JavaScript

- JavaScript is a client side programming language for web.
- **Client Side Programming-** It is the program that runs on the client machine (browser) and deals with the user interface/display and any other processing that can happen on client machine like reading/writing cookies.
- JavaScript was initially created to "make web pages alive".
- Now, JavaScript can execute not only in the browser but also on the server, or on any device that has a special program called **JavaScript Engine.**

- **JavaScript in browser: -**
  - Add new HTML to the page, change the existing content, and modify styles.
  - React to user actions, run on mouse click, pointer movements, and key presses.
  - Send request over the network to remote servers, download and upload files (so called AJAX & COMET Technologies).
  - Get and set cookies, ask questions to the visitor, show messages.
  - Remember the data on the client-side (local storage).

- **JavaScript on Server: -**
  - JavaScript can be used at server side coding.
  - We can run JavaScript outside the browser, with the help of Node.js.
  - **Node.js** is an open-source, cross platform, backend, JavaScript runtime environment that executed JavaScript code outside a web browser.

- **ECMAScript: -**
  - ECMAScript is a standard.
  - Languages such as ActionScript, JavaScript, and JScript all use ECMAScript as it core.

- **JavaScript Engine: -**
  - It's a computer program that executes JavaScript code.

- **Run JavaScript using Node Js: -**
  - Install Node.js from https://nodejs.org/en
  - Open file path in cmd. Then provide **node <FileName>** -> Enter
    Ex- node second.js

- **JavaScript Variables: -**
  - JavaScript variables are containers for storing data values.
  - A **variable** is "named storage" for data.

1. **let-** Values can be changed. **Scope-** Block
2. **const-** Values can't be changed.
3. **var-** Values can be changed but it was being used in old JavaScript. We can re-declare a variable using **var** keyword. **Scope-** Global or function

- **DataType: -** It is a type of data that a variable can hold.

1. **Primitive DataType-** It's stored in stack.
   - number = 50, 75, 25.25 etc.
   - string = 'John', "Hello! Welcome" etc.

- boolean = true, false
- null = null
- undefined = undefined
- symbol

2. **Reference DataType-** It's stored in heap.
   - Array
   - Object
   - Date
   - Function

❖ **Difference between "==" & "===" operators: -**

- **==** -> Compares values
- **===** -> Compares values and type of data.

Ex-

```
var b=1;
var c='1';
console.log(b==c);    //true
console.log(b===c);  //false
```

❖ **What is NaN:-** Not a number- It represents a value which is not a legal number.
Function- isNaN()

❖ **Reference Comparison: -**

```
var a= [2];
var b= [2];
console.log(a==b); //false
```

❖ **Scope & Scope Check: -** Scope defined the accessibility of a variable. There are 3 types of scope in JS-

1. **Global Scope-** Variables can be accessed anywhere in the code.
2. **Local or Function Scope-** Variables can be accessed inside the function.
   Ex-
   ```
   function myFun(){
            var a=5;
   }
   ```
3. **Block Scope-** Variables can be accessed within the specified block. Related to **let & const** variables.
   Ex-
   ```
   If(){
       let a= 10;
   }
   ```

❖ **<u>String in JavaScript</u>: -**

- The **String** object is used to represent and manipulate a sequence of characters.
- Creating String-
  - Using Literal
    Ex- let message = 'Hello'; //type String
  - Using String() constructor
    Ex- let message = new String('Hello'); //type- object

- Use `` for multiline string.

- **String Interpolation: -**

```
let a=12;
let b=37;
let c=a+b;
console.log(`Sum of ${a} and ${b} is ${c}.`);
```

- **String Functions: -**

```
let s1="I love JavaScript";
console.log(s1); //I love JavaScript
```

1. **String length**

```
let len= s1.length;
console.log("String length: ", len); //String length:  17
```

2. **String concatenation**

```
let s2="and Java";
let s3= s1.concat(" ",s2);
console.log(s3); //I love JavaScript and Java
```

3. **Character at specific index**

```
let ch = s1.charAt(3);
console.log(ch); //o
```

4. **Index of specific character- returns 1st occurrence of char**

```
let ind = s1.indexOf('a');
console.log(ind); //8
```

5. **Last index of a character- starts from end**

```
let ind2 = s1.lastIndexOf("a");
console.log(ind2); //10
```

6. **UpperCase**

```
console.log(s1.toUpperCase()); //I LOVE JAVASCRIPT
```

7. **Lowercases**

```
console.log(s1.toLowerCase()); //i love javascript
```

8. **Substring from String: substring(start, end) -> end index is excluded**

```
let subString= s1.substring(0, 5);
console.log(subString); //I lov
```

9. **startsWith**

```
console.log(s1.startsWith('I lo'));  //true
```

10. endsWith

```
console.log(s1.endsWith('java')); //false
```

11. Split from particular position

```
let sp= s1.split(" ");
console.log(typeof sp); //object
console.log(sp); //['I', 'love', 'JavaScript']
```

12. Trim: Removes leading & trailing spaces

```
let s4 = "     Content with extra spaces    ";
let s5 = s4.trim();
console.log(s5); //Content with extra spaces
```

❖ **<u>String Operations</u>: -**

1. To check equality of two strings

```
//String literal
let str1 = "Javascript";
let str2 = "Javascript";
//== is case sensitive
console.log(str1 == str2); //true
console.log(str1 === str2); //true

//String object
let str3 = new String("Java");
let str4 = new String("Java");
console.log(str3 == str4); //false
console.log(str3 === str4); //false
```

2. Comparing two strings

```
//ASCII: A-65 Z-90, a-97 z-122
let s1 = "A";
let s2 = "B";
let result = s1.localeCompare(s2);
//s1=s2: 0, s1<s2: -ve, s1>s2: +ve
console.log(result); //-1
```

3. To fetch ASCII value of a Character

```
console.log(s2.charCodeAt(0)); //66
```

4. Replace a specific word with another

```
let oldString = "I hate JavaScript";
let newString = oldString.replace("hate", "love");
console.log(newString); //I love JavaScript
```

5. Accessing character- string[0]

```
console.log(oldString[0]); //I
```

6. **Coverting String to Array using split() function**

```
//Split from specific position
let oldStringArray = oldString.split(" ");
console.log(oldStringArray); //['I', 'hate', 'JavaScript']

//Split each character into array
let newStringArray = newString.split("");
console.log(newStringArray); //['I', ' ', 'l', 'o', 'v', 'e', ' ', 'J', 'a', 'v', 'a', 'S', 'c', 'r', 'i', 'p', 't']
```

❖ **<u>Type Conversion in JS</u>: -**

1. Implicit(Automatic Conversion)
2. Explicit(Manual Conversion)

- **String Conversion-**

```
let a = 12;
console.log(typeof a); //number

let as = String(a);
console.log(typeof as); //String
console.log(as); //12- String

console.log(a.toString()); //12-String
```

- **Numeric Conversion-**

```
let value = "1221";
let f1 = "23.51";
let value1 = Number(value);
console.log(typeof value1); //number
console.log(value + 5); //12215
console.log(value1 + 5); //1226

let pInt = parseInt(value);
console.log(typeof pInt); //number

let f2 = Number(f1);
console.log(typeof f2); //number

let pFloat = parseFloat(f1);
console.log(typeof pFloat); //number

console.log(Number(undefined)); //NaN
console.log(Number(null)); //0
console.log(Number(true)); //1
console.log(Number(false)); //0
```

- **Boolean Conversion-**

```
//Boolean Conversion
let ex="false";
console.log(ex); //false
console.log(typeof ex); //String

let exb= Boolean(ex);
console.log(exb); //true
//If String contains 1 or more characters then it will become "true" on conversion
console.log(typeof exb);  //boolean
```

❖ **<u>Array</u>: -**

- **Creating Array**

```
//Method:1
let arr = [12, 56, 11, 8, 90];
console.log(arr); //[12, 56, 11, 8, 90]
console.log(typeof arr); //object

//Method:2
let student = new Array("Vishal", "Durgesh", "Sonu");
console.log(student); //['Vishal', 'Durgesh', 'Sonu']

//Accessing Array elements
console.log(arr[0]); //12
console.log(arr[6]); //undefined
```

- **A JS array can contain elements of different datatypes**.

Ex-

```
const myDetails = ["Vishal", 9935495382, 25];
console.log(myDetails); //['Vishal', 9935495382, 25]

//Array inside array
let emp = ["Name", 27, ["Pizza", "Cold-drink"]];
console.log(emp[2][1]); //Cold-drink
```

- **Array size is not fixed**.

Ex-

```
student.push("Shivam");
console.log(student); //['Vishal', 'Durgesh', 'Sonu', 'Shivam']
console.log("Size of Student array: ", student.length); //4
```

❖ **<u>Array Operations</u>: -**

```
let names = ["John", "Peter", "Ram"];
console.log(names); //['John', 'Peter', 'Ram']
```

- **Loop over an array**

```
names.forEach(function(item, index){
  console.log(index, " => ", item);
  /*
     0 ' => ' 'John'
     1 ' => ' 'Peter'
     2 ' => ' 'Ram'
  */
})
```

- **Add an element at last**

```
names.push("Shyam");
console.log(names); //['John', 'Peter', 'Ram', 'Shyam']
```

- **Add element at begining**

```
names.unshift("Karan");
console.log(names); //['Karan', 'John', 'Peter', 'Ram', 'Shyam']
```

- **Remove last element**

```
let r = names.pop();
console.log(r); //Shyam
console.log(names); //['Karan', 'John', 'Peter', 'Ram']
```

- **Remove element from begining**

```
let r1= names.shift();
console.log(r1); //Karan
console.log(names); //['John', 'Peter', 'Ram']
```

- **Get index of element**

```
let i = names.indexOf("John");
console.log(i); //0
```

- **Remove element from specific position**

```
names.splice(1, 1); //arg1- Position, arg2- no. of elements to be removed
console.log(names); //['John', 'Ram']
```

- **Copy an array**

```
let copy = names.slice();
console.log(copy); //['John', 'Ram']
```

- **Sum of an array elements**

```
let num = [12,54, 29];
let sum = 0;
num.forEach(function(item, index){
  sum = sum+item;
})
console.log("Sum: ", sum); //Sum: 95
```

❖ **Functions in JavaScript: -** Functions are the block of code designed to perform specific task.
Syntax- function function_name(args if any){
----code----
}

- **Creating Function: -**

- **Method:1- Traditional Approach**

```
function sayHello() {
  console.log("Hello!"); //Hello!
  console.log("How are you?"); //How are you?
}

sayHello(); //function call

//Function with args & return type
function sum(a, b) {   //a,b: Formal parameter
  let c = a + b;
  return c;
}

let s = sum(12, 43); //12,43: Actual Parameter
console.log("Sum: ", s); //Sum:  55
```

- **Methhod:2- Arrow Function**

```
const toCelsius = (fahrenheit) => {
  let temp = (5 / 9) * (fahrenheit - 32);
  return temp;
};

let celsius = toCelsius(120);
console.log("Temp in celsius: ", celsius); //48.88
```

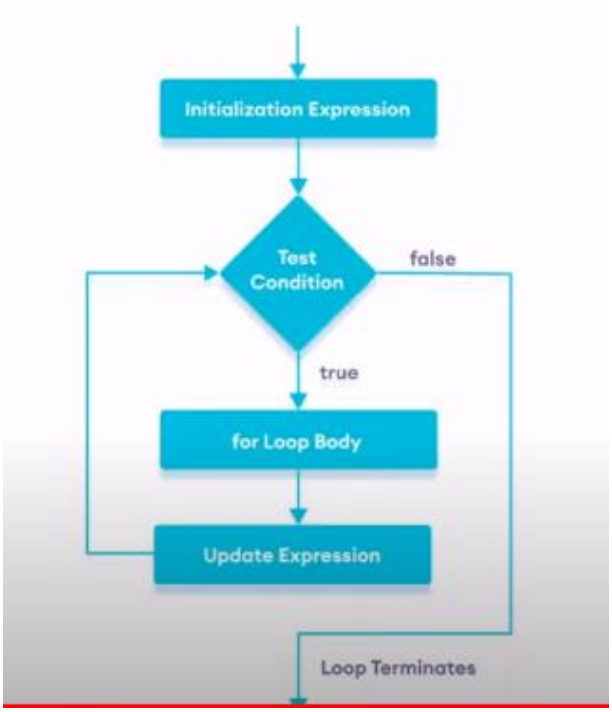
❖ **Truthy Value-** +ve/-ve numbers, **true**, String having 1 or more characters.
❖ **Falsy Value-** 0, **false**, Empty string(“”), undefined, null, NaN.

❖ **Loops in JavaScript: -**

- Loops are used to execute the same block of code again and again, as long as certain condition is met.
- The basic idea behind a loop is to automate the repetitive tasks within a program to save the time and effort.
  JavaScript now supports 5 different types of loops-
  1. for
  2. while
  3. do..while
  4. for..in -> object properties
  5. for..of -> string, array

1. **for loop-**
   - Repeat the statements until certain conditions are met.

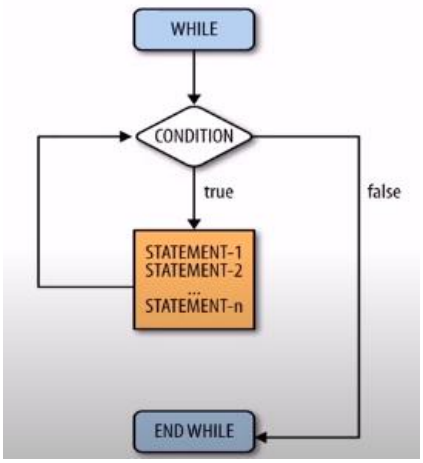
- **Syntax-**
  for(initialization; condition; increment/decrement){
  ----code block----
  }



- **Example-**

```
<script>
  for(let i=1; i<=10; i++){
    console.log(`Hey ${i}`);
  }
</script>
```

2. **while loop-**
   - Repeats the statement code until the specified condition evaluates to true.
   - **Syntax-**
     while(condition){
     --code block--
     }

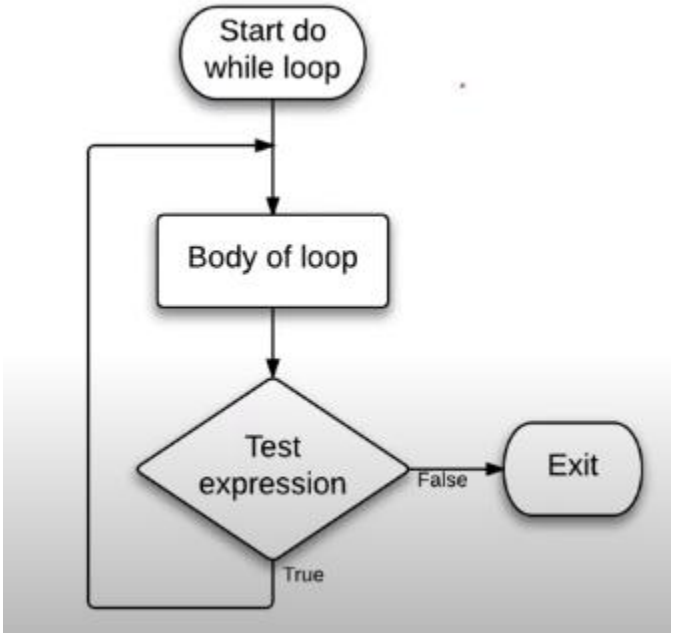


- **Example**-

```
let j=1;
while(j<=10){
   console.log(`${j} Name: Vishal Srivastava`);
   j++;
}
```

3. **do..while loop-**
   - Block of code repeats at least once before the condition check.
   - **Syntax**-

```
do{
    --code block--
}
while(condition);
```



   - **Example**-

```
let k=1;
do{
   console.log(`Value of k: ${k}`);
   k++;
}while(k<=5);
```

4. **for..in loop-**
   - The for-in loop is a special type of loop that iterates over the properties of an object or elements of an array.
   - **Syntax-**

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

   - **Example-**

```
let student = {
  name: "John",
```

```
 phone: 9921013530,
 address: "Amsterdam",
};
console.log(student);

for (let key in student) {
 console.log(`${key} => ${student[key]}`);
 /*
    name => John
    phone => 9921013530
    address => Amsterdam
    */
}

//Array
let arr = ["ABC", "DEF", "GHI"];
for (var p in arr) {
 console.log(arr[p]);
 /*
    ABC
    DEF
    GHI
    */
}
```

5. **for-of loop-**
   - ES-6 introduces a new for-of loop, which allows us to iterate over arrays or other iterable objects(ex-strings) very easily.
   - **Syntax-**
     for*(variable* ***of*** *iterableobject)*
     {
     //code to be executed
     }
   - **Example-**

```
let arr2 = ["ABC", "DEF", "GHI"];
console.log(arr2); //['ABC', 'DEF', 'GHI']

for(let item of arr2){
 console.log(item);
 /*
 ABC
 DEF
 GHI
 */
}
```

❖ **Objects in JavaScript: -**

- JavaScript is an object based language.
- A JavaScript object is a collection of key value pairs.
- **Syntax**- {key1:value1, key2:value2, …., etc.}
- **Example**-

```
{
 name: 'Vishal',
 phone: 992132,
 address: 'Ayodhya'
};
```

- Objects can also have functions inside them.

Ex-

```
{
 name: 'Vishal',
 phone: 992132,
 address: 'Ayodhya',
 display: function()
          {
           console.log(this.name);
          }
}
```

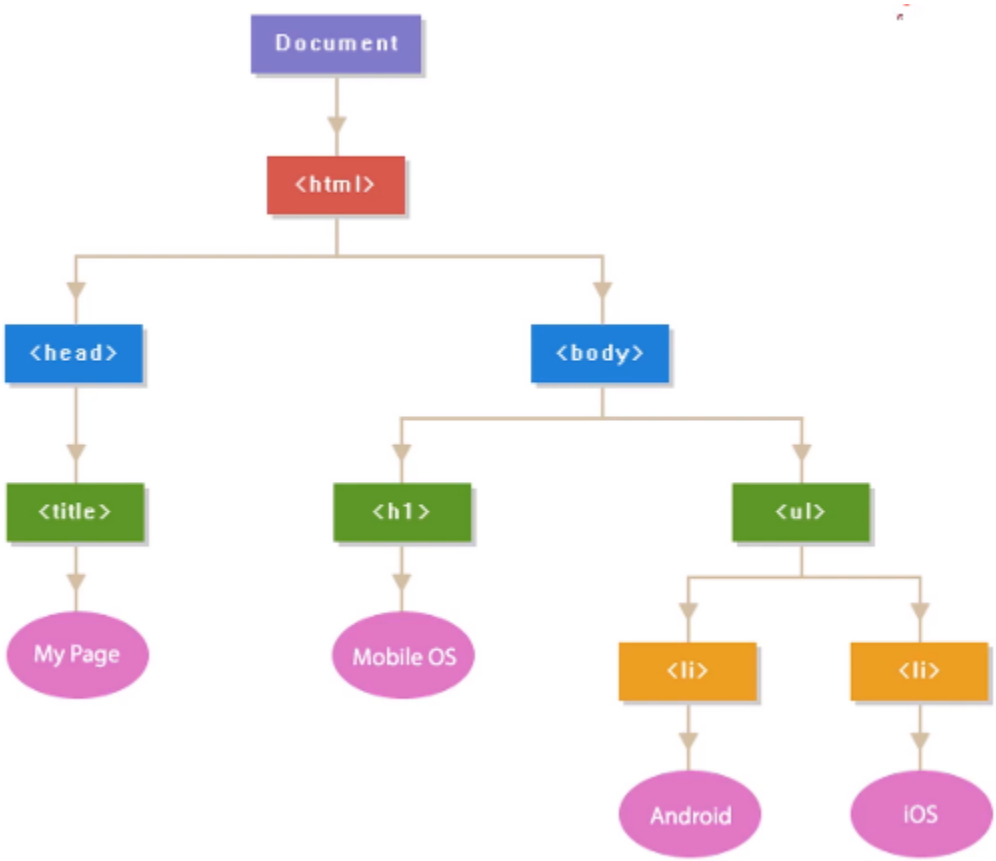
- Object properties can be accessed in two ways- **object.key** or **object[key]**
- **Example-**

```
let ob1 =
{
   "first name": "Hritik",
   "last name": "Roshan",
   age: 35,
   address: 'Mumbai',
   contact: 9876543210,
   display: function(){
      console.log("Displaying properties");
      console.log(this.contact);
   }
};
console.log(ob1);  //{first name: 'Hritik', last name: 'Roshan', age: 35, address: 'Mumbai', contact: 9876543210}

//Accessing the properties
ob1.display(); //Displaying properties 9876543210
console.log(ob1.age); //35

console.log(ob1["first name"]); //Hritik
console.log(ob1["last name"]); //Roshan
console.log(ob1["address"]); //Mumbai
```

```
//Changing values of property
ob1["first name"] = "John";
console.log(ob1["first name"]); //John

//Adding dynamic properties
ob1["email"] = "vishal123@gmail.com";
console.log(ob1); //{first name: 'John', last name: 'Roshan', age: 35, contact: 9876543210, email: 'vishal123@gmail.com'}

//Deleting property
delete ob1["last name"];
console.log(ob1); //{first name: 'John', age: 35, contact: 9876543210, email: 'vishal123@gmail.com'}
```

❖ **DOM- Document Object Model: -**

- It is a platform and language independent model to represent the HTML or XML documents.
- It defines the logical structure of the documents and the way in which they can be accessed and manipulated by an application program.
- In the DOM, all parts of the document, such as elements, attributes, text etc. are organized in a hierarchical tree like structure.
- In DOM terminology, these individual parts of the document are known as **nodes.**



❖ **DOM Selectors: -**

```
<body>
        <h1>DOM Selectors</h1>
        <h2 id="banner">Vishal Srivastava</h2>
        <h1 class="myClass">Hello! Good Morning</h1>
        <h1 class="myClass">Hello! Good Morning</h1>
        <h1 class="myClass">Hello! Good Morning</h1>

        <!-- JavaScript -->
        <script>
            console.log("Body Section");

            let ht = document.documentElement;
            let bo = document.body;
            let he = document.head;
            let ti = document.title;

            console.log(ht);
            console.log(bo);
            console.log(he);
            console.log(ti);

            //Select element by Id
            let h2 = document.getElementById("banner");
            console.log(h2); //<h2 id="banner">Vishal Srivastava</h2>
            h2.innerHTML = "Modified text";
            h2.style.color = "red";
            h2.style.backgroundColor = "yellow";

            //Select element by class name
            let el1 = document.getElementsByClassName("myClass");
            console.log(typeof el1); //object
            console.log(el1);

            for(let e of el1){
                console.log(e);
               // <h1 class="myClass">Hello! Good Morning</h1>
              // <h1 class="myClass">Hello! Good Morning</h1>
              // <h1 class="myClass">Hello! Good Morning</h1>
              e.style.color = "blue";
              e.style.backgroundColor = "yellow";
            }

            //Select element with the help of CSS selector
            //querySelector => single element
            let els1= document.querySelector('.myClass');
```

```
                console.log(els1); //<h1 class="myClass">Hello! Good Morning</h1>

                //querySelectorAll => list of elements
                let els2= document.querySelectorAll('.myClass');
                console.log(els2); //will return all the 3 matching nodes

                for(let el of els2){
                        console.log(el);
                        el.style.color = "green";
                }
  </script>
</body>
```

❖ **DOM Styling: -**

```
<body>
 <style>
   .red{
      background-color: red;
      color: white;
   }
 </style>
 <button onclick="changeStyle()">CHANGE STYLE</button>
 <button onclick="getStyle()">GET INFO</button>

 <h2 id="main">Hello, Vishal Srivastava</h2>
 <h2 id="sub1">Software Engineer</h2>
 <h2 id="sub2">Cognizant Technology Solutions</h2>

 <script>
  //select the element
  let mainOb = document.getElementById("main");

  //applying style on element
  const changeStyle = () => {
    mainOb.style.color = "blue";
    mainOb.style.fontSize = "40px";
    mainOb.style.border = "1px solid red";
    mainOb.style.padding = "20px";
    mainOb.style.background = "#e2e2e2";

    document.body.style.background = "black";
    document.body.style.color = "white";

    mainOb.className = mainOb.className + "red";
    //JS script styling will have preference over CSS styling
  };
```

```
  const getStyle = () => {
    //to get user defined properties
    console.log(mainOb.style.color); //blue
    console.log(mainOb.style.fontSize); //40px
    console.log(mainOb.style.border); //1px solid red

    //To get the property which is not set by user
    let style = window.getComputedStyle(mainOb);
    console.log(style.getPropertyValue("color")); //rgb(0, 0, 0)
    console.log(style.getPropertyValue("font")); //700 24px "Times New Roman"
  }
 </script>
</body>
```

❖ **DOM Set & Get Attributes: -**

- Get the attribute value-
  value = getAttribute(attribute_name);
- Set the attribute value-
  setAttribute(attribute_name, 'value');
- Example-

```
<body>
  <h1>DOM- Get & Set Attribute</h1>
  <a id="url" class="alert alert-danger"  href="http://www.google.com">Google</a>
  <button onclick="getAtt()">Get Attribute</button>
  <button onclick="setAtt()">Set Attribute</button>
  <script>
    const getAtt = () => {
      //Get the attribute
      let urlOb = document.getElementById("url");
      let h = urlOb.getAttribute("href");
      console.log(h); //http://www.google.com
      console.log(urlOb.getAttribute("id")); //url
      console.log(urlOb.getAttribute("class")); //alert alert-danger
    }

    const setAtt = () => {
      //Set the attribute
      let urlOb = document.getElementById("url");
      urlOb.setAttribute("href", "http://www.yahoo.com");
    }
  </script>
 </body>
```

❖ **DOM Manipulation: -**

```
<html>
  <body>
    <div id="main">
```

```
        <h1 id="title">Hello World!</h1>
        <p id="hint">This is a simple paragraph</p>
      </div>
      <button onclick="addNode()">Add Node</button>

      <script>
        function addNode(){
          // console.log("Adding node");
          // Creating node
          let divElement = document.createElement("div");

          //Create a text node
          let textElement1 = document.createTextNode("WOW! This is magic.")
          let textElement2 = document.createTextNode("This is dynamic text.")

          //adding text node to divElement
          divElement.appendChild(textElement1);
          divElement.appendChild(textElement2);
          console.log(divElement);

          let mainElement = document.getElementById("main");
          mainElement.appendChild(divElement);

          let titleElement = document.getElementById("title");
          // mainElement.insertBefore(divElement, titleElement);

          titleElement.innerHTML = "<h1>Hello Dear!</h1>";

          mainElement.removeChild(titleElement);
        }
      </script>
    </body>
  </html>
```

- **<u>DOM Manipulation</u>: -**

```
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>JavaScript Navigation</title>
   </head>
   <body>
    <div id="container">
     <h1 id="title">We are learning JavaScript</h1>
     <p id="content">
      <span>JavaScript is one of the most popular programming languages for
       frontend as well as backend</span>
     </p>
```

```
  <h2>Happy Learning!</h2>

 </div>
 <button onclick="change()">Click Here for Magic</button>

 <script>
  function change() {
    let containerOb = document.getElementById("container");

    console.log(containerOb.firstElementChild.nodeName);

    //first child
    let firstChild = containerOb.firstElementChild;
    console.log(firstChild); //<h1 id="title">Hi! Going to change!</h1>
    firstChild.innerHTML = "Hi! Going to change!";

    //last child
    let lastChild = containerOb.lastElementChild;
    console.log(lastChild); //<h2 style="background-color: red; color: white;">Happy Learning!</h2>
    lastChild.style.backgroundColor = "red";
    lastChild.style.color = "white";

    //all child nodes
    let nodes = containerOb.children;
    for(let n of nodes){
       console.log(n.nodeName); //h1, p, h2
    }

    //Parent nodes
    let parent = containerOb.parentNode;
    console.log(parent.nodeName); //body

    //previousSibling & nextSibling
    let titleOb = document.getElementById("title");

    let next = titleOb.nextElementSibling;
    console.log(next.nodeName); //p
  }
 </script>
</body>
```

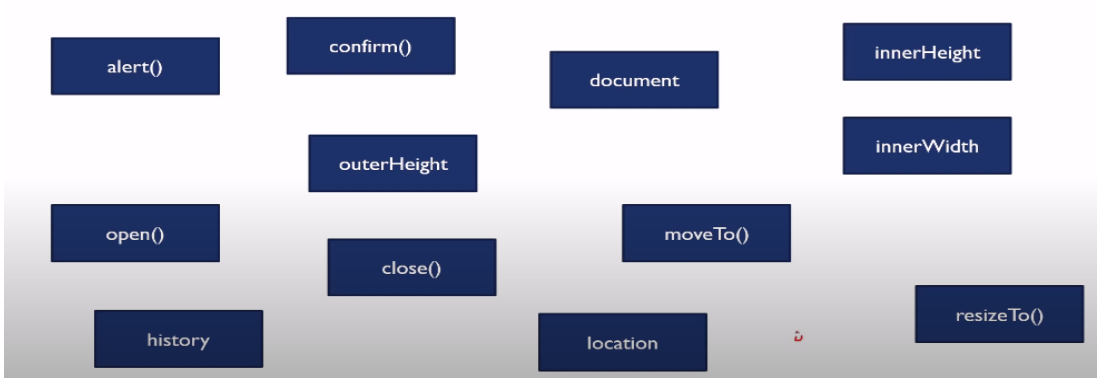
❖ **JavaScript Timer**: -

```
<body>
  <h1>JavaScript Timer</h1>
  <button onclick="start()">Start</button>
  <button onclick="repeat()">Repeat</button>
  <button onclick="stop()">Stop</button>
  <script>
    function myCode(){
      console.log("Have a good day!");
    }
    //setTimeout()- executes code after a delay
    function start(){
      console.log("Hello Vishal! How are you?");
      setTimeout(myCode, 5000);
    }


    //setInterval()- executes code at regular interval
    let timer;
    function repeat(){
      console.log("Welcome to INDIA");
      timer = setInterval(myCode, 2000);
    }

    function stop(){
      clearInterval(timer);
    }
  </script>
</body>
```

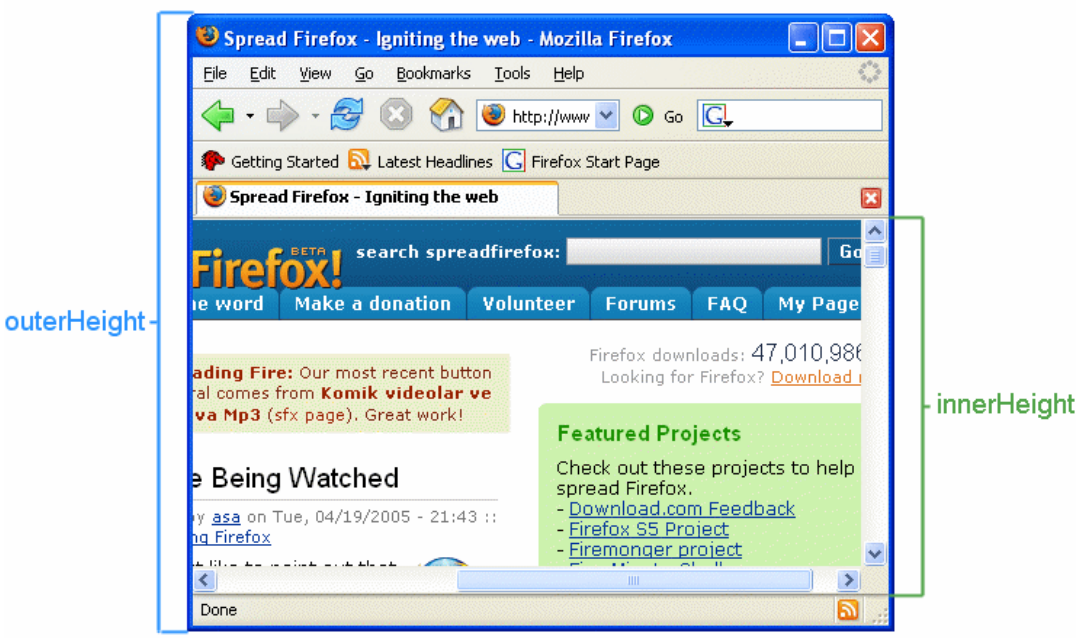
❖ **Window Object: -**

- It represents the browser's window. A window object is supported by all browsers.
- Global variables are properties of the window object.
- Global functions are method of the window object.
- Even the document object is a property of the window object.

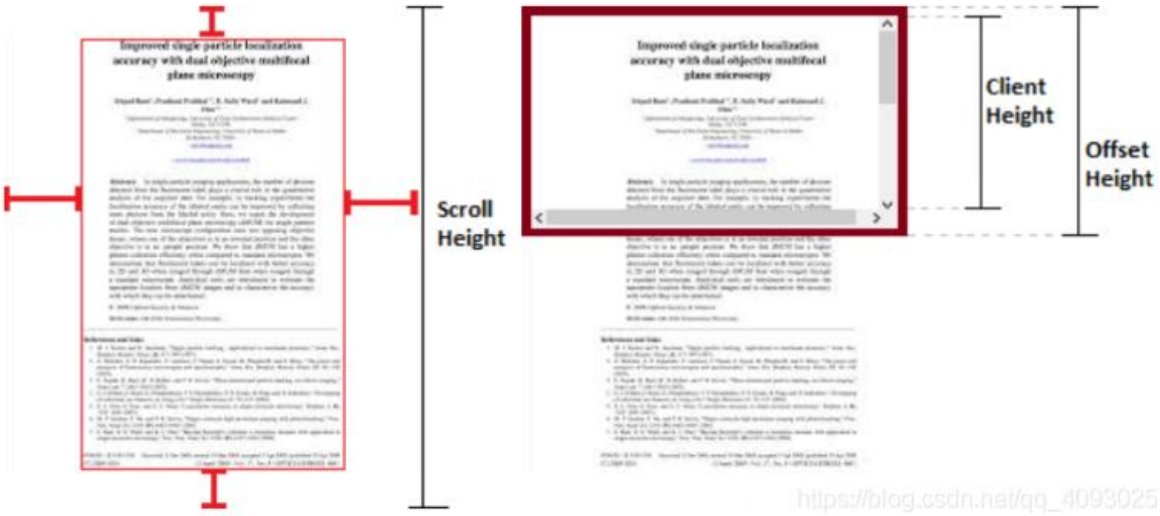
➢ **Important Properties and Methods-**



- **innerHeight-** Height of the tab in the browser window. Or It's the height of the content area.
- **outerHeight-** Height of the browser window.



- **scrollHeight-** Height of entire content & padding(visible or not).
  Height of all content + padding, despite of height of element
- **clientHeight-** Height of Visible content & padding.
- **offsetHeight-** Height of visible content & padding + border + scrollbar



❖ **<u>Location Object</u>: -** Location object can be used to get the current page address and to redirect the browser to a new page.

- **location.href** returns the url of the current page.
- **location.hostname** returns the domain name of the web host.
- **location.pathname** returns the path and file name of the current page.
- **location.protocol** returns the web protocol used(http: or https:).
- **location.assign** loads a new document.
- **location.port** return the number of the internet host port.

❖ **History Object: -** History object can contain browser history and use to perform operations with history.

- history.back()
- history.forward()

❖ **Date & Time: -**

- Build-in Date object allows us to get the local date time by accessing computer system clock through browser.
- Date object is created with the help of constructor. Ex-
  var d1 = new Date(); -> current date & time
  var d2 = new Date(2023,0,21,14,35,20,50);
  Year, m , d, hr,min, sec,ms
  var d3 = new Date("28 July 2023");
  var d4 = new Date(151735680000);
  ms
- **Important methods-**

```
var d = new Date();
alert(d.toDateString());// Display an abbreviated date string
alert(d.toLocaleDateString());// Display a localized date string
alert(d.toISOString()); // Display the ISO standardized date string
alert(d.toUTCString()); // Display a date string converted to UTC time
alert(d.toString()); // Display the full date string with local time zone

d.getDate()
d.getDay()
d.getMonth()
d.getFullYear()
```

- Ex-

```
//Current Date
var d1 = new Date();
console.log(d1); //Fri Jul 28 2023 17:09:50 GMT+0530 (India Standard Time)
console.log(d1.toLocaleDateString()); //7/28/2023
console.log(d1.toLocaleTimeString()); //5:09:50 PM
console.log(d1.toUTCString()); //Fri, 28 Jul 2023 11:39:50 GMT
console.log(`Date: ${d1.getDate()}`); //Date: 28
console.log(`Month: ${d1.getMonth() + 1}`); //Month: 7
console.log(`Year: ${d1.getFullYear()}`); //Year: 2023
console.log(`Day: ${d1.getDay() + 1}`); //Day: 6
console.log(`Hour: ${d1.getHours()}`); //Hour: 17
console.log(`Minute: ${d1.getMinutes()}`); //Minute: 14
console.log(`Second: ${d1.getSeconds()}`); //Second: 18

//Custom date
var d2 = new Date("22 January 2023");
console.log(d2); //Sun Jan 22 2023 00:00:00 GMT+0530 (India Standard Time)

var d3 = new Date(2023,0,22);
console.log(d3); //Sun Jan 22 2023 00:00:00 GMT+0530 (India Standard Time)
```

❖ **Math Object: -**

```
//Area of Circle
function area_of_circle(r) {
 let area = Math.PI * Math.pow(r, 2);
 return area;
}
console.log(`Area of circle with radius 5 is: ${area_of_circle(5)}`); //Area of circle with radius 5 is: 78.53981633974483

//Absolute value
let a = -123;
console.log(Math.abs(a)); //123

//Random number
let ran = Math.random();
let two_digit_random_number = ran * 100;
console.log(two_digit_random_number);

//discard decimal
console.log(Math.trunc(two_digit_random_number));

//Square root
console.log(Math.sqrt(25)); //5

//ceil- next higher integer
console.log(Math.ceil(3.23)); //4
console.log(Math.ceil(-3.23)); //-3

//floor-lowest integer
console.log(Math.floor(3.23)); //3

//min & max
console.log(Math.min(15, 17, 7, 11, 9)); //7
console.log(Math.max(15, 17, 7, 11, 9)); //17

//trigonometry
console.log(Math.sin(30 * (Math.PI / 180))); //0.49999999999999994
```

❖ **Event & Event Handling: -**

```
<body>
 <button onclick="my_fun()">Click Me</button>
 <button id="go_live_btn">Go Live</button>
 <input
  type="text"
  onfocus="change_color()"
  onkeyup="update_text()"
```

```
    id="field"
   />
   <h1 id="heading">This is dummy text.</h1>

   <script>
    const my_fun = () => {
     alert("This alert is from function");
    };

    const change_color = () => {
     document.getElementById("field").style.backgroundColor = "aqua";
    };

    const update_text = () => {
     let data = document.getElementById("field").value;
     document.getElementById("heading").innerHTML = data;
    };

    let go_live_btn_ob = document.getElementById("go_live_btn");

    go_live_btn_ob.addEventListener("click", () => {
     alert("You are live now.");
    });

    go_live_btn_ob.addEventListener("click", () => {
     document.body.style.backgroundColor = "black";
     document.body.style.color = "white";
    });

    go_live_btn_ob.addEventListener("mouseover", () => {
     go_live_btn_ob.style.backgroundColor = "red";
    });
   </script>
  </body>
```

- **<u>Web Storage</u>: -**
  - With web storage, websites store data locally on user's browser.
  - There're two types of storage- **localStorage & sessionStorage**
    1. **localStorage-** for multiple sessions with no expiration.
    2. **sessionStorage-** for single session(data is lost when browser is closed).
  - Both objects store key-value pair.
  - **Storing data-**
    localStorage.setItem(key, value);
    Ex- localStorage.setItem("name" , "Vishal");
  - **Getting data-**
    localStorage.getItem(key);
  - **Removing data-**
    localStorage.removeData(key);

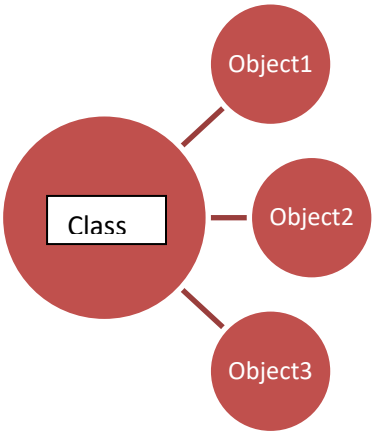
❖ **Object Getters & Setters: -**

```
<body>
 <script>
  //student object
  let student1 = {
   name: "Ajay Pandey",
   college: "IIT Kanpur",
   city: "Kanpur",

   get getName() {
    return this.name;
   },

   set setName(name) {
    console.log("Setting value to object using set");
    this.name = name;
   },

   get getCollege() {
    return this.college;
   },

   set setCollege(college) {
    console.log("Setting value to object using set");
    this.college = college;
   },

   display: function () {
    return `${this.name} : ${this.college} : ${this.city}`;
   },
  };

  console.log(student1.display()); //Ajay Pandey : IIT Kanpur : Kanpur

  //Change the name using setter
  student1.setName = "Vishal Srivastava";
  console.log(student1.display()); //Vishal Srivastava : IIT Kanpur : Kanpur

  //Getter
  console.log(student1.getName); //Vishal Srivastava

  //Object constructor
  function Employee(id, phone, age) {
   this.empId = id;
   this.empPhone = phone;
   this.empAge = age;
```

```
    this.display = function () {
      return this.empId + " : " + this.empPhone + " : " + this.empAge;
    };
  }

  let e1 = new Employee(101, 4923232, 23);
  console.log(e1.display()); //101 : 4923232 : 23

  let e2 = new Employee(102, 4616321136, 26);
  console.log(e2.display()); //102 : 4616321136 : 26
 </script>
</body>
```

❖ **Class & Object: -**

- ECMAScript2015, also known as ES6, introduced JavaScript classes.
- JavaScript classes are templates for JavaScript objects.
- A Class can have multiple objects.



- **class** keyword is used to create a class in JS.
  Syntax-

```
class ClassName{
   //body of class
}
```

❖ **Constructor Method: -**

- It should have exact name **"constructor".**
- It is executed automatically when a new object is created.
- It is used to initialize object properties.
- Syntax-

```
class ClassName{
  constructor()
  {
    ....
  }
}
```

- Example of Class & Object-

```
<body>
 <h1>Class & Object in JavaScript</h1>

 <script>
    class Student
    {
       constructor(name, phone)
       {
          this.name = name;
          this.phone = phone
          console.log("Object created...");
       }

       showDetails()
       {
          console.log("Name: "+this.name);
          console.log("Phone: "+this.phone);
          console.log("----------------------");
       }

       getName(){
          return this.name.toUpperCase();
       }
    }

    //Creating object
    let s1 = new Student("Vishal", 41523263);
    s1.showDetails();
    console.log(s1.getName());

 </script>
</body>
```

❖ **ES6 New Features: -**

1. **Arrow Function-**

```
const square = (x) => x*x;
console.log(square(5)); //25

let array = [10, 20, 30];
array.forEach((e) => console.log(e)); //10 20 30
```

2. **Multiline String-**

```
let str = `Hi!
I'm Vishal Srivastava.
How are you?
`;
console.log(str);

// Hi!
// I'm Vishal Srivastava.
// How are you?
```

3. **Default Parameters-**

```
const sum1 = (x, y) => x + y;
console.log(sum1()); //NaN
console.log(sum1(2, 3)); //5

const sum2 = (x = 1, y = 2) => x + y; //default values of x & y
console.log(sum2()); //3
console.log(sum2(2, 3)); //5
```

4. **Template Literals-**

```
let x = 12;
let y = 45;

let str1 = `Value of x: ${x}
Value of y: ${y}`;
console.log(str1);
// Value of x: 12
// Value of y: 45
```

5. **Destructuring Assignment-**

```
//Array Destructuring
let myArr1 = [22, 55];
let [a1, b1] = myArr1;
console.log(a1); //22
console.log(b1); //55

let myArr2 = [11, 22, 33, 44, 55];
let [a2, b2, ...c] = myArr2;
```

```
console.log(a2); //11
console.log(b2); //22
console.log(c); //[ 33, 44, 55 ]

//Object Destructuring
let obj = {
  city: "Ayodhya",
  state: "Uttar Pradesh",
};

let { city, state } = obj;
console.log(city); //Ayodhya
console.log(state); //Uttar Pradesh
```

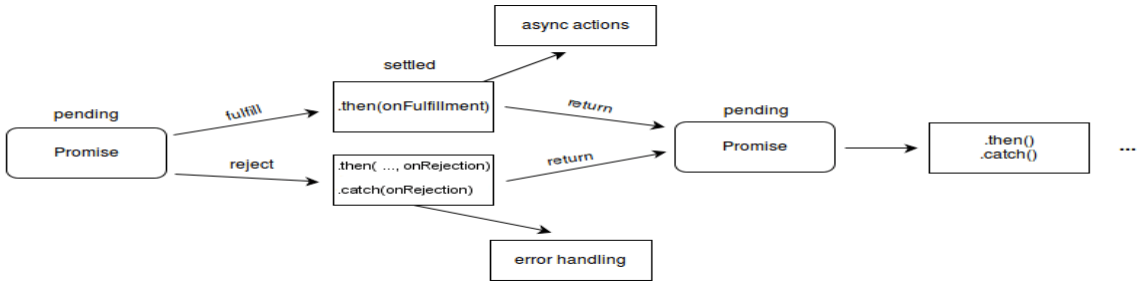
6. **Enhanced Object Literals-**

```
let name = "Ram",
  country = "India";
let student = {
  name,
  country,
};
console.log(student); //{ name: 'Ram', country: 'India' }
```

```
function createObject(p, q, r) {
  return { p, q, r };
}
console.log(createObject(1, 2, 3)); //{ p: 1, q: 2, r: 3 }
```

7. **Promise in JS-**

- The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.
- A Promise is a proxy for a value not necessarily known when the promise is created. It allows you to associate handlers with an asynchronous action's eventual success value or failure reason. This lets asynchronous methods return values like synchronous methods: instead of immediately returning the final value, the asynchronous method returns a promise to supply the value at some point in the future.
- A Promise is in one of these states:
  1. pending: initial state, neither fulfilled nor rejected.
  2. fulfilled: meaning that the operation was completed successfully.
  3. rejected: meaning that the operation failed.
- A promise is said to be settled if it is either fulfilled or rejected, but not pending.



- Example-

```
let prom = new Promise((resolve, reject) => {
  // "Producing Code" (May take some time)
  let flag = true;

  setTimeout(() => {
    if (flag) {
      resolve();
    } else {
      reject();
    }
  }, 3000);
});

prom
  .then(() => {
    console.log("Promise is resolved");
  })
  .catch(() => {
    console.log("Promise is rejected");
  });

console.log("First line after Promise");
console.log("Second line after Promise");
console.log("Third line after Promise");
console.log("Fourth line after Promise");
//Output
// First line after Promise
// Second line after Promise
// Third line after Promise
// Fourth line after Promise
// Promise is resolved
```

8. **Class & Object-**

```
class Student{
   constructor(firstName, lastName, city){
      this.firstName=firstName;
      this.lastName=lastName;
      this.city=city;
   }

   display(){
      console.log(`${this.firstName} ${this.lastName} lives in ${this.city}.`);
   }
}

let student1 = new Student("Vishal", "Srivastava","Ayodhya");
student1.display(); //Vishal Srivastava lives in Ayodhya.
```

❖ **Modules in JS: -**

- JavaScript modules allow you to break up your code into separate files.
- This makes it easier to maintain a code-base.
- Modules are imported from external files with the **import** statement.

```
import {var1, display} from "./helper.js"
import printName from "./helper.js"; //default export
```

- Modules also rely on type="module" in the <script> tag.

```
Ex- <script type="module" src="main.js"></script>
```

- Modules that we want to export should be done using **export** keyword.

Ex-

```
//In-line individually
export let var1 =25;

export function display(){
   console.log("Variable: "+var1);
}

//All at once
export {var1, function};
```

- There can be only one **default export** in a file.

```
//Main File
import printName from "./helper.js"; //default export

//External File
export default function printName(){
   console.log("My name is Vishal");
}
```