

# Unit-3

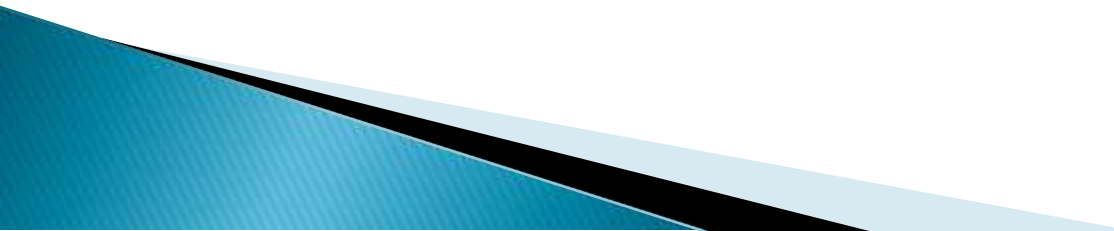
## Java script

By: Niral Jadav

# Introduction

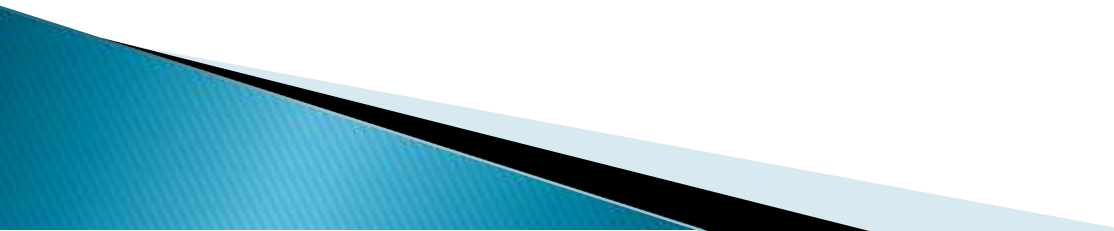
- ▶ JavaScript is a lightweight, interpreted programming language.
- ▶ It is designed for creating network-centric applications.
- ▶ It is complimentary to and integrated with Java.
- ▶ JavaScript is very easy to implement because it is integrated with HTML.

# Applications of Javascript Programming

- ▶ Client side validation
  - ▶ Manipulating HTML Pages
  - ▶ User Notifications
  - ▶ Back-end Data Loading
  - ▶ Presentations
  - ▶ Server Applications
- 

# Hello World using Javascript

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



# JavaScript Data Types

- ▶ JavaScript variables can hold different data types: numbers, strings, objects and more:

```
let length = 16;           // Number
let lastName = "Johnson"; // String
let x = {firstName:"John", lastName:"Doe"}; // Object
```

➤ let x = 16 + 4 + "Volvo";

Result:

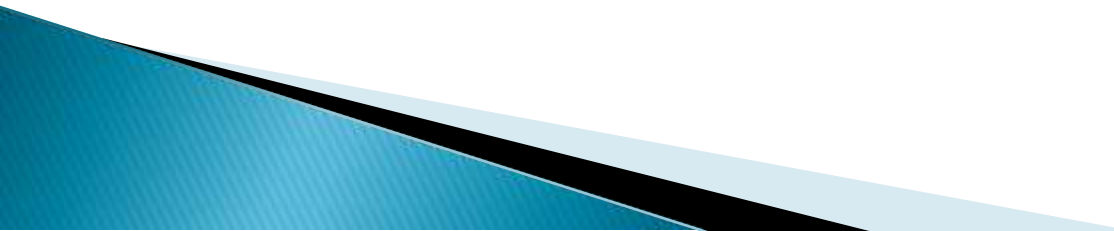
20Volvo

▶ let x = "Volvo" + 16 + 4;

Result:

Volvo164

```
let x;      // Now x is undefined
x = 5;      // Now x is a Number
x = "John"; // Now x is a String
let carName1 = "Volvo XC60"; // Using double quotes
let carName2 = 'Volvo XC60'; // Using single quotes
let x1 = 34.00; // Written with decimals
let x2 = 34;    // Written without decimals
let y = 123e5;  // 123000000
let z = 123e-5; // 0.00123
```



# JavaScript Booleans

```
let x = 5;
```

```
let y = 5;
```

```
let z = 6;
```

```
(x == y)    // Returns true
```

```
(x == z)    // Returns false
```



## JavaScript Arrays


```
const cars = ["Saab", "Volvo", "BMW"];
```

## JavaScript Objects

```
const person = {firstName:"John", lastName:"Doe", age:50,  
eyeColor:"blue"};
```

## The typeof Operator

```
typeof ""           // Returns "string"  
typeof "John"       // Returns "string"  
typeof 0             // Returns "number"  
typeof 314           // Returns "number"  
typeof 3.14          // Returns "number"  
typeof (3)           // Returns "number"  
typeof (3 + 4)       // Returns "number"
```



## Undefined

```
let car; // Value is undefined, type is undefined
```

## Empty Values

```
let car = ""; // The value is "", the typeof is "string"
```

# JavaScript Variables

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

```
<!--
```

```
var money;
```

```
var name;
```

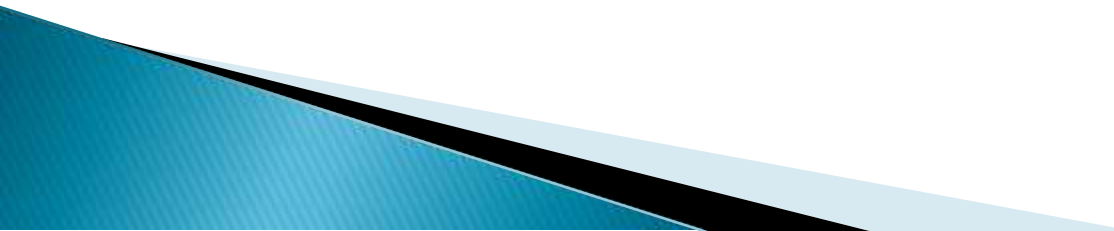
```
var age, dob;
```

```
//-->
```

```
</script>
```



```
<script type = "text/javascript">  
  <!--  
  var name = "Ali";  
  var money;  
    money = 2000.50;  
  //-->  
</script>
```



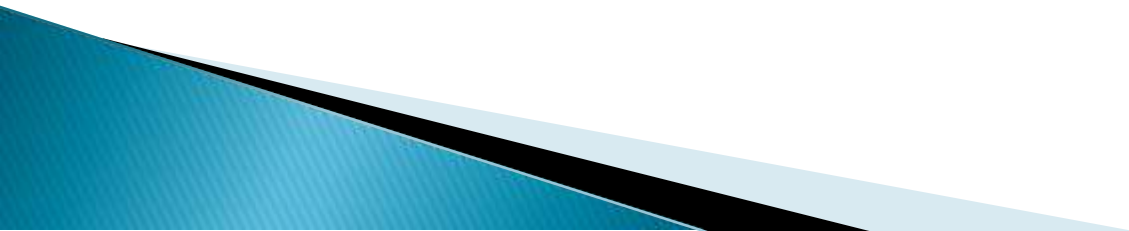
# JavaScript Variable Scope

## **Global Variables –**

A global variable has global scope which means it can be defined anywhere in your JavaScript code.

## **Local Variables –**

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



# JavaScript Arrays

- ▶ Syntax

```
const array_name = [item1, item2, ...];
```

# Array Properties and Methods

- ▶ `cars.length` // Returns the number of elements
- ▶ `cars.sort()` // Sorts the array

```
const fruits =  
["Banana", "Orange", "Apple", "Mango"];  
fruits.length; // Returns 4
```

```
const fruits =  
["Banana", "Orange", "Apple", "Mango"];  
fruits[0]; // Returns "Banana"
```

```
const fruits =  
["Banana", "Orange", "Apple", "Mango"];  
fruits[fruits.length - 1]; // Returns  
"Mango"
```

# Looping Array Elements

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];  
let fLen = fruits.length;  
  
text = "<ul>";  
for (let i = 0; i < fLen; i++) {  
  text += "<li>" + fruits[i] + "</li>";  
}  
text += "</ul>";
```



# Adding Array Elements

- ▶ `const fruits = ["Banana", "Orange", "Apple"];`  
`fruits.push("Lemon");` // Adds a new element (Lemon) to fruits

# Associative Arrays

```
const person = [];  
person[0] = "John";  
person[1] = "Doe";  
person[2] = 46;  
person.length;    // Will return 3  
person[0];        // Will return "John"
```

```
const person = {};  
person["firstName"] = "John";  
person["lastName"] = "Doe";  
person["age"] = 46;  
person.length;    // Will return 0  
person[0];        // Will return undefined
```

# How to Recognize an Array

```
const fruits = ["Banana", "Orange", "Apple"];  
typeof fruits;    // returns object
```

```
Array.isArray(fruits);    // returns true
```

```
const fruits = ["Banana", "Orange", "Apple"];  
  
fruits instanceof Array;    // returns true
```

# Array functions

## Add an item to the end of an Array

```
let newLength = fruits.push('Orange') //  
["Apple", "Banana", "Orange"]
```

## Remove an item from the end of an Array

```
let last = fruits.pop() // remove Orange (from the  
end)  
// ["Apple", "Banana"]
```

## Remove an item from the beginning of an Array

```
let first = fruits.shift() // remove Apple from the front  
// ["Banana"]
```

## Remove an item from the beginning of an Array

```
let first = fruits.unshift("Orange") // remove Apple from the front  
// ["Orange", "Banana"]
```

# Array functions

## Find the index of an item in the Array

```
fruits.push('Mango')  
// ["Strawberry", "Banana", "Mango"]  
  
let pos = fruits.indexOf('Banana')  
// 1
```


## Copy array

```
let shallowCopy = fruits.slice()  
// this is how to make a copy  
// ["Strawberry", "Mango"]
```

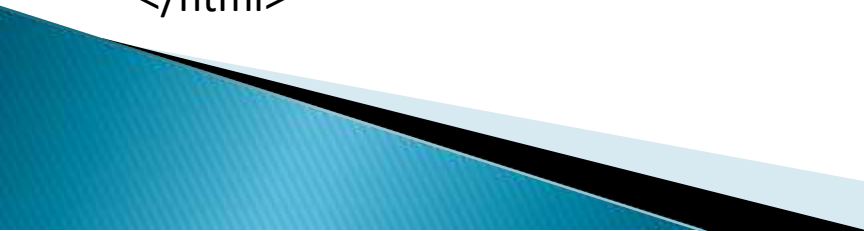
# JavaScript - Functions

## Syntax

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



```
<html>
<head>
  <script type = "text/javascript">
    function sayHello()
    {
      document.write ("Hello there!");
    }
  </script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type = "button" onclick = "sayHello()" value = "Say Hello">
</form>
<p>Use different text in write method and then try...</p>
</body>
</html>
```



# Conditions

## If condition

```
if(a > 10)
{
    alert("A is > that
10");
}
```

## If condition

```
if (hour < 18) {
    greeting = "Good day";
} else {
    greeting = "Good
evening";
}
```

```
max = a > b ? a : b ;
```

## switch

```
switch(expression)
{
    case lbl1:
        // code to execute
        break;
    case lbl2:
        // code to execute
        break;
}
```



# Loops

## for loop

Use when you know how many repetitions you want to do

### syntax

```
for(initialize ; condition  
; increment) { ... }
```

### example

```
for(x=0;x<10;x++) {  
    // Code Here  
}
```

## while loop

Loops through block of code while condition is true

### syntax

```
while(condition) { ... }
```

### example

```
while (x<10) {  
    //Code Here  
}
```

## do while loop

Execute block at least once then repeat while condition is true

### syntax

```
do{ ... } while  
(condition);
```

### example

```
do{  
    // Code Here  
} while (x<10)
```

## for loop

```
for (let i = 0; i < 5; i++)  
{  
    text += "The number  
is " + i + "<br>";  
}
```

## while loop

```
let count = 1; while  
(count < 10) {  
    console.log(count);  
    count +=2; }
```

## Do while loop

```
do {    text +=  
"<br>The number is " +  
i;    i++;  
    } while (i < 5);  
document.getElementById("demo").innerHTML =  
text;}
```

# Pop up Boxes

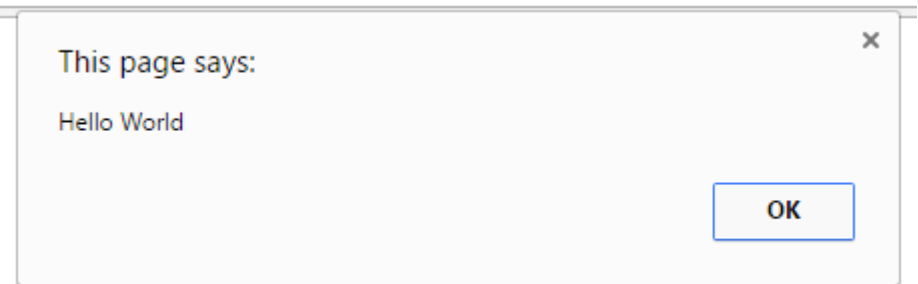
- ▶ Popup boxes can be used to raise an alert, or to get confirmation on any input or to have a kind of input from the users.
- ▶ JavaScript supports three types of popup boxes.
  - Alert box
  - Confirm box
  - Prompt box

# Alert Box

- ▶ An alert box is 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.
- ▶ It can be used to display the result of validation.

## Code

```
<html>
  <head>
    <title>Alert
Box</title>
  </head>
  <body>
    <script>
      alert("Hello
World");
    </script>
  </body>
</html>
```

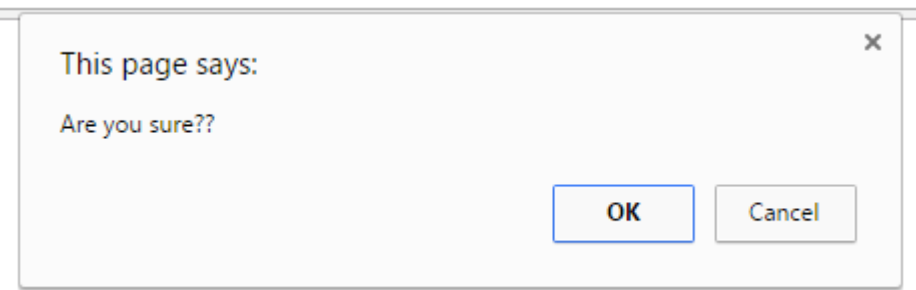


# Confirm Box

- ▶ A confirm box is used if you want the user to 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.
- ▶ Example :

## Code

```
<script>
  var a = confirm("Are you
sure??");
  if(a==true) {
    alert("User Accepted");
  }
  else {
    alert("User Canceled");
  }
</script>
```

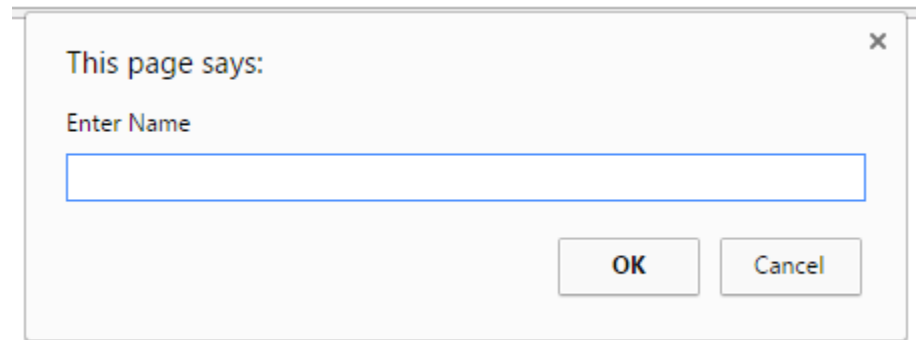


# Prompt Box

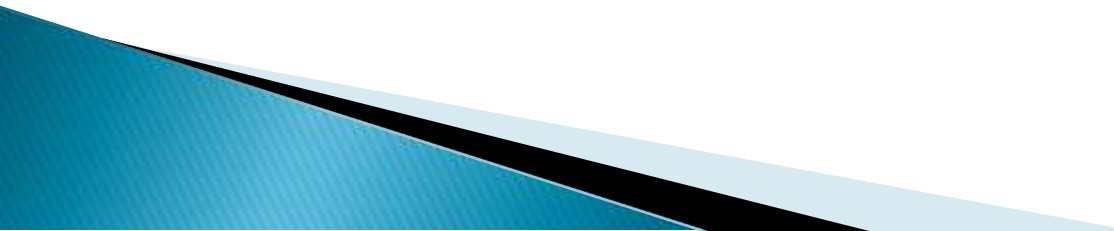
- ▶ A prompt box is used if you want the user to input a value.
- ▶ When a prompt box pops up, user have to click either "OK" or "Cancel" to proceed, If the user clicks "OK" the box returns the input value, If the user clicks "Cancel" the box returns null.
- ▶ Example:

## Code

```
<script>  
  var a = prompt("Enter Name");  
  alert("User Entered " + a);  
</script>
```



# JavaScript - Objects

- ▶ JavaScript is an Object Oriented Programming (OOP) language.
  - ▶ A programming language can be called object-oriented if it provides four basic capabilities to developers –
  - ▶ **Encapsulation** – the capability to store related information, whether data or methods, together in an object.
  - ▶ **Aggregation** – the capability to store one object inside another object.
  - ▶ **Inheritance** – the capability of a class to rely upon another class (or number of classes) for some of its properties and methods.
  - ▶ **Polymorphism** – the capability to write one function or method that works in a variety of different ways.
- 

## Object Properties:

- ▶ The syntax for adding a property to an object is –  
`objectName.objectProperty = propertyValue;`
- ▶ **For example** – The following code gets the document title using the "**title**" property of the **document** object.  
`var str = document.title;`

## Object Methods:

- ▶ **For example** – Following is a simple example to show how to use the **write()** method of document object to write any content on the document.  
`document.write("This is test");`

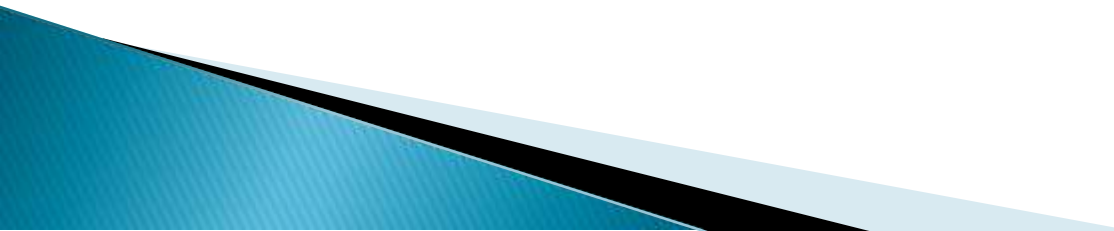
# User-Defined Objects

- ▶ The new Operator
- ▶ The **new** operator is used to create an instance of an object.
- ▶ To create an object, the **new** operator is followed by the constructor method.

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

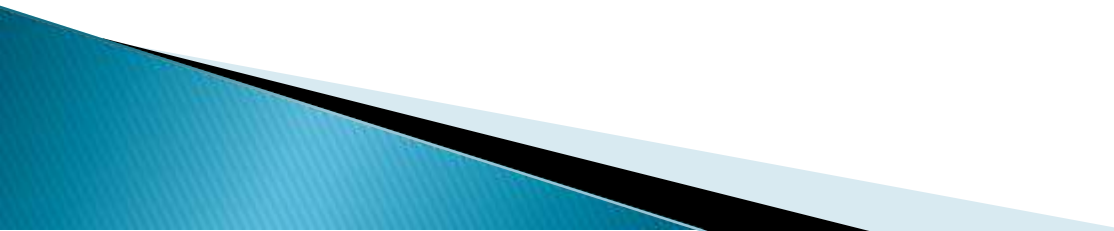
```
var books = new Array("C++", "Perl", "Java");
```

```
var day = new Date("August 15, 1947");
```

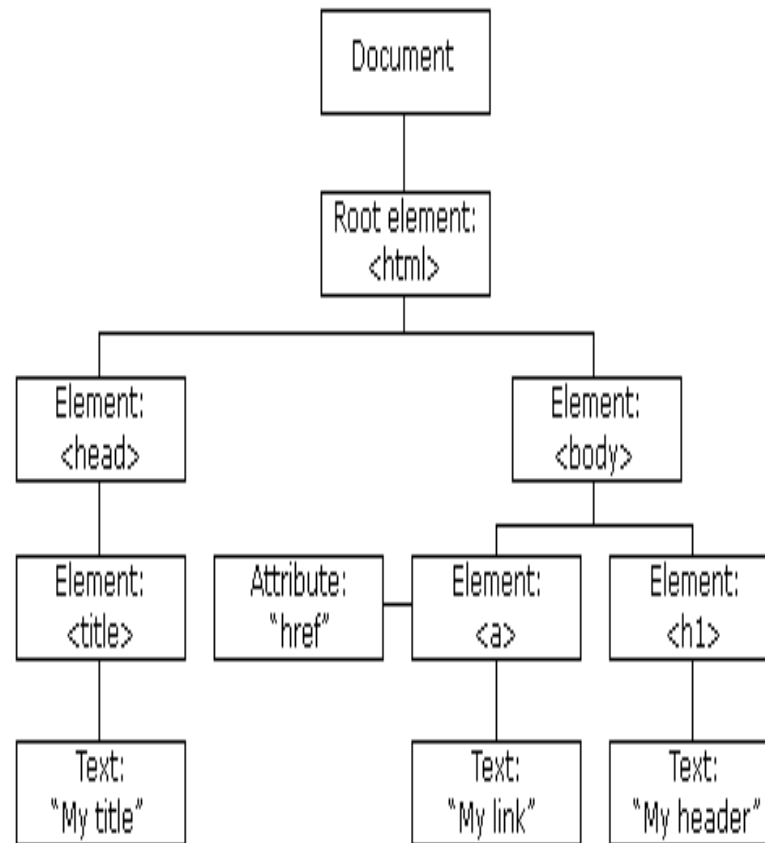




# The Object() Constructor

- ▶ The Object() Constructor
  - ▶ A constructor is a function that creates and initializes an object.
  - ▶ JavaScript provides a special constructor function called **Object()** to build the object.
  - ▶ The return value of the **Object()** constructor is assigned to a variable.
- 

```
<html>
  <head>
    <title>User-defined objects
  </title>
  <script type = "text/javascript">
    var book = new Object(); // Create the object
    book.subject = "JavaScript for impatient programmers ";
    // Assign properties to the object
    book.author = "Dr. Axel Rauschmayer ";
  </script>
</head>
<body>
  <script type = "text/javascript">
    document.write("Book name is : " + book.subject + "<br>");
    document.write("Book author is : " + book.author + "<br>");
  </script>
</body>
```



# Document Object Methods

Method	Description
<code>write()</code>	Writes HTML expressions or JavaScript code to a document
<code>writeln()</code>	Same as <code>write()</code> , but adds a newline character after each statement
<code>open()</code>	Opens an output stream to collect the output from <code>document.write()</code> or <code>document.writeln()</code>
<code>close()</code>	Closes the output stream previously opened with <code>document.open()</code>
<code>getElementById()</code>	Accesses element with a specified id
<code>getElementsByName()</code>	Accesses all elements with a specified name
<code>getElementsByTagName()</code>	Accesses all elements with a specified tag name
<code>setTimeout(), clearTimeout()</code>	Set a time period for calling a function once; or cancel it.

# getElementById()

➤ When we suppose to get the reference of the element from HTML in JavaScript using id specified in the HTML we can use this method.

➤ Example :

## HTML

```
<html>
  <body>
    <input type="text"
id="myText">
  </body>
</html>
```

## JavaScript

```
<script>
  function myFunction()
  {
    var txt = document.getElementById("myText");
    alert(txt.value);
  }
</script>
```

# getElementsByName()

- When we suppose to get the reference of the elements from HTML in JavaScript using name specified in the HTML we can use this method.
- It will return the array of elements with the provided name.
- Example :

## HTML

```
<html>
  <body>
    <input type="text"
      name="myText">
  </body>
</html>
```

## JavaScript

```
<script>
function myFunction()
{
  a=document.getElementsByName("myText")[0];
  alert(a.value);
}
</script>
```

# getElementsByTagName()

- When we suppose to get the reference of the elements from HTML in JavaScript using name of the tag specified in the HTML we can use this method.
- It will return the array of elements with the provided tag name.
- Example :

## HTML

```
<html>
  <body>
    <input type="text"
name="uname">
    <input type="text"
name="pword">
  </body>
</html>
```

## JavaScript

```
<script>
function myFunction() {

a=document.getElementsByTagName("input
");
  alert(a[0].value);
  alert(a[1].value);
}
</script>
```

# Forms using DOM

➤ We can access the elements of form in DOM quite easily using the name/id of the form.

➤ Example :

## HTML

```
<html>
  <body>
    <form name="myForm">
      <input type="text"
name="uname">
      <input type="text"
name="pword">
      <input type="button"
onClick="f()">
    </form>
  </body>
</html>
```

## JS

```
function f()
{
  var a = document.forms["myForm"];
  var u = a.uname.value;
  var p = a.pword.value;
  if(u=="admin" && p=="123")
  {
    alert("valid");
  }
  else
  {
    alert("Invalid");
  }
}
```



# JavaScript Built-in Functions

## constructor()

- ▶ Returns the function that created this object's instance. By default this is the Number object.

## toString()

- ▶ Returns the string representation of the number's value.

## concat()

- ▶ Combines the text of two strings and returns a new string.

## length()

- ▶ Returns the length of the string.

## split()

- ▶ Splits a String object into an array of strings by separating the string into substrings.
- 

### substr()

- ▶ Returns the characters in a string beginning at the specified location through the specified number of characters.

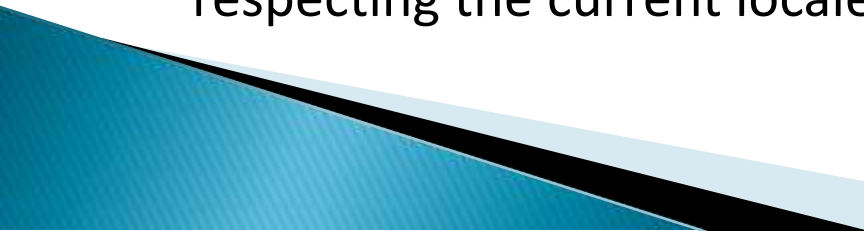
### substring()

- ▶ Returns the characters in a string between two indexes into the string.

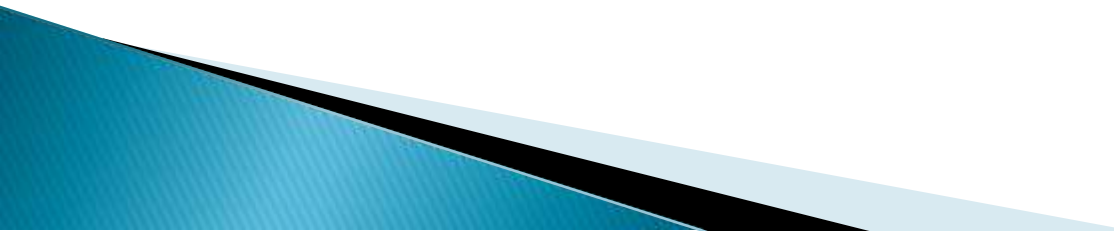
### toLocaleLowerCase()

- ▶ The characters within a string are converted to lower case while respecting the current locale.

### toLocaleUpperCase()

- ▶ The characters within a string are converted to upper case while respecting the current locale.
- 

# Validation

- ▶ Validation is the process of **checking** data against a **standard** or **requirement**.
  - ▶ Form validation normally used to occur at the server, after client entered necessary data and then pressed the Submit button.
  - ▶ If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information.
  - ▶ This was really a lengthy process which used to put a lot of burden on the server.
  - ▶ JavaScript provides a way to validate form's data on the client's computer before sending it to the web server.
- 

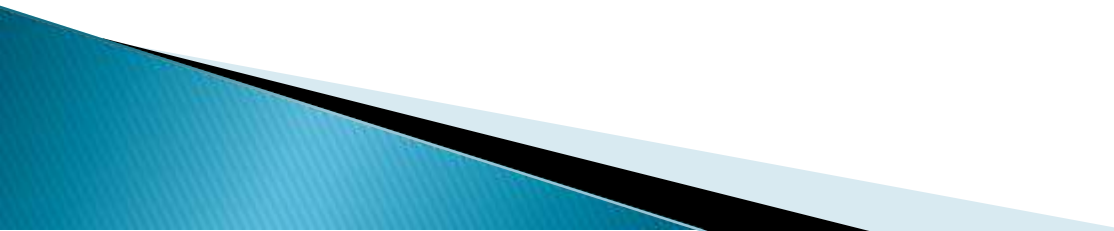
Form validation generally performs two functions.

### 1. Basic Validation

- Emptiness
- Confirm Password
- Length Validation etc.....

### 2. Data Format Validation

Secondly, the data that is entered must be checked for correct **form** and **value**.

- Email Validation
  - Mobile Number Validation
  - Enrollment Number Validation etc....
- 

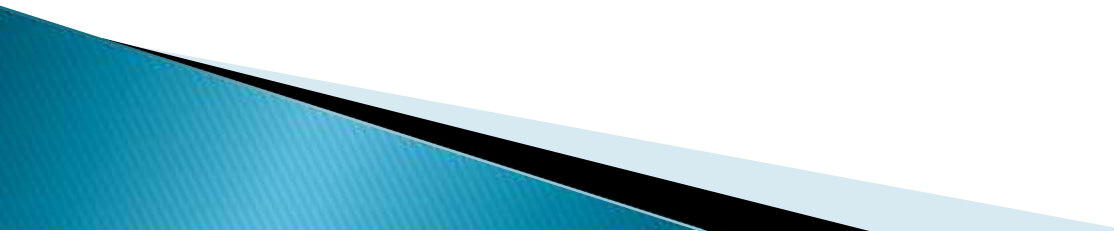
# RegExp

Quantifier	Description
$n^+$	Matches any string that contains at least one $n$
$n^*$	Matches any string that contains zero or more occurrences of $n$
$n?$	Matches any string that contains zero or one occurrences of $n$
$^n$	Matches any string with $n$ at the beginning of it
$n\{X,\}$	Matches any string that contains a sequence of at least $X$ $n$ 's

```
<script>  
let text = "Hellooo World! ";  
let result = text.match(/o+/g);  
document.getElementById("demo").innerHTML = result;  
</script>
```

```
<script>  
let text = "Hellooo World! ";  
let result = text.match(/lo*/g);  
document.getElementById("demo").innerHTML = result;  
</script>
```

# What is an Event ?

- ▶ JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
  - ▶ When the page loads, it is called an event. When the user clicks a button, that click too is an event.
  - ▶ Other examples include events like pressing any key, closing a window, resizing a window, etc.
- 

## onclick Event Type

- ▶ This is the most frequently used event type which occurs when a user clicks the left button of his mouse.

```
<form>
```

```
<input type = "button" onclick = "sayHello()" value = "Say Hello" />
```

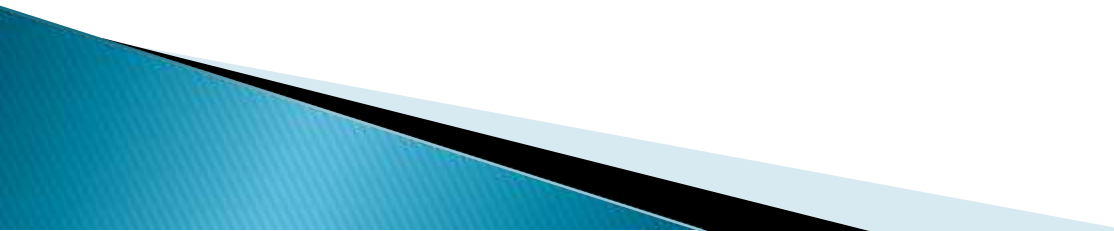
```
</form>
```

## onmouseover and onmouseout

- ▶ These two event types will help you create nice effects with images or even with text as well.
- ▶ The **onmouseover** event triggers when you bring your mouse over any element and the **onmouseout** triggers when you move your mouse out from that element.



```
<script language="Javascript" type="text/Javascript">  
  <!--  
    function mouseoverevent()  
    {  
      alert("This is JavaTpoint");  
    }  
  //-->  
</script>
```



## **onsubmit Event Type**

- ▶ **onsubmit** is an event that occurs when you try to submit a form. You can put your form validation against this event type.

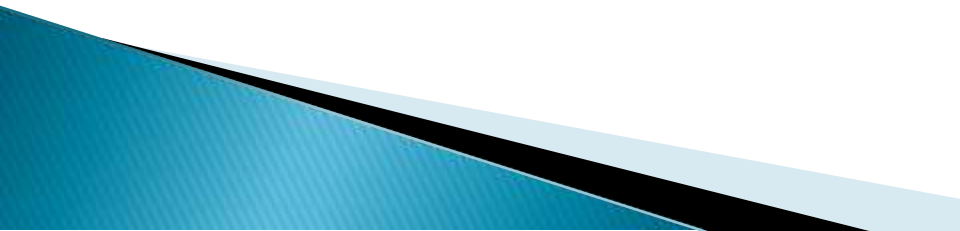
# JavaScript Callbacks

- ▶ A callback is a function passed as an argument to another function.
- ▶ Using a callback, you could call the calculator function (myCalculator) with a callback, and let the calculator function run the callback after the calculation is finished:

```
function myDisplayer(some) {  
    document.getElementById("demo").innerHTML = some;  
}
```

```
function myCalculator(num1, num2, myCallback) {  
    let sum = num1 + num2;  
    myCallback(sum);  
}
```

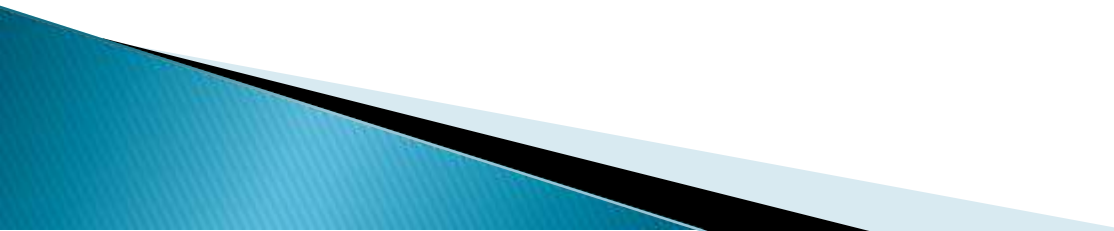
```
myCalculator(5, 5, myDisplayer);
```



# Function as a argument in JavaScript

```
function greet() {  
  return 'Hello';  
}  
// passing function greet() as a parameter  
function name(user, func)  
{  
  // accessing passed function  
  const message = func();  
  console.log(`${message} ${user}`);  
}  
name('John', greet);  
name('Jack', greet);  
name('Neil', greet);
```

# JSON

- ▶ JSON stands for **JavaScript Object Notation**
  - ▶ JSON is a lightweight data-interchange format
  - ▶ JSON is plain text written in JavaScript object notation
  - ▶ JSON is used to send data between computers
  - ▶ JSON is language independent
- 

- ▶ The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.
- ▶ Since the format is text only, JSON data can easily be sent between computers, and used by any programming language.
- ▶ JavaScript has a built in function for converting JSON strings into JavaScript objects:  
JSON.parse()
- ▶ JavaScript also has a built in function for converting an object into a JSON string:  
JSON.stringify()