup that uses more than one server.

**70. How are object properties assigned?**

Assigning properties to objects is done in the same way as a value is assigned to a variable. For example, a form object's action value is assigned as 'submit' in the following manner - Document.form.action="submit"

**71. What is the method for reading and writing a file in JavaScript?**

This can be done by Using JavaScript extensions (runs from JavaScript Editor), example for opening of a file -

fh = fopen(getScriptPath(), 0);

**72. How are DOM utilized in JavaScript?**

DOM stands for Document Object Model and is responsible for how various objects in a document interact with each other. DOM is required for developing web pages, which includes objects like paragraph, links, etc. These objects can be operated to include actions like add or delete. DOM is also required to add extra capabilities to a web page. On top of that, the use of API gives an advantage over other existing models.

**73. How are event handlers utilized in JavaScript?**

Events are the actions that result from activities, such as clicking a link or filling a form, by the user. An event handler is required to manage proper execution of all these events. Event handlers are an extra attribute of the object. This attribute includes event's name and the action taken if the event takes place.

**74. Explain the role of deferred scripts in JavaScript?**

By default, the parsing of the HTML code, during page loading, is paused until the script has not stopped executing. It means, if the server is slow or the script is particularly heavy, then the webpage is displayed with a delay. While using Deferred, scripts delays execution of the script till the time HTML parser is running. This reduces the loading time of web pages and they get displayed faster.

**75. What are the various functional components in JavaScript?**

The different functional components in JavaScript are-

**First-class functions:** Functions in JavaScript are utilized as first class objects. This usually means that these functions can be passed as arguments to other functions, returned as values from other functions, assigned to variables or can also be stored in data structures.

**Nested functions:** The functions, which are defined inside other functions, are called Nested functions. They are called 'everytime' the main function is invoked.

**76. Write about the errors shown in JavaScript?**

JavaScript gives a message if it encounters an error. The recognized errors are -

* Load-time errors: The errors shown at the time of the page loading are counted under Load-time errors. These errors are encountered by the use of improper syntax, and thus are detected while the page is getting loaded.
* Run-time errors: This is the error that comes up while the program is running. It is caused by illegal operations, for example, division of a number by zero, or trying to access a non-existent area of the memory.
* Logic errors: It is caused by the use of syntactically correct code, which does not fulfill the required task. For example, an infinite loop.

**77. What are Screen objects?**

Screen objects are used to read the information from the client's screen. The properties of screen objects are -

* AvailHeight: Gives the height of client's screen
* AvailWidth: Gives the width of client's screen.
* ColorDepth: Gives the bit depth of images on the client's screen
* Height: Gives the total height of the client's screen, including the taskbar
* Width: Gives the total width of the client's screen, including the taskbar

**78. Explain the unshift() method ?**

This method is functional at the starting of the array, unlike the push(). It adds the desired number of elements to the top of an array. For example -

var name = [ "john" ];

name.unshift( "charlie" );

name.unshift( "joseph", "Jane" );

console.log(name);

The output is shown below:

[" joseph "," Jane ", " charlie ", " john "]

**79. Define unescape() and escape() functions?**

The escape () function is responsible for coding a string so as to make the transfer of the information from one computer to the other, across a network.

For Example:

<script>

document.write(escape("Hello? How are you!"));

</script>

Output: Hello%3F%20How%20are%20you%21

The unescape() function is very important as it decodes the coded string.

It works in the following way. For example:

<script>

document.write(unescape("Hello%3F%20How%20are%20you%21"));

</script>

Output: Hello? How are you!

**80. What are the decodeURI() and encodeURI()?**

EncodeURl() is used to convert URL into their hex coding. And DecodeURI() is used to convert the encoded URL back to normal.

<script>

var uri="my test.asp?name=ståle&car=saab";

document.write(encodeURI(uri)+ "<br>");

document.write(decodeURI(uri));

</script>

Output -

my%20test.asp?name=st%C3%A5le&car=saab

my test.asp?name=ståle&car=saab

**81. Why it is not advised to use innerHTML in JavaScript?**

innerHTML content is refreshed every time and thus is slower. There is no scope for validation in innerHTML and, therefore, it is easier to insert rouge code in the document and, thus, make the web page unstable.

**82. What does the following statement declares?**

var myArray = [[[]]];

It declares a three dimensional array.

**83. How are JavaScript and ECMA Script related?**

ECMA Script are like rules and guideline while Javascript is a scripting language used for web development.

**84. What is namespacing in JavaScript and how is it used?**

Namespacing is used for grouping the desired functions, variables etc. under a unique name. It is a name that has been attached to the desired functions, objects and properties. This improves modularity in the coding and enables code reuse.

**85. How can JavaScript codes be hidden from old browsers that don't support JavaScript?**

For hiding JavaScript codes from old browsers:

Add "<!--" without the quotes in the code just after the <script> tag.

Add "//-->" without the quotes in the code just before the <script> tag.

Old browsers will now treat this JavaScript code as a long HTML comment. While, a browser that supports JavaScript, will take the "<!--" and "//-->" as one-line comments.

Day 3.3:

JavaScript Unit Testing Frameworks

## **What is JavaScript?**

* JavaScript is a programming language which is defined as high level, dynamic and interpreted language used with HTML web applications
* JavaScript is also used for other than web documents such as PDFs and desktop widgets and became popular for server-side web application
* JavaScript is object-based script and follows the prototype

## **JavaScript Unit Testing**

**JavaScript Unit Tests**are usually run in the browser or on the frontend. JavaScript Test code is written for a page of the website or a single module of an application, and then this code is combined with HTML as an inline event handler. These unit tests are organized one by one in a Suite. Each and every suite contains number of tests designed to be executed for a separate module. Most importantly they don't conflict with any other module and runs with fewer dependencies on each other (some critical situation may cause dependencies).

## **Challenges in JavaScript Unit Testing**

There are certain problems one can find while performing JavaScript Unit Testing such as;

1. Many other languages support unit testing in browsers, in the stable as well as in runtime environment but JavaScript can not
2. You can understand some system actions with other languages, but this is not the case with JavaScript
3. Some JavaScript are written for a web application may have multiple dependencies
4. JavaScript is good to use in combination with HTML and CSS rather than on the web
5. Difficulties with page rendering and DOM manipulation

To avoid such issues what you can do is;

1. Do not use global variables
2. Do not manipulate predefined objects
3. Design core functionalities based on library
4. Try to create small pieces of functionalities with lesser dependencies

## **JavaScript Unit Testing Frameworks**

Sometimes you find the error message on your screen regarding such as 'Unable to load example.js' or any other JavaScript error regarding version control, these vulnerabilities comes under JavaScript Unit Testing.

We will review some tools and frameworks that are being used to perform JavaScript Unit Testing.

1. [Unit.js](http://unitjs.com/): It is known as an open source assertion library running on browser and Node.js. It is extremely compatible with other JavaScript Unit Testing frameworks like Mocha, Karma, Jasmine, QUnit, Protractor, etc. Provides the full documented API of assertion list

2. [QUnit](https://qunitjs.com/): It is used for both client-side as well as server-side JavaScript Unit Testing. This Free framework is used for jQuery projects. It follows Common JS unit testing Specification for unit testing. It supports the Node Long-term Support Schedule.

3. [Jasmine](https://jasmine.github.io/): Jasmine is the behavior-driven development framework for JavaScript unit Testing. It is used for testing both synchronous and asynchronous JavaScript Code. It does not require DOM and comes with the easy syntax that can be Written for any test.

4. [Karma](https://karma-runner.github.io/2.0/index.html): Karma is an open source productive testing environment. Easy workflow control Running on the command line. Offers the freedom to write the tests with Jasmine, Mocha, and QUnit. You can run the test on real devices with easy debugging.

5. [Mocha](https://mochajs.org/): Mocha runs on Node.js and in the browser. Mocha performs asynchronous Testing in a simpler way. Provides accuracy and flexibility in reporting. Provides tremendous support of rich features such as test-specific timeouts, JavaScript APIs etc.

6. [Jest](https://jestjs.io/): Jest is used by Facebook so far to test all of the JavaScript code. It provides the 'zero-configuration' testing experience. Supports independent and non-interrupting running test without any conflict. Do not require any other setup configuration and libraries.

7. [AVA](https://github.com/avajs): AVA is simple JavaScript Unit Testing Framework. Tests are being run in parallel and serially. Parallel tests run without interrupting each other. AVA Supports asynchronous testing as well. AVA uses subprocesses to run the test.

## **Summary**

* JavaScript Unit Testing may become tedious and tricky sometimes as it is performed for the front-end basically. One can use the JS libraries to for adding little ease. The challenge might become bigger as JavaScript is getting incorporated with Node.js and TypeScript.
* You should keep three things in mind while performing the test such as; The feature that needs to be tested, the final output and the expected output. Some tools and framework may help you in performing this task. Above mentioned tool lists is mentioned with most popular and useful frameworks used for JavaScript Unit Testing.
* More than these with upcoming challenges in performing testing there, some more powerful frameworks and tools may get evolved in future.