Web Programming

Unit 3: Java Script

Compiled by: Pallavi Tawde

pallavi.sawant@vsit.edu.in

Vidyalankar School of Information Technology

NAAC Accredited College

Vidyalankar School of Information Technology Wadala (E), Mumbai
www.vsit.edu.in

VSIT | Vidyalankar School of Information Technology

NAAC ACCREDITED COLLEGE

Certificate

This is to certify that the e-book titled "Web Programming" comprises all elementary learning tools for a better understating of the relevant concepts. This e-book is comprehensively compiled as per the predefined eight parameters and guidelines.

Date: 05-12-2019

Pallavi

Mrs. Pallavi Tawde
Department of BSc IT

Unit III: Java Script

Contents:

- Introduction
- Variables
- Operators
- Control statements
- Functions
- Events
- Properties and methods
- Objects
- Regular expression

Recommended Books

- JavaScript 2.0: The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider

Prerequisites and Linking

Unit III	Pre-requisites	Linking				
Java Script	Sem. I	Sem. II	Sem. III	Sem. IV	Sem. V	Sem. VI
	Imperative Programming	-	Python Programming	Core Java	Advanced Java	

WHAT IS JAVASCRIPT?

JavaScript is the scripting language of the Web.
JavaScript is used in millions of Web pages to add functionality, validate forms, detect browsers, and much more.

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without
- preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Are Java and JavaScript the same?

NO!
Java and JavaScript are two completely different languages in both concept and design!
Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript do?

- ***JavaScript gives HTML designers a programming tool :*** HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small of code into their HTML pages

- ***JavaScript can put dynamic text into an HTML page :*** A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page

- ***JavaScript can react to events :*** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

- ***JavaScript can read and write HTML elements :*** A JavaScript can read and change the content of an HTML element

- ***JavaScript can be used to validate data :*** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing

- ***JavaScript can be used to detect the visitor's browser :*** A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser

- ***JavaScript can be used to create cookies :*** A JavaScript can be used to store and retrieve information on the visitor's computer

Where to Put the JavaScript

JavaScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, or at a later event, such as when a user clicks a button. When this is the case we put the script inside a function.

Scripts in <head>

Scripts to be executed when they are called, or when an event is triggered, are placed in functions.

Put your functions in the head section, this way they are all in one place, and they do not interfere with page content.

Example

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event");
}
</script>
</head>
<body onload="message()">
</body>
</html>
```

Scripts in <body>

If you don't want your script to be placed inside a function, or if your script should write page content, it should be placed in the body section.

Example

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("This message is written by JavaScript");
</script>
</body>
</html>
```

Scripts in <head> and <body>

You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

Example

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event");
}
</script>
</head>
<body onload="message()">
<script type="text/javascript">
document.write("This message is written by JavaScript");
</script>
</body>
</html>
```

How to use External JavaScript file
If you want to run the same JavaScript on several pages, without having to write the same script on every page, you can write a JavaScript in an external file.
Save the external JavaScript file with a .js file extension.

Note: The external script cannot contain the <script></script> tags!
To use the external script, point to the .js file in the "src" attribute of the <script> tag:

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

The example below shows how to use JavaScript to write text on a web page:
```
<html>
<body>
<script type="text/javascript">
document.write("This is my first JavaScript!");
</script>
</body>
</html>
```

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

JavaScript Statements
A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do.

This JavaScript statement tells the browser to write "Hello Dolly" to the web page:

document.write("Hello Dolly");

It is normal to add a semicolon at the end of each executable statement. Most people think this is a good programming practice, and most often you will see this in JavaScript examples on the web.

The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Because of this you will often see examples without the semicolon at the end.

Note: Using semicolons makes it possible to write multiple statements on one line.

JavaScript Code
JavaScript code (or just JavaScript) is a sequence of JavaScript statements. Each statement is executed by the browser in the sequence they are written.

This example will write a heading and two paragraphs to a web page:

Example
```
<script type="text/javascript"> document.write("<h1>This
is a heading</h1>"); document.write("<p>This is a
paragraph.</p>"); document.write("<p>This is another
paragraph.</p>");
</script>
```

JavaScript Blocks
JavaScript statements can be grouped together in blocks.
Blocks start with a left curly bracket {, and ends with a right curly bracket }.
The purpose of a block is to make the sequence of statements execute together. This example will write a heading and two paragraphs to a web page:

Example
```
<script type="text/javascript">
{
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
}
</script>
```

JavaScript Comments
Comments can be added to explain the JavaScript, or to make the code more readable.
Single line comments start with //.

The following example uses single line comments to explain the code:

Example
```
<script type="text/javascript">
// Write a heading
document.write("<h1>This is a heading</h1>");
// Write two paragraphs: document.write("<p>This
is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Multi-Line Comments
Multi line comments start with /* and end with */.
The following example uses a multi line comment to explain the code:

Example
```
<script type="text/javascript">
/*
The code below will write
one heading and two paragraphs
*/
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

Using Comments to Prevent Execution
In the following example the comment is used to prevent the execution of a single code line (can be suitable for debugging):

Example
```
<script type="text/javascript">
//document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

Introduction to JavaScript

https://www.youtube.com/watch?v=yQaAGmHNn9s

JavaScript Variables

Variables are "containers" for storing information.

As with algebra, JavaScript variables are used to hold values or expressions.
A variable can have a short name, like x, or a more descriptive name, like carname. Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)

- Variable names must begin with a letter or the underscore character

Declaring (Creating) JavaScript Variables
Creating variables in JavaScript is most often referred to as "declaring" variables. You can declare JavaScript variables with the **var** keyword:

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).
However, you can also assign values to the variables when you declare them: var

```
x=5;
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.
Note: When you assign a text value to a variable, use quotes around the value.

Assigning Values to Undeclared JavaScript Variables
If you assign values to variables that have not yet been declared, the variables will automatically be declared.
These statements:

```
 x=5;
carname="Volvo";
```

have the same effect as:

```
var x=5;
var carname="Volvo";
```

Redeclaring JavaScript Variables

If you redeclare a JavaScript variable, it will not lose its original value.

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

After the execution of the statements above, the variable x will still have the value of 5. The value of x is not reset (or cleared) when you redeclare it.

JavaScript Arithmetic
As with algebra, you can do arithmetic operations with JavaScript variables: y=x-

```
5;
z=y+5;
```

JavaScript Arithmetic Operators
Arithmetic operators are used to perform arithmetic between variables and/or values.
Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators
Assignment operators are used to assign values to JavaScript variables. Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings
The + operator can also be used to add string variables or text values together. To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".
To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains: "What a very nice day"

Adding Strings and Numbers
The rule is: **If you add a number and a string, the result will be a string!**

Example

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

<u>Output:</u>

```
10
55
55
55
```

The rule is: If you add a number and a string, the result will be a string.

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Comparison Operators
Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that **x=5**, the table below explains the comparison operators:

Operator	**Description**	**Example**
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

How Can it be Used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

if (age<18) document.write("Too young");

You will learn more about the use of conditional statements in the next chapter of this tutorial.

Logical Operators

Logical operators are used to determine the logic between variables or values Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	**Description**	**Example**
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax:

variablename=(condition)?value1:value2

Example

greeting=(visitor=="PRES")?"Dear President ":"Dear ";

If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President " else it will be assigned "Dear".

Control Statements

1. **Conditional Statements**

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.
In JavaScript we have the following conditional statements:

- ***if statement :*** use this statement to execute some code only if a specified condition is true
- ***if...else statement :*** use this statement to execute some code if the condition is true and another code if the condition is false
- ***if...else if....else statement :*** use this statement to select one of many blocks of code to be executed
- ***switch statement :*** use this statement to select one of many blocks of code to be

executed

If Statement
Use the if statement to execute some code only if a specified condition is true.

Syntax

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

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Example

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10
var d=new Date();
var time=d.getHours();
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
</script>
```

Notice that there is no ..else.. in this syntax. You tell the browser to execute some code only if the specified condition is true.

If...else Statement
Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.
Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
else
  {
  code to be executed if condition is not true
  }
```

Example

```
<script type="text/javascript">
//If the time is less than 10, you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting. var
d = new Date();
var time = d.getHours(); if
(time < 10)
  {
  document.write("Good morning!");
  }
 else
  {
```

```
  document.write("Good day!");
  }
</script>
```

If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

<u>Syntax</u>

```
if (condition1)
  {
  code to be executed if condition1 is true
  }
else if (condition2)
  {
  code to be executed if condition2 is true}
 else
  {
  code to be executed if condition1 and condition2 are not true
  }
```

Example

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours() if
(time<10)
  {
  document.write("<b>Good morning</b>");
  }
else if (time>10 && time<16)
  {
  document.write("<b>Good day</b>");
  }
 else
  {
  document.write("<b>Hello World!</b>");
  }
</script>
```

The JavaScript Switch Statement

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

<u>Syntax</u>

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  **break;**
**default:**
  ***code to be executed if n is different from case 1 and 2***
**}**
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

Example

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
theDay=d.getDay();


switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this weekend!");
}
</script>
```

Conditional Statements

https://www.youtube.com/watch?v=5gjr15aWp24

2. JavaScript Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
}
```

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

The while Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.
The while loop loops through a block of code while a specified condition is true.

Syntax

```
while (var<=endvalue)
  {
  code to be executed
  }
```

Note: The <= could be any comparing operator.

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
</script>
</body>
```

```
</html>
```

The do...while Loop
The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
do
  {
  code to be executed
  }
while (var<=endvalue);
```

Example
The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
while (i<=5);
</script></body></html>
```

https://www.youtube.com/watch?v=XfzzTc1sQO4

JavaScript Break and Continue Statements The

break Statement
The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example
```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
     break;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body>
</html>
```

The continue Statement
The continue statement will break the current loop and continue with the next value.

Example
```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    continue;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body>
```

JavaScript For...In Statement

The for...in statement loops through the elements of an array or through the properties of an object.

Syntax

```
for (variable in object)
  {
  code to be executed
  }
```

Note: The code in the body of the for...in loop is executed once for each element/property.

Note: The variable argument can be a named variable, an array element, or a property of an object.

Example

Use the for...in statement to loop through an array:

Example

```
<html>
<body>
<script type="text/javascript">
var x;
var mycars = new Array();// array declaration
 mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (x in mycars)
  {
  document.write(mycars[x] + "<br>");
  }
</script>
</body>
</html>
```

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

alert("sometext");

Example

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
}
</script>
</head>
<body>
<input type="button" onclick="show_alert()" value="Show alert box" />
</body>
</html>
```

Confirm Box

A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

confirm("sometext");

Example

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button"); if
(r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>
<input type="button" onclick="show_confirm()" value="Show confirm box" />
</body>
</html>
```

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

Example

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter"); if
(name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</head>
<body>
<input type="button" onclick="show_prompt()" value="Show prompt box" />
</body>
</html>
```

JavaScript Functions

A function will be executed by an event or by a call to the function.

JavaScript Functions

To keep the browser from executing a script when the page loads, you can put your script into a function.
A function contains code that will be executed by an event or by a call to the function.
You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).
Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could

be wise to put functions in the <head> section.

How to Define a Function

Syntax

```
function functionname(var1,var2,...,varX)
{
some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The {and the} defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

JavaScript Function Example

Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

If the line: alert("Hello world!!") in the example above had not been put within a function, it would have been executed as soon as the page was loaded. Now, the script is not executed before a user hits the input button. The function displaymessage() will be executed if the input button is clicked.
You will learn more about JavaScript events in the JS Events chapter.

The return Statement

The return statement is used to specify the value that is returned from the function.
So, functions that are going to return a value must use the return statement. The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script>
</body>
</html>
```

JavaScript Comma Operator

Comma operator(,) evaluates both of its operands and returns the value of the second operand. This operator is primarily used inside a for loop,to allow multiple variables to be updated each time through the loop.

Javascript new

Javascript new operator creates an instance of an object. Object type can be user-defined or built-in. new is the keyword.

Javascript new Syntax:

new constructor(arguments);

constructor: constructor of the object. constructor has the same name of object

JavaScript new with built-in object

```
<html>
<body>
<script type="text/javascript">
var str = new String("HelloWorld");
document.write(str);
</script>
</html>
```

Javascript delete

Javascript delete operator deletes property of an object. delete is the keyword.

Javascript delete Syntax:

delete object.property;
delete object["property"];
delete object[index];

Returns :
returns false only if the property exists and cannot be deleted. It returns true in all other cases.

JavaScript delete example

```
<html>
<body>
<script type="text/javascript">
var obj=new Array(0,1,2);
document.writeln(obj+"<br>"); //print 0,1,2

var ret = delete obj[0]; //deleting 0 index
document.writeln(obj+"<br>"); //print ,1,2
document.writeln("return="+ret+"<br>"); //prints true

document.writeln("<br>");

var student = new Student("Ravi",30); //new creates object
student.dump(); //prints c

delete student.age; //age is deleted.
student.dump(); //prints Ravi,undefined.ie. here age is undefined.
```

```
function Student(name,age)
{
this.name=name;
this.age=age;
this.dump= function()
    {
    document.writeln(this.name+","+this.age+"<br>");
    }
}
</script>
</html>
```

Javascript this

Javascript this refers to global object.But it behaves differently compared to other langues. this is the keyword.

JavaScript this inside a function

```
<html>
<body>
<script type="text/javascript">
function Student(name)
{
    this.name=name; //this represents object.
    this.getName = function()
    {
           return this.name; //this represents object.
    }
}
 var student = new Student("Ravi");
 document.writeln(student.getName());
</script>
</html>
```

Javascript void

Javascript void operator evaluates the given expression and then returns undefined. Normally void is used in HTML code to evaluate expression. ex: void(0). void is the keyword.

Javascript void Syntax: **void(expression);**

JavaScript void example

```
<html>
<script type="text/javascript">
 </script>
<body>
<a href="javascript:void(0);">Click Here</a> Clicking here will not do anything
<br/>

<a href="javascript:void(document.body.style.backgroundColor='gray');"> Click
Here </a> Clicking here will change the background color<br/>

<a href="javascript:void(alert('Hi'));">
Click Here
</a> Clicking here will show alert message </html>
```

JavaScript Events
Events are actions that can be detected by JavaScript.
By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.
Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.
Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

onLoad and onUnload
The onLoad and onUnload events are triggered when the user enters or leaves the page.
The onLoad event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

Both the onLoad and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

onFocus, onBlur and onChange
The onFocus, onBlur and onChange events are often used in combination with validation of form fields.

Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

onSubmit
The onSubmit event is used to validate ALL form fields before submitting it.
Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.htm" onsubmit="return checkForm()">
```

onMouseOver and onMouseOut
onMouseOver and onMouseOut are often used to create "animated" buttons.
Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com" onmouseover="alert('An onMouseOver
event');return false"><img src="w3s.gif" alt="W3Schools " /></a>
```

Events

https://www.youtube.com/watch?v=9rvB27xXO_I

JavaScript Objects Introduction

JavaScript is an Object Oriented Programming (OOP) language.
An OOP language allows you to define your own objects and make your own variable types.

Object Oriented Programming

JavaScript is an Object Oriented Programming (OOP) language. An OOP language allows you to define your own objects and make your own variable types.
However, creating your own objects will be explained later, in the Advanced JavaScript section. We will start by looking at the built-in JavaScript objects, and how they are used. The next pages will explain each built-in JavaScript object in detail.
Note that an object is just a special kind of data. An object has properties and methods.

Properties

Properties are the values associated with an object.
In the following example we are using the length property of the String object to return the number of characters in a string:

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

The output of the code above will be: 12

Methods

Methods are the actions that can be performed on objects.
In the following example we are using the toUpperCase() method of the String object to display a text in uppercase letters:

```
    <script type="text/javascript">
    var str="Hello world!";
    document.write(str.toUpperCase());
    </script>
```

The output of the code above will be:

```
    HELLO WORLD!
```

JavaScript String Object

The String object is used to manipulate a stored piece of text.

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

<script type="text/javascript">
var txt = "Hello World!";
document.write("<p>Big: " + txt.big() + "</p>");
document.write("<p>Small: " + txt.small() + "</p>");
document.write("<p>Bold: " + txt.bold() + "</p>");
document.write("<p>Italic: " + txt.italics() + "</p>");
document.write("<p>Fixed: " + txt.fixed() + "</p>");
document.write("<p>Strike: " + txt.strike() + "</p>");

document.write("<p>Fontcolor: " + txt.fontcolor("green") + "</p>");
document.write("<p>Fontsize: " + txt.fontsize(6) + "</p>");
document.write("<p>Subscript: " + txt.sub() + "</p>");
document.write("<p>Superscript: " + txt.sup() + "</p>");
document.write("<p>Link: " + txt.link("http://www.w3schools.com") + "</p>");
document.write("<p>Blink: " + txt.blink() + " (does not work in IE, Chrome, or
Safari)</p>");
</script>
</body>
</html>
```

OUTPUT:

Big: Hello World!
Small: Hello World!
Bold: **Hello World!**
Italic: *Hello World!*

Fixed: Hello World!
Strike: ~~Hello World!~~
Fontcolor: Hello World!
Fontsize: Hello World!
Subscript: Hello World!
Superscript: Hello World!
Link: Hello World!
Blink: Hello World! (does not work in IE, Chrome, or Safari)

```
<html>
<body>
<script type="text/javascript">
var txt="Hello World!";
document.write(txt.toLowerCase() + "<br />");
document.write(txt.toUpperCase());
</script>
</body>
</html>
```

OUTPUT:

```
hello world!
HELLO WORLD!
```

String object

The String object is used to manipulate a stored piece of text.

Examples of use:

The following example uses the length property of the String object to find the length of a string:

```
var txt="Hello world!";
document.write(txt.length);
```

The code above will result in the following output:

```
12
```

The following example uses the toUpperCase() method of the String object to convert a string to uppercase letters:

```
var txt="Hello world!";
document.write(txt.toUpperCase());
```

The code above will result in the following output:

```
HELLO WORLD!
```

JavaScript Date Object

The Date object is used to work with dates and times.

1. **Return today's date and time**

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

```
<script type="text/javascript">
var d=new Date();
document.write(d);
</script>
</body>
</html>
```

o/p:- Wed Oct 20 12:43:35

2. **getTime()**

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
document.write(d.getTime() + " milliseconds since 1970/01/01");
</script>
</body>
</html>
```

O/P: 1287560688703 milliseconds since 1970/01/01

Create a Date Object

The Date object is used to work with dates and times. Date objects are created with the Date() constructor.
There are four ways of instantiating a date:

new Date() // current date and time
new Date(milliseconds) //milliseconds since 1970/01/01 new
Date(dateString)
new Date(year, month, day, hours, minutes, seconds, milliseconds)

Most parameters above are optional. Not specifying, causes 0 to be passed in.

Once a Date object is created, a number of methods allow you to operate on it. Most methods allow you to get and set the year, month, day, hour, minute, second, and milliseconds of the object, using either local time or UTC (universal, or GMT) time.

All dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC) with a day containing 86,400,000 milliseconds.

Some examples of instantiating a date:

```
today = new Date()
d1 = new Date("October 13, 1975 11:13:00") d2
= new Date(79,5,24)
d3 = new Date(79,5,24,11,33,0)
```

Set Dates

We can easily manipulate the date by using the methods available for the Date object.
In the example below we set a Date object to a specific date (14th January 2010): var

```
myDate=new Date();
myDate.setFullYear(2010,0,14);
```

And in the following example we set a Date object to be 5 days into the future:

```
var myDate=new Date();
myDate.setDate(myDate.getDate()+5);
```

Note: If adding five days to a date shifts the month or year, the changes are handled automatically by the Date object itself!

Compare Two Dates
The Date object is also used to compare two dates.

The following example compares today's date with the 14th January 2010: var
```
myDate=new Date();
myDate.setFullYear(2010,0,14);
var today = new Date();

if (myDate>today)
  {
  alert("Today is before 14th January 2010");
  }
else
  {
  alert("Today is after 14th January 2010");
  }
```

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

What is an Array?
An array is a special variable, which can hold more than one value, at a time.
If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
cars1="Saab";
cars2="Volvo";
cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array!
An array can hold all your variable values under a single name. And you can access the values by referring to the array name.
Each element in the array has its own ID so that it can be easily accessed.

Create an Array
An array can be defined in three ways.
The following code creates an Array object called myCars:

i)
```
var myCars=new Array(); // regular array (add an optional integer
myCars[0]="Saab";       // argument to control array's size)
```

```
myCars[1]="Volvo";
myCars[2]="BMW";
```

ii) var myCars=new Array("Saab","Volvo","BMW"); // condensed array

iii) var myCars=["Saab","Volvo","BMW"]; // literal array

Note: If you specify numbers or true/false values inside the array then the variable type will be Number or Boolean, instead of String.

- **<u>Create an array</u>**

```
<html>
<body>

<script type="text/javascript">
var mycars = new Array();
mycars[0] = "Saab"; mycars[1]
= "Volvo"; mycars[2] =
"BMW";

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

<u>Output</u>

```
Saab
Volvo
BMW
```

 <u>For...In Statement</u>

```
 <html>
<body>
<script
type="text/javascript"> var x;
var mycars = new
Array(); mycars[0] =
"Saab"; mycars[1] =
"Volvo"; mycars[2] =
"BMW";

for (x in mycars)
{
document.write(mycars[x] + "<br />");
}
</script>
</body>
</html>
```

Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

The following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Saab
```

Modify Values in an Array

To modify a value in an existing array, just add a new value to the array with a specified index number:

```
myCars[0]="Opel";
```

Now, the following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Opel
```

https://www.youtube.com/watch?v=nEvBcwlpkBQ

JavaScript Boolean Object

The Boolean object is used to convert a non-Boolean value to a Boolean value (true or false).

Create a Boolean Object

The Boolean object represents two values: "true" or "false".
The following code creates a Boolean object called myBoolean:

var myBoolean=new Boolean();

Note: If the Boolean object has no initial value or if it is 0, -0, null, "", false, undefined, or NaN, the object is set to false. Otherwise it is true (even with the string "false")!

All the following lines of code create Boolean objects with an initial value of false: var
myBoolean=new Boolean();
var myBoolean=new Boolean(0); var
myBoolean=new Boolean(null); var
myBoolean=new Boolean(""); var
myBoolean=new Boolean(false); var
myBoolean=new Boolean(NaN);

And all the following lines of code create Boolean objects with an initial value of true: var
myBoolean=new Boolean(1);
var myBoolean=new Boolean(true); var
myBoolean=new Boolean("true"); var
myBoolean=new Boolean("false");
var myBoolean=new Boolean("Richard");

JavaScript Math Object

```
<html>
<body>
<script type="text/javascript">
document.write(Math.round(0.60) + "<br />");
document.write(Math.round(0.50) + "<br />");
document.write(Math.round(0.49) + "<br />");
document.write(Math.round(-4.40) + "<br />");
document.write(Math.round(-4.60));
</script>
</body>
</html>
```

<u>OUTPUT:</u>

```
1
1
0
-4
-5
```

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

```
//return a random number between 0 and 1
document.write(Math.random() + "<br />");
//return a random integer between 0 and 10
document.write(Math.floor(Math.random()*11));
</script>
</body>
</html>
```

OUTPUT:

```
0.39987146027751624
5
```

```
<html>
<body>
<script type="text/javascript">
document.write(Math.max(5,10) + "<br />");
document.write(Math.max(0,150,30,20,38) + "<br />");
document.write(Math.max(-5,10) + "<br />");
document.write(Math.max(-5,-10) + "<br />");
document.write(Math.max(1.5,2.5));
</script>
</body>
</html>
```

JavaScript RegExp Object

RegExp, is short for regular expression. Regular expression is an object that describes a pattern of characters.When you search in a text, you can use a pattern to describe what you are searching for.

A simple pattern can be one single character.

A more complicated pattern can consist of more characters, and can be used for parsing, format checking, substitution and more.

Regular expressions are used to perform powerful pattern-matching and "search-and-replace" functions on text.

Syntax

var txt=new RegExp(pattern,modifiers);

or more simply:

var txt=/pattern/modifiers;

- pattern specifies the pattern of an expression
- modifiers specify if a search should be global, case-sensitive, etc.

RegExp Modifiers

Modifiers are used to perform case-insensitive and global searches.
The i modifier is used to perform case-insensitive matching. The g modifier is used to perform a global match (find all matches rather than stopping after the first match).

Example 1

Do a case-insensitive search for "w3schools" in a string: var

```
str="Visit W3Schools";
var patt1=/w3schools/i;
```

The marked text below shows where the expression gets a match: Visit
W3Schools

Do a global search for "is":

```
var str="Is this all there is?";
var patt1=/is/g;
```

The marked text below shows where the expression gets a match: Is this
all there is?

test()

The test() method searches a string for a specified value, and returns true or false, depending on the result.

The following example searches a string for the character "e":

Example

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
true
```

exec()

The exec() method searches a string for a specified value, and returns the text of the found value. If no match is found, it returns *null.*

The following example searches a string for the character "e":

Example 1

```
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
e
```

Form Validation

https://www.youtube.com/watch?v=RzVURjrIFxo

□ □ □ □ □

- **GQs**

1. What is Javascript?
2. Write a short note on Relational operators and logical operators in Javascript.
3. Explain the use of if else statement in javascript with an example.
4. What is the need of scripting language. List the different scripting languages.
5. Explain client side & server side scripting.
6. Explain the different types of operators available in java script.
7. Explain the different types of conditional statements.
8. Explain while loop with example.
9. Explain for loop with example.
10. Explain For..In statement in Java
11. Explain switch statement with example.
12. Explain the use of break & continue statements.
13. What is array? Explain with example.
14. Define event. Explain different types of events.
15. Explain Data, Math & string functions with e.g.
16. Explain Image object with example.
17. Write a javascript code to accept a character. Display if it is a vowel or a consonant.
18. Write a javascript function to accept a number. Calculate its factorial.
19. Write a javascript function to display tables from 2 to 10 in tabular format.
20. Design a HTML page to accept 3 numbers as sides of a triangle.write a javascript code to check if the triangle is right angled, isosceles, equilateral or invalid. Consider the following right angled -----if a2+b2=c2
    Isosceles-----------if any two sides are equal
    Equilateral---------if all sides are equal
    Three values can be the dimensions of a triangle if and only if the sum of every pair of values is greater than the third value. Otherwise,it is an invalid triangle.
21. Write a javascript function to print all prime numbers from 1 to 100
22. Write a javascript function to print sum of ten numbers taken from user.
23. Write a javascript program to find sum of digit of entered number.
24. Write a javascript program to find reverse of number.
25. Write a javascript program to check whether entered number is prime or not.

- **Multiple Choice Questions:**

1.	Javascript is_______________language	
	a. Client side	b. Server side
	c. static	d. None
2.	The browser uses_____tag to detect javascript.	
	a. <js></js>	b. <scripting></scripting>
	c. <script type=”text/javascript”> <script>	d. <javascript></javascript>
3.	Consider the following snippet code **var** string1 = ”123”; **var** intvalue = 123; alert(string1 + intvalue); The result would be	
	a. 123246	b. 246
	c. Exception	**d. 123123**
4.	What is the use of "this" keyword in javascript?	
	a. It refers to current object	b. It referes to previous object
	c. It is a variable which contains value	d. None of the above
5.	A .... mostly used to take users choice on any option and displays a dialog box with two buttons Ok and Cancel.	
	a. Alert dialog box	b. Prompt dialog box
	c. Information dialog box	**d. Conformation dialog box**
6.	The JavaScript .... class represents regular expressions	
	a. RegExpObj	b. RegExpClass
	c. RegExp	**d.** StringExp
7.	Which of the following can't be done with client-side JavaScript?	
	a. Validating a form	**b.** Sending a form's contents by email
	c. Storing the form's contents to a database file on the server	None of the above d.
8.	Which of the following is not a valid JavaScript variable name?	
	a. 2names	b. FirstAndLast
	c. _first_and_last_names	d. Names2
9.	What is the correct JavaScript syntax to write "Hello World"?	
	a. System.out.println("Hello World")	b. println ("Hello World")

	c. document.write("Hello World")	d. response.write("Hello World")
10.	Which of the following event fires when the form element loses the focus: <button>, <input>, <label>, <select>, <textarea>?	
	a. onfocus	**b. onblur**
	c. onclick	d. ondblclick
11.	Which of the following is the structure of an if statement?	
	a. if (conditional expression is true) thenexecute this codeend if	b. if (conditional expression is true)execute this codeend if
	c. if (conditional expression is true) {then execute this code>->}	d. if (conditional expression is true) then {execute this code}
12.	Which of the following function of String object returns the calling string value converted to lower case?	
	a. toLocaleLowerCase()	b. tolwcase()
	c. toLowerCase()	d. toLowerString()
13.	What are variables used for in JavaScript Programs?	
	a. Storing numbers, dates, or other values	b. Varying randomly
	c. Causing high-school algebra flashbacks	d. None of the above
14.	What are the looping structures are available in javascripts?	
	a. For,foreach	b. Do-while, foreach
	c. Foreach,while	**d. For,while,do-while**
15.	What is the output of the following code snippet? var str = "Apple, Banana, Kiwi"; var res = str.substring(7,13);	
	a. Banana	b. Apple
	c. Kiwi	d. Banana,Kiwi