Session: 15

# *Functions and Objects*

## Objectives

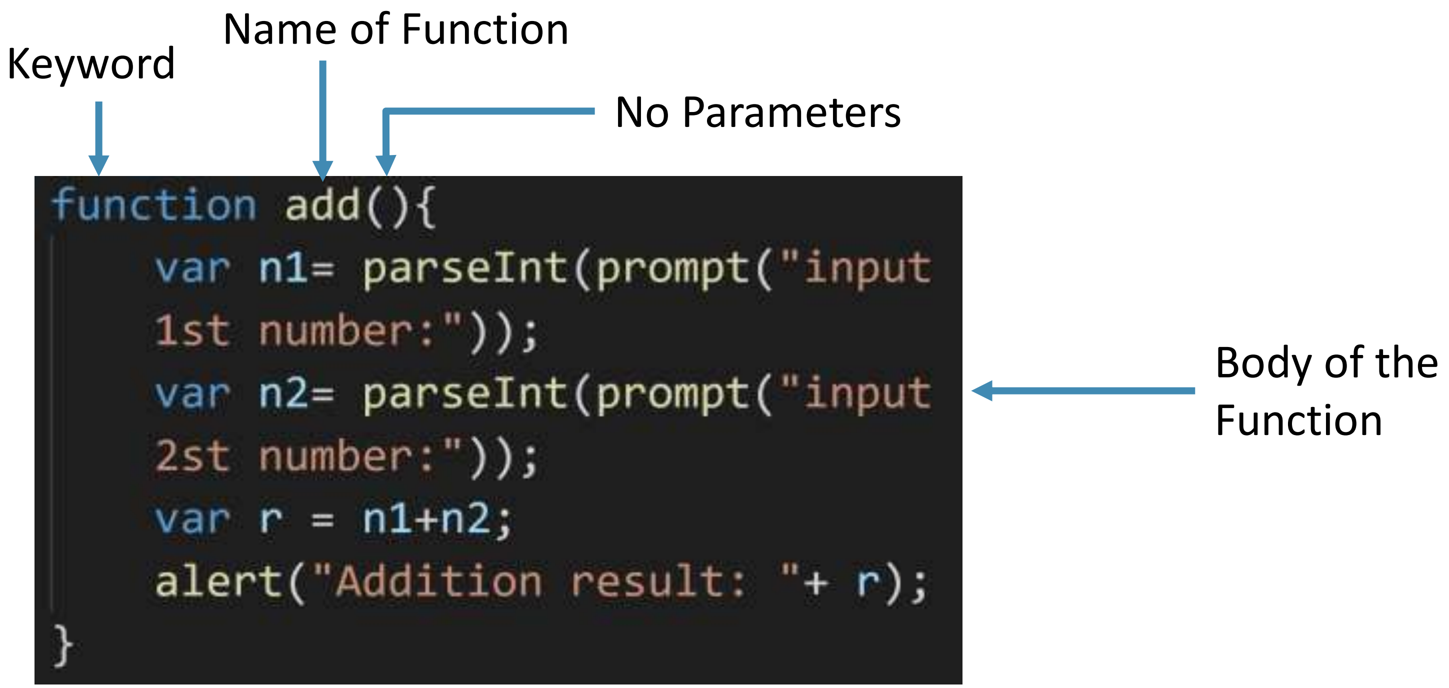
- Explain functions
- Explain parameterized functions
- Explain return statement
- Describe objects
- Explain different browser objects
- Describe Document Object Model (DOM)

## Functions

- Is an independent reusable block of code that performs certain operations on variables and expressions to fulfill a task.
- Might accept parameters, which are variables or values
- Might return the resultant value to display it in the browser after the operations have been performed.
- JavaScript function is always created under the script element.
- JavaScript supports both user-defined and built-in functions.

# Declaring and Defining Functions

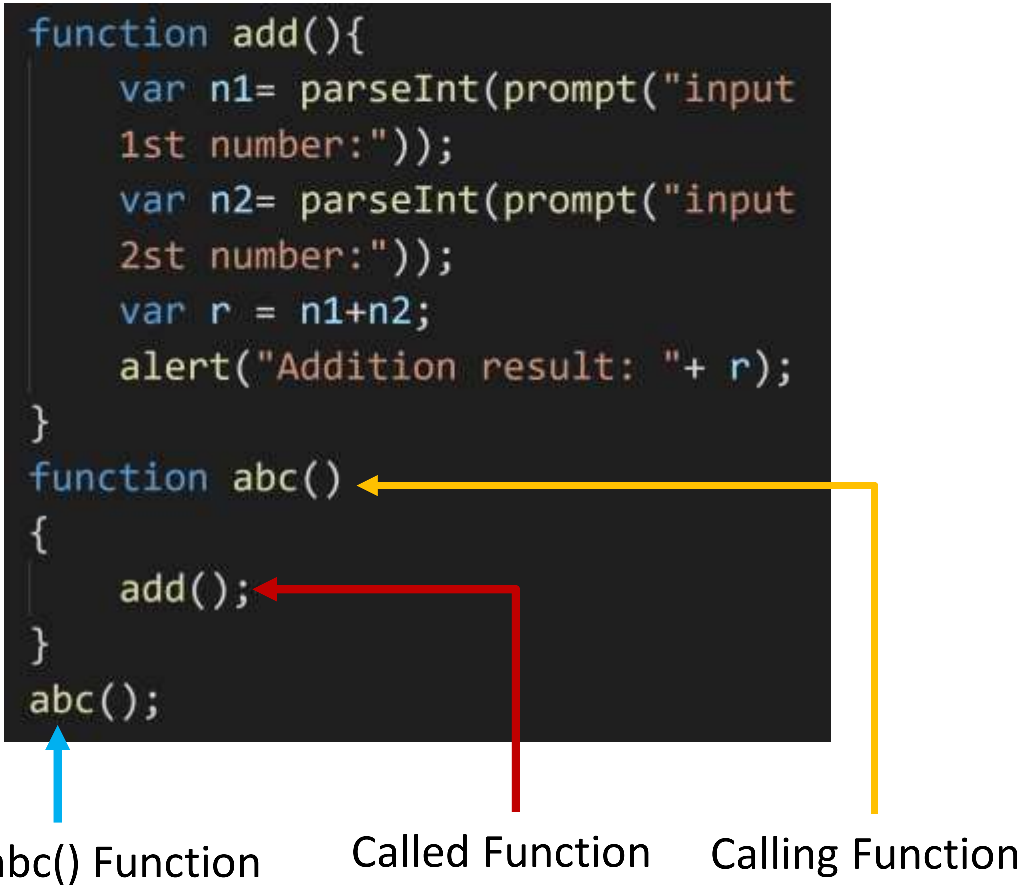
```
function function_name( list of parameter )
{
        // Body of the function
}
```



```
function add(){
    var n1= parseInt(prompt("input
    1st number:"));
    var n2= parseInt(prompt("input
    2st number:"));
    var r = n1+n2;
    alert("Addition result: "+ r);
}
```

# Invoking Functions

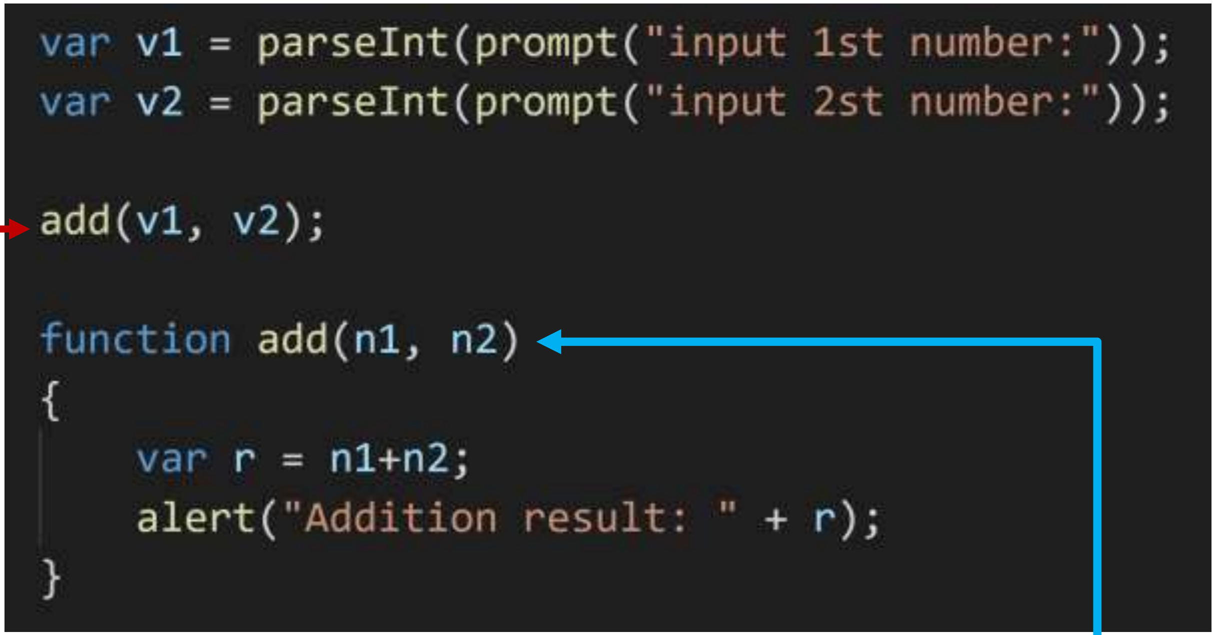
- A function need to be invoked / called to execute it in the browser.
- To invoke a function, specify the function name followed by parenthesis outside the function block.



```
function add(){
    var n1= parseInt(prompt("input
    1st number:"));
    var n2= parseInt(prompt("input
    2st number:"));
    var r = n1+n2;
    alert("Addition result: "+ r);
}
function abc()
{
    add();
}
abc();
```

## Parameterized Functions

- Are values on which the function needs to perform operations



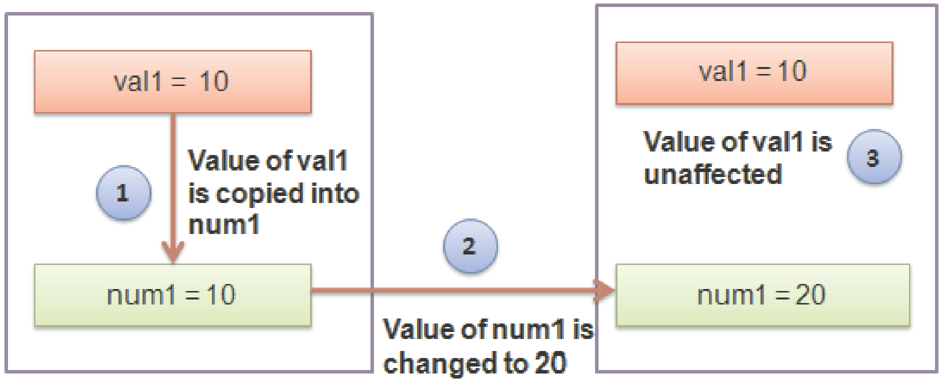
```
var v1 = parseInt(prompt("input 1st number:"));
var v2 = parseInt(prompt("input 2st number:"));

add(v1, v2);

function add(n1, n2)
{
    var r = n1+n2;
    alert("Addition result: " + r);
}
```

Invoke **add()** Function
by passing v1 and v2

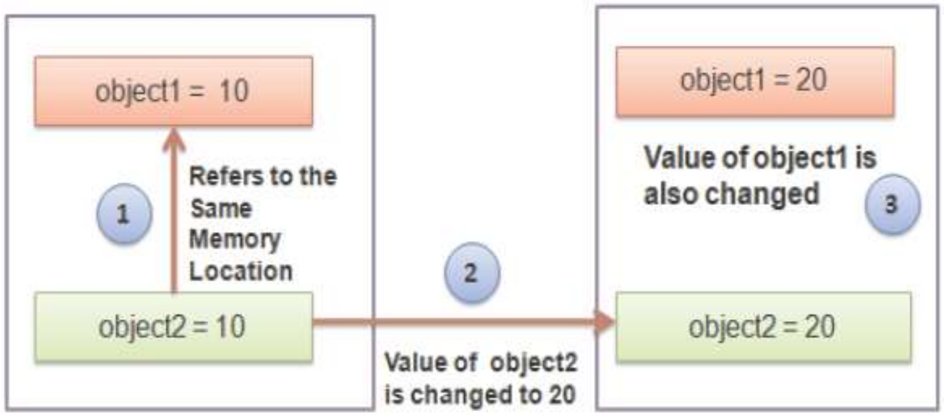
Holds value of **v1** and **v2**

# Ways of Passing Arguments

- Passing by value - Passing variables as arguments to a function. Called function do not change the values of the parameters passed to it.


val1 = 10
1
Value of val1 is copied into num1
num1 = 10
2
Value of num1 is changed to 20
val1 = 10
Value of val1 is unaffected
3
num1 = 20

- Passing by reference - Passing objects as arguments to a function.


object1 = 10
1
Refers to the Same Memory Location
object2 = 10
2
Value of object2 is changed to 20
object1 = 20
Value of object1 is also changed
3
object2 = 20

## return Statement

- Returns the control to the calling function.
- Allows sending the result back to the calling function - begins with return keyword followed by the variable or value.
- Can also be used to halt the execution of the function.

# Objects

- Are entities with properties and methods and resemble to real life objects.
- Properties specify the characteristics or attributes of an object.
- Methods identify the behavior of an object.

Object: Car

| | |
|---|---|
| **Properties** | Make - ford<br>Color - green<br>Wheels – four |
| **Methods** | run()<br>stop() |

Object: Bird

| | |
|---|---|
| **Properties** | Type - pigeon<br>Color - gray<br>Wings - two |
| **Methods** | eat()<br>fly() |

## Creating Custom Objects 1-2

- **Direct Method**

  var object_name = **new Object();**

  where,

  **object_name**: the name of the object.

  **new**: keyword that allocates memory to the object. This is known as instantiation of an object.

  **Object**: built-in JavaScript object that allows creating custom objects.

- **Template Method**

  var object_name **= new object_type(**list of arguments**);**

```
<script>
   var doctor = new Object();     // create an object by direct method
   var st = new Student('James', 23);   //create an object by template method
</script>
```

## Creating Custom Objects 2-2

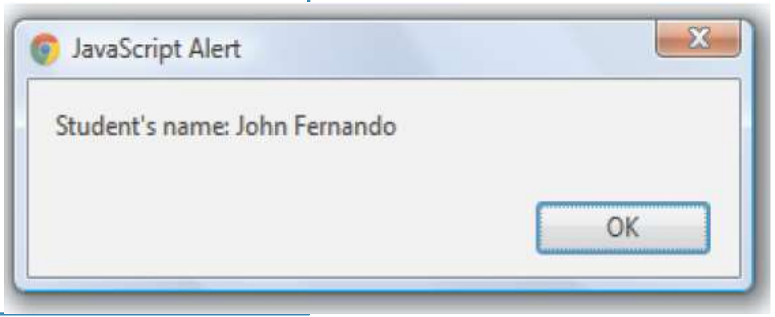
- A constructor function is a reusable block that specifies the type of object, its properties, and its methods.
- It might or might not take any parameters.
- When an object is initialized by new keyword: memory will be allocated to it and constructor function will be invoked.
- Syntax to create a constructor function is as follows:

```
function object_type(list of parameters)
{
        // Body specifying properties and methods
}
```

# Creating properties for custom objects 1-2

- Properties specify the characteristics of an object.
- To access a property of an object, specify the object name followed by a period and the property name.

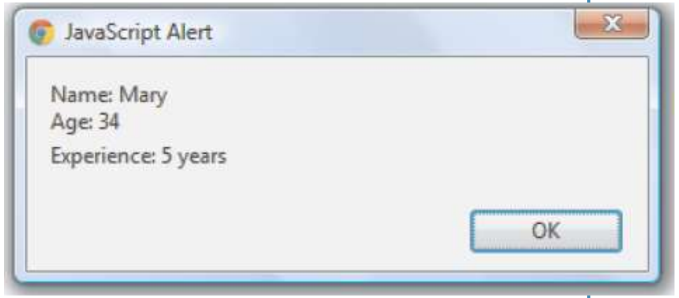
```
<script>
  var student = new Object();
  student.first_name = 'John';
  student.last_name = 'Fernando';
  student.age = '15';
  alert('Student\'s name: '
  + student.first_name
  + ' ' +student.last_name);
</script>
```


JavaScript Alert

Student's name: John Fernando

# Creating Properties for Custom Objects 2-2

```
<script>
  // To define the object type
  function Employee(name, age, experience)
  {
     this.name = name;
     this.age= age;
     this.experience= experience;
  }

  // Creates an object using new keyword
  var emp = new Employee('Mary', '34', '5 years');

  alert(" Name: " + emp.name + "\n Age: " + emp.age +
"\n Experience: " + emp.experience);

</script>
```



```
Name: Mary
Age: 34
Experience: 5 years
```

## Creating Methods for Custom Objects

- Methods are similar to JavaScript functions.
- A method is associated with an object and is executed by referring to that object but a function is executed independently.

```
<script>
   var square = new Object();
   square.length = 5;
   square.cal_area = function() {
     return square.length * square.length;
   }
   alert("Area: " + square.cal_area());
</script>
```

# Built-in Objects

- Object model of JavaScript forms the foundation of the language.
- These objects help to provide custom functionalities in the script.
- JavaScript treats the primitive data types as objects and provide equivalent object for each of them.
- JavaScript objects are categorized as built-in, browser, and HTML objects.
- Built-in objects are static objects which can be used to extend the functionality in the script.
- Browser objects, such as window, history, and navigator are used to work with the browser window.
- HTML objects, such as form, anchor, and so on are used to access elements on the Web page.

## String Object 1-2

- is a set of characters that are surrounded by single or double quotes.
- allows you to perform different text operations on them.
- is instantiated with the new keyword
- Syntax:
  **var object_name = new String(“Set of characters”) ;**

- Following table lists properties of the String object.

| Property | Description |
|---|---|
| length | Retrieves the number of characters in a string. |
| prototype | Adds user-defined properties and methods to the String instance. |

# String Object 1-2

| Method | Description |
|---|---|
| **charAt()** | Retrieves a character from a particular position within a string. |
| **concat()** | Merges characters from one string with the characters from another string and retrieves a single new string. |
| **indexOf()** | Retrieves the position at which the specified string value first occurred in the string. |
| **lastIndexOf()** | Retrieves the position at which the specified string value last occurred in the string. |
| **replace()** | Matches a regular expression with the string and replaces it with a new string. |
| **search()** | Searches for a match where the string is in the same format as specified by a regular expression. |
| **split()** | Divides the string into substrings and defines an array of these substrings. |
| **substring()** | Retrieves a part of a string between the specified positions of a string. |
| **toLowerCase()** | Specifies the lowercase display of the string. |

# Math Object

- allows to perform mathematical operations on numeric values.
- provides static properties and methods to perform mathematical operations.
- Properties and methods are declared as static, thus they can be invoked directly with the object name.
  - Syntax to access the properties of the Math object :

    ```
    var variable = Math.PropertyName;
    ```

    Example: var pi = Math.PI;

  - Syntax to invoke the methods of the Math object :

    ```
    var variable = Math.MethodName(optional parameters);
    ```

    Example: var x = Math.sqrt(26);

## Date Object

- Allows to define and manipulate the date time values.
- Syntax to instantiate the Date object :

```
var  object_name = new Date();
var  object_name = new Date(milliseconds);
var  object_name = new Date(year,month,day,hour,minutes, seconds, milliseconds);
var object_name = new Date("dateString");
```

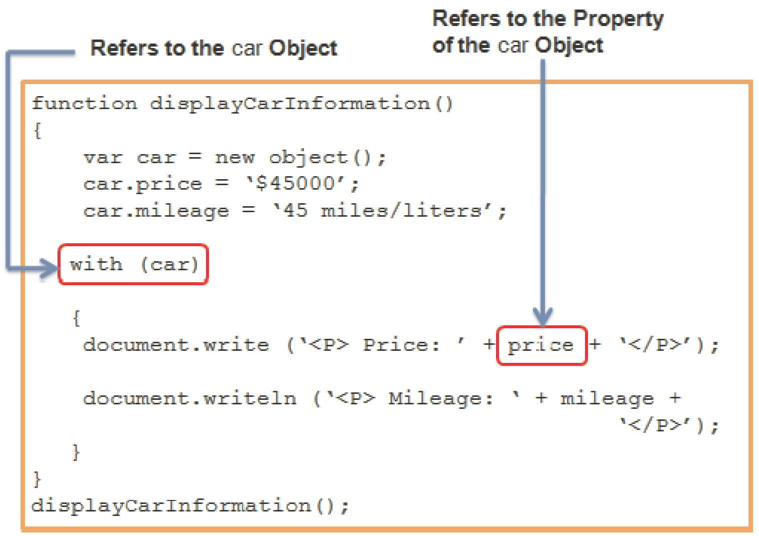
- Methods: getDate(), getDay(), getTime(), getFullYear()

```
function display() {
        var today = new Date() ;
        var dd = today.getDate() ;
        var mm = today.getMonth() + 1;
        var yy = today.getFullYear() ;
        alert("Today is " + dd + "/" + mm + "/" + yy);
}
display() ;
```

# with Statement

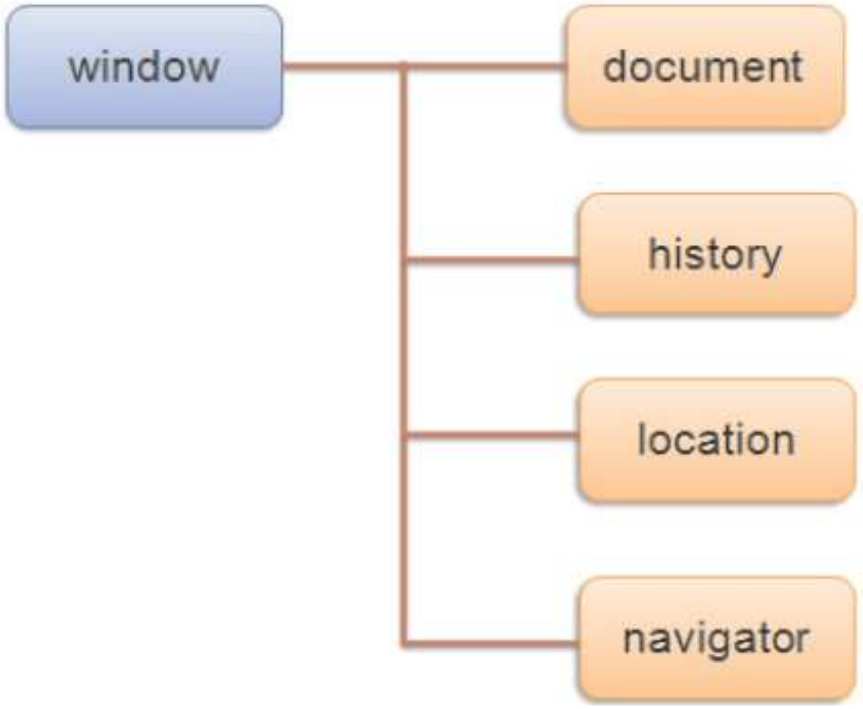
- allows to remove the object reference for each JavaScript statement.
- starts with the **with** keyword followed by the open and close brackets, which holds the statements that refer to object.
- increases the readability of the code and also reduces time required in writing each object reference in every related statement.
- Syntax :

```
with(object_name)
{
        // statements
}
```



```
function displayCarInformation()
{
    var car = new object();
    car.price = '$45000';
    car.mileage = '45 miles/liters';

    with (car)

   {
    document.write ('<P> Price: ' + price + '</P>');

    document.writeln ('<P> Mileage: ' + mileage +
                                          '</P>');
   }
}
displayCarInformation();
```

## Browser Objects



# window Object

- Represents a browser window & contains browser information
- Provides properties that allows setting a default text for status bar, name of browser, and so on
- Is the top level object in the JS hierarchy
- All the objects in the hierarchy are descendants of window object

| Property |
|---|
| defaultStatus |
| document |
| history |
| location |

| Method |
|---|
| alert() |
| confirm() |
| createPopup() |
| focus() |
| open() |
| prompt() |

## history Object

- Is a part of the window object
- Is an array containing a set of URLs visited by a user in browser window
- Allow referring to particular URL by specifying its index number in the array
- Enable to determine the number of URLs in history list by using length property.

| Method |
|---|
| back() |
| forward() |
| go() |

## history Object

```
<body>
    <button onclick=“ goBack() “ > Go Back </button>
    <button onclick=“ goForward() “ > Go Forward </button>
    <button onclick=“ go2Back() “ > Go 2 Pages Back </button>
</body>

<script>
    document.write("Number of URLs in history list: " + history.length);
    function goBack() {
        history.back();
    }
    function go2Back() {
        history.go(-2);
    }
    function go2Forward() {
        history.forward();
    }
</script>
```

## navigator Object

- Contains information about the browser used by the client
- Allows to retrieve information, such as name, version number …
- Following table lists the properties of the navigator object

| Property | Description |
|---|---|
| appName | Retrieves the name of the browser. |
| appVersion | Retrieves the version number and platform of the browser. |
| browserLanguage | Retrieves the language of the browser. |
| cookieEnabled | Determines whether the cookies are enabled in the browser. |
| platform | Retrieves the machine type such as Win32, of the client browser. |

# location Object

- Allows to access complete information of the URL loaded in browser window

| Property/Method | Description |
|---|---|
| **host** | Retrieves hostname and port number of the URL. |
| **href** | Specifies or retrieves the entire URL. |
| **pathname** | Specifies or retrieves the path name of the URL. |
| **assign()** | Loads a new document with the specified URL. |
| **reload()** | Reloads the current document by again sending the request to the server. |
| **replace()** | Overwrites the URL history for the current document with the new document. |

## location Object

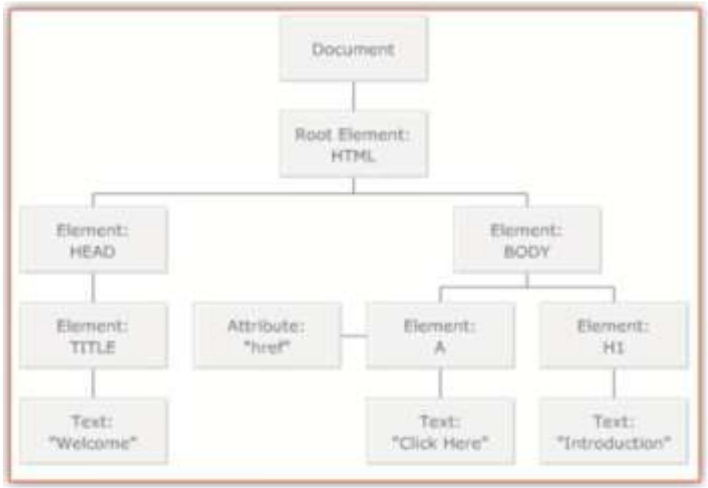
```
<body>
    <div id="note"> </div>
    <button onclick="“F1()"> Get entire URL of current page </button>
    < button onclick="“F2()"> Get protocol of currentmURL </button>
    < button onclick="“F3()"> Get path of current URL </button>
    < button onclick="location.reload()"> Reload Page  </button>
</body>
```

```
<script>
var note = document.getElementById("note");

function F1() {
    note.innerHTML= location.href ;
}
```

```
function F2() {
    note.innerHTML= location.protocol ;
}

function F3() {
    note.innerHTML= location.pathname ;
}
</script>
```

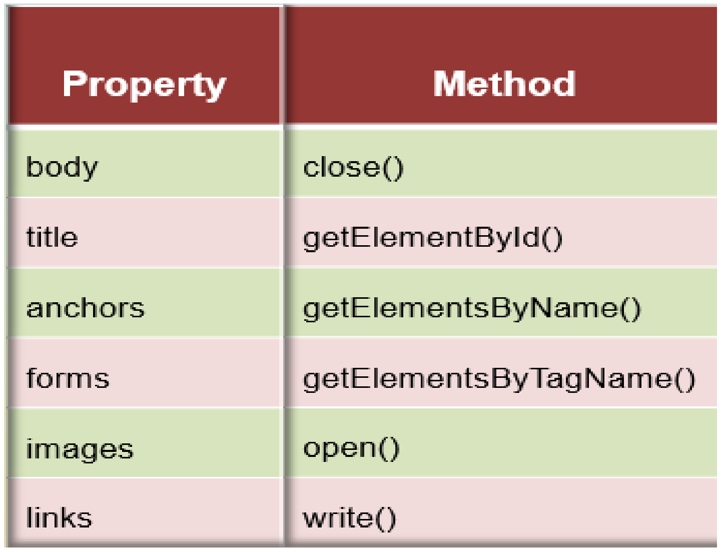
# Document Object Model 1-2

- JS allows the user to access HTML elements and also change the existing structure of an HTML page by using Document Object Model (DOM) specification.
- DOM is an Application Programming Interface (API) that defines the object structure for accessing and manipulating HTML elements.
- DOM is used with JS to add, modify, or delete elements and contents on the Web page.



## Document Object Model 2-2

- All the nodes contain properties that provide information about the node.
- The note properties are as follows:
  - **nodeName** - name of the node. It contains the tag name of the HTML element in upper case.
  - **nodeValue** - text contained within the node. This property is only available for attribute nodes and not for document and element nodes.
  - **nodeType** - type of the node. For example, the document node, element node...
- HTML DOM provides standard objects for HTML documents and some of these are as follows:
  - Document object
  - Form object
  - Link object
  - Table object

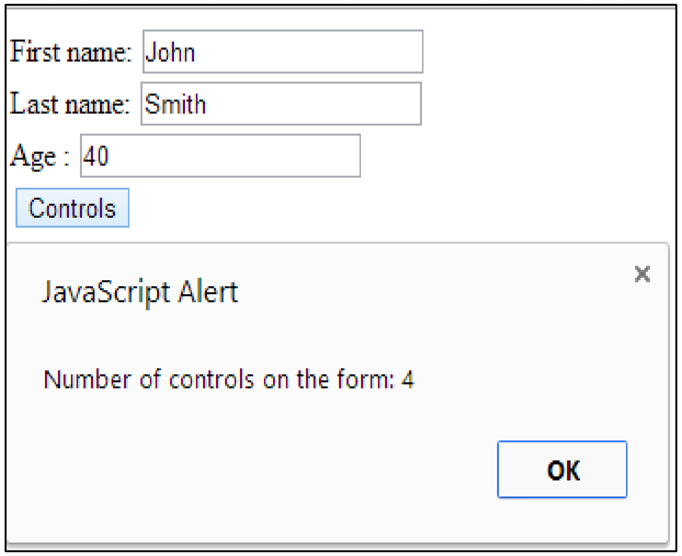


| Property | Method |
|---|---|
| body | close() |
| title | getElementById() |
| anchors | getElementsByName() |
| forms | getElementsByTagName() |
| images | open() |
| links | write() |

# Form Object

- Accepts input from the user and sends the user data for validation.
- A single HTML document can contain multiple forms.
- DOM specification provides a form object that represents an HTML form which is created for each <form> tag.

```
<head>
  <title> Form Object </title>
  <script>
    function display_length() {
      var count =document.getElementById("form1").length;
      alert("Number of controls on the form: " + count);
    }
  </script>
</head>
<body>
  <form id="form1" >
    First name: <input type="text" name="firstname" value="John" /><br />
    Last name: <input type="text" name="lastname" value="Smith" /><br />
    Age : <input type="text" name="age" value="40" /><br />
    <input type="button" value = "Controls" onClick="display_length()"/>
  </form>
</body>
</html>
```



First name: John
Last name: Smith
Age : 40

JavaScript Alert

Number of controls on the form: 4

## Summary

- A function is reusable piece of code, which performs calculations on parameters and other variables.
- The return statement passes the resultant output to the calling function after the execution of the called function.
- Objects are entities with properties and methods and resemble to real life objects.
- There are two ways to create a custom object namely, by directly instantiating the Object object or by creating a constructor function.
- JavaScript provides various built-in objects, such as String, Math, and Date.
- JavaScript also provides browser objects, such as window, history, location, and navigator.
- DOM is a standard technique for dynamically accessing and manipulating HTML elements. The DOM provides a document object which is used within the JavaScript to access all HTML elements presented on the page.