Client-Side Scripting Language (22519)

Vidyalankar Polytechnic

Unit 2.1
Array

Declaring an Array

- An array can hold many values under a single name, and you can access the values by referring to an index number.
- An array is a single variable that is used to store different elements.

Syntax:

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

Example :

```
var cars = [“Ritz", " Honda", "BMW"];

var person = ["Sagar",99,"Dadar",75.5];
```

Declaring an Array

There are basically two ways to declare an array.

Way 1 : The new Array() Constructor

Way 2 : The Literal Notation

Way 1 : The new Array() Constructor

- The Array() constructor creates Array objects.
- You can declare an array with the "new" keyword to instantiate the array in memory.

```
var  x = new Array();              //an empty array

var  x = new Array(10);          //an empty array for 10 elements
```

Way 2 : The Literal Notation

- Instead of new Array() , you can use square brackets [].
- When we declare array using square brackets is called the "array literal notation":

```
var  x = [];                    //an empty array

var  x = [5];                   //array with one element
```

Initializing an Array

An array in JavaScript can be defined and initialized in two ways, array literal and Array constructor syntax.

Example using Array Constructor (method 1)

```
//creates an array having elements 10,20,30,40,50
var house =  new Array ( 10,20,30,40,50);
//creates an array of 5 undefined elements
var house1 = new Array(5);
//creates an array with element 1BHK
var home = new Array("1BHK");
```

Initializing an Array

Example using Array Literal (for Method 2):

```
 //initializing while declaring
var house = [ “1BHK”, “2BHK”, “3BHK”, “4BHK”];
//initializing after declaring
house[0] = “1BHK”;
house[1] = “2BHK”;
house[2] = “3BHK”;
house[3] = “4BHK”;
```

Defining Array Elements

- An array in JavaScript can hold different elements
- We can store Numbers, Strings and Boolean in a single array.

Example:

//storing Numbers, Strings and Boolean in an array

```
var house= [“1BHK”, 1200, “3BHK”, 1600, true];
```

Looping an Array

Loops are handy, if you want to run the same code over and over again, each time with a different value.

We can use arrays within loops and access array elements using loops in java scripts .

For Example –

```
for (i = 0; i < cars.length; i++)
{
  document.write(cars[i]+"<br>");
}
```

Example : Looping an Array

```
<html>
<body>
<h2>JavaScript For Loop</h2>
<script>
var cars = ["BMW", "Volvo", "Ford", "Fiat"];
var text = "";
var i;
for (i = 0; i < cars.length; i++)
{
  document.write(cars[i]+"<br>");
}
</script>
</body>
</html>
```

OUTPUT:

```
JavaScript For Loop
BMW
Volvo
Ford
Fiat
```

Adding an Element to Array

Method1:

The easiest way to add a new element to an array is using the push() method.

The push() method adds new items to the end of an array, and returns the new length.

Syntax:

array.push(*item1, item2, ..., itemX*);

Example:

```
var fruits = [ "Banana", "Orange", "Apple", "Mango” ];
fruits.push( "Lemon” );    // adds a new element (Lemon) to fruits
```

Adding an Element to Array

Method 2:

The unshift() method adds one or more elements to the beginning of an array and returns the new length of the array.

Syntax:

array.unshift(*item1, item2, ..., itemX*);

Example:

```
var fruits = [ "Banana", "Orange", "Apple", "Mango” ];
fruits.unshift( "Lemon","Pineapple” );
```

Sorting an Array

The array.sort() is an inbuilt method in JavaScript which is used to sort the array.

Syntax:

```
array.sort();
```

Here array is the set of values which is going to be sorted.

Reversing an Array

- ✓ The **reverse()** method reverses the elements in an array.
- ✓ You can use it to sort an array in descending order.
- ✓ **Syntax:** array.reverse();
- ✓ **Example:**

Example : Sorting and Reversing an Array

```
<script>
var fruits = ["Banana", "Watermelon", "Chikoo", "Mango", "Orange",
"Apple"];
fruits.sort();
document.write(fruits+"<br>");
fruits.reverse();
document.write(fruits+"<br>");
</script>
```

```
Apple,Banana,Chikoo,Mango,Orange,Watermelon
Watermelon,Orange,Mango,Chikoo,Banana,Apple
```

Combining an Array Element into String

- The array.join() method is an inbuilt function in JavaScript which is used to join the elements of an array into a string.
- The elements of the string will be separated by a specified separator and its default value is comma(,).

Syntax:

```
array.join(separator);
```

Parameters:

Separator: It is Optional . it can be either used as parameter or not. Its default value is comma(,).

Return Value: It returns the String which contain the collection of array's elements.

Combining an Array Element into String

Example 1 :

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var energy = fruits.join();
```

Output:

```
Banana,Orange,Apple,Mango
```

Example 2:

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
var energy = fruits.join(“ and “);
```

Output:

```
Banana and Orange and Apple and Mango
```

Combining an Array Element into String

- The array.concat() method creates a new array by concatenating two arrays.

Syntax:

```
array.concat( );
```

Example:

```
   var CO_Subject = ["PHP", "CSS", "Java"];
    var Math_Subject= ["Applied Math", "Elements of Maths"];
    var subjects = CO_Subject.concat(Math_Subject);
    document.write(subjects);
```

Output: **PHP,CSS,Java,Applied Math,Elements of Maths**

Changing Element of an Array

- JavaScript gives you several ways to modify arrays.
- One of the way is to give an existing array element a new value.
- This is as easy as assigning the value. Follow these steps in your JavaScript Console:

1.Create a new array with the following statement:

```
var people = [“Rahul",“Virat",“Dhoni"];
```

2. Print out the values of the array elements with following:

```
document.write(people);
```

Changing Element of an Array

3.Change the value of the first element by entering this statement, and then press Return or Enter:

```
people[0] = “Sachin";
```

4. Print the values of the array’s element now, using the following statement:

```
document.write(people);
```

The value of the first array element has been changed from “Rahul” to “Sachin”.

```
var people = [“Sachin ",“Virat",“Dhoni"];
```

Vidyalankar Polytechnic

Changing Element of an Array

- ✓ Shifting is equivalent to popping, working on the first element instead of the last.
- ✓ The **shift()** method removes the first array element and "shifts" all other elements to a lower index.
- ✓ **Syntax:** array.shift();

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits);
 {
   fruits.shift();
   document.write("<br>"+fruits);
}
```

```
Banana,Orange,Apple,Mango
Orange,Apple,Mango
```

Changing Element of an Array

- ✓ Array elements are accessed using their index number.
- ✓ Array indexes start with 0.
- ✓ **Example:**

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits+"<br>");
fruits[2] = "Kiwi";
document.write(fruits);
```

```
Banana,Orange,Apple,Mango
Banana,Orange,Kiwi,Mango
```

Changing Element of an Array

- ✓ The length property provides an easy way to append a new element to an array:
- ✓ Example:

```
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits+"<br>");
fruits[fruits.length] = "Kiwi";
document.write(fruits+"<br>");
fruits[fruits.length] = "Chikoo";
document.write(fruits);
</script>
```

```
Banana,Orange,Apple,Mango
Banana,Orange,Apple,Mango,Kiwi
Banana,Orange,Apple,Mango,Kiwi,Chikoo
```

Changing Element of an Array

- ✓ The **splice()** method can be used to add new items to an array, and removes elements from an array.
- ✓ **Syntax:**

arr.splice(start_index,removed_elements, list_of_elemnts_to_be_added)

- ✓ **Parameter:**
  - The first parameter defines the position where new elements should be added (spliced in).
  - The second parameter defines how many elements should be removed.
  - The list_of_elemnts_to_be_added parameter define the new elements to be added(optional).

Changing Element of an Array

```
<script>
var fruits = ["Banana", "Watermelon", "Chikoo", "Mango", "Orange",
"Apple"];
document.write(fruits+"<br>");
fruits.splice(2,2, "Lemon", "Kiwi");
document.write(fruits+"<br>");
fruits.splice(0,2); //removes first 2 elements from array
document.write(fruits+"<br>");
</script>
```

```
Banana,Watermelon,Chikoo,Mango,Orange,Apple
Banana,Watermelon,Lemon,Kiwi,Orange,Apple
Lemon,Kiwi,Orange,Apple
```

Changing Element of an Array

- ✓ The slice() method slices out a piece of an array into a new array.
- ✓ Syntax:

  arr.slice(array starting from array element 1);

- ✓ **Parameter:**
  - slices out a part of an array starting from array element 1.

Changing Element of an Array

```
<script>
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
document.write(fruits);
var citrus = fruits.slice(2);
document.write("<br>"+citrus);
</script>
```

```
Banana,Orange,Lemon,Apple,Mango
Lemon,Apple,Mango
```

Changing Element of an Array

- ✓ The pop() method is used to remove the last element of the array and also returns the removed element.
- ✓ This function decreases the length of the array by 1.
- ✓ **Syntax:** arr.pop();
- ✓ **Example:**

```
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits+"<br>");
fruits.pop();
document.write(fruits);
</script>
```

```
Banana,Orange,Apple,Mango
Banana,Orange,Apple
```

IndexOf ()

- ✓ The **indexOf()** method searches the array for the specified item, and returns its position.
- ✓ **Syntax:** arr.indexOf(element, start);
  - •element: This parameter holds the element which index will be return.
  - •start: This parameter is optional and it holds the starting point of the array, where to begin the search the default value is 0.

- ✓ Returns -1 if the item is not found.
- ✓ If the item is present more than once, the indexOf method returns the position of the first occurrence.

lastIndexOf()

- ✓ The **lastIndexOf()** method is used to find the index of the last occurrence of the search element.
- ✓ **Syntax:**

```
arr.lastIndexOf(element, start);
```

- **element:** This parameter holds the element which index will be return.
- **start:** This parameter is optional and it holds the starting point of the array, where to begin the search the default value is 0.
- **Return value:** This method returns the index of the first occurrence of the element. If the element cannot be found in the array, then this method returns -1.

Example

```
<script>
  var fruits = ["Banana", "Orange", "Apple", "Mango","Apple","Pine-
apple"];
  var a = fruits.indexOf("Apple");
  document.write("Index of an Apple is:"+a);
    var a = fruits.lastIndexOf("Apple");
  document.write("<br>Index of an Apple is:"+a);
     var a = fruits.lastIndexOf("Lime");
  document.write("<br>Index of an Lime is:"+a);
</script>
```

```
Index of an Apple is:2
Index of an Apple is:4
Index of an Lime is:-1
```

2D Array

- ✓ The **two-dimensional array** is a *collection of items which share a common name and they are organized as a matrix in the form of rows and columns*.
- ✓ The two-dimensional array is an array of arrays, so we create an array of one-dimensional array objects.
- ✓ Example:

```
var branch = [
            ['Computer Engg', "CO"],
            ['Information Technology', "IF"],
            ['Electronics and Telecommunication', "EJ"]
      ];
```

Multi-dimensional Array

✓ Example:

```
var my_ans = new Array(); // declaring array
my_ans.push({0:45,1:55,2:65});
my_ans.push({0:145,1:155,2:165});
my_ans.push({0:245,1:255,2:265});
```

Objects as Associative Array

- Arrays are JavaScript objects.
- The dot (.) operator can be used to access object property .
- The [] operator can also be used to access array property .

Thus, the following two JavaScript expressions have the same value:

```
object.property ;
object["property"] ;
```

To refer to an object property using array notation, simply pass the property name as a String to the array square brackets applied to the object, as follows:

```
objectName["propertyName" ] ;
```

Example : Object as an Associative Array

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

var object1 = new Object;
object1.name = “Girija";
object1.nationality = "Indian";

document.write(" property name: " + object1["name"] );
document.write("<br>");
document.write(" property nationality: " + object1["nationality" ] );

</script>

</body>
</html>
```

Object as an Associative Array

OUTPUT :

```
property name:Girija
property nationality: Indian
```

Unit 2.2
Function

Function

- ✓ A function is a subprogram designed to perform a specific task.
- ✓ Functions are executed when they are called. This is known as invoking a function.
- ✓ Values can be passed into functions and used within the function.
- ✓ Functions always return a value. In JavaScript, if no return value is specified, the function will return undefined.

Defining a Function

A function definition (also called a function declaration, or function statement) consists of the function Keyword , followed by :

- The name of the function.
- A list of parameters to the function, enclosed in parentheses and separated by commas.
- The JavaScript statements that define the function, enclosed in curly brackets, { }.

Defining a Function

Syntax:

```
function name(parameter1, parameter2, parameter3)
{
  // code to be executed
}
```

For example :

```
function square(number)
{
  return number * number;
}
```

Writing a Function

```
<html>
<body>
<h2>JavaScript Functions</h2>
<script>
function myFunction(p1, p2)
{
  return p1 * p2;
}
document.write( myFunction(4, 3) );
</script>
</body>
</html>
```

Function Definition

Function Call

OUTPUT :

```
JavaScript Functions

12
```

Adding arguments (Calling function with arguments)

- ✓ You can pass arguments to a function.
- ✓ These are variables, either numbers or strings, with which the function is supposed to do something.
- ✓ Of course the output of the function depends on the arguments you give it.

Syntax:

```
function function_name(arg1, arg2)
{
  lines of code to be executed
}
```

Calling Function

- You can pass arguments to a function.
- These are variables, either numbers or strings, with which the function is supposed to do something.
- Of course the output of the function depends on the arguments you give it.

Adding arguments
(Calling function with arguments)

```
<html>
<body>
        <h1>Demo: JavaScript function parameters</h1>
        <script>
                function ShowMessage(firstName, lastName)
               {
                        alert("Hello " + firstName + " " + lastName);
                }
                ShowMessage("Steve", "Jobs");
                ShowMessage("Bill", "Gates");
                ShowMessage(100, 200);
  </script>
</body>
</html>
```

Adding parameters

Calling function with arguments

Calling Function by using call () and apply()

- ✓Call () method is a predefined JavaScript method.
- ✓It can be used to invoke (call) a method with an owner object as an argument (parameter).
- ✓The methods call() and apply() allow you to invoke the function.
- ✓The call() method takes arguments separately.
- ✓The apply() method takes arguments as an array.

Vidyalankar Polytechnic

Calling Function by using call ()

```
var person =     {
 fullName: function()
  {        return this.firstName + " " + this.lastName;       }
 }
 var person1 =
{   firstName:"Yash",
   lastName: "Desai"
}
var x = person.fullName.call(person1);
document.write(x+"<br>");
```

Calling Function by apply()

```
var person =  {
                        fullName: function(city, country)
                        {
   return this.firstName + " " + this.lastName + "," + city + "," + country;
                         }
                }
var person1 = {          firstName:"Chirag",
                          lastName: "Shetty"
                   }
var x = person.fullName.apply(person1, ["Mumbai", "India"]);
document.write(x);
```

Scope of Variables and arguments

- Scope determines the accessibility (visibility) of variables.
- In JavaScript there are two types of scope:
    - ✓ Local scope
    - ✓ Global scope

- JavaScript has function scope: Each function creates a new scope. Scope determines the accessibility (visibility) of these variables. Variables defined inside a function are not accessible (visible) from outside the function.

Scope of Variables and arguments

Local JavaScript Variables

- Variables declared within a JavaScript function, become LOCAL to the function.
- Local variables have Function scope: They can only be accessed from within the function.
- Local variables are created when a function starts, and deleted when the function is completed.

Local variable

Example:

```
function myFunction()
{
  var carName = "Volvo";
  // code here CAN use carName
}
```

Scope of Variables and arguments

Global JavaScript Variables

- A variable declared outside a function, becomes GLOBAL.
- A global variable has global scope: All scripts and functions on a web page can access it.

Example:

```
var carName = "Volvo";

// code here can use carName

function myFunction()
{
  // code here can also use carName
}
```

Global variable

Calling a Function from HTML

- ✓ A function can be called from HTML code.
- ✓ Rather than explicitly calling a function, it will be called in response to an event, such as when the page is loaded or unloaded by the browser.
- ✓ For example,

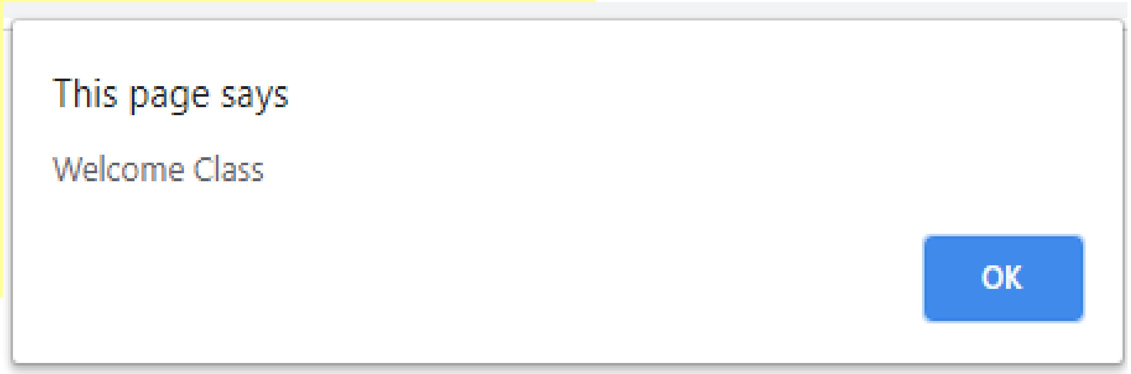
```
<body onload="welcome()">
```

Calling a Function from HTML

```
<html>
<head>
<script type="text/javascript">
function welcome()
{
alert("Welcome Class")
}
</script>
</head>
<body onload="welcome()">
</body>
</html>
```

Function Definition

Function Call


This page says

Welcome Class

Function calling another function

- A function can call another function inside it.
- Whenever a function is called within another function control is transferred from calling function to the called function.
- In a large scale application which comprises of many functions there may be requirement that one function is calling another and which further calling another in chain.

Function calling another function

Example:

```
function ShowMessage()
{
    alert("Hello World!");
}


function display()
{
ShowMessage();
}
```

Function calling another functio

Output:

Hello World!

Returning value from function

The return statement ends function execution and specifies a value to be returned to the function caller.

Syntax:

```
return value;
```

(where value is Optional. It specifies the value to be returned to the function caller. If omitted, it returns undefined.)

Returning value from function

Example: Calculate the product of two numbers, and return the result

```
var x = myFunction(4, 3);        // Function is called


function myFunction(a, b)
{
  return a * b;             // Function returns the product of a and b
}
```

Unit 2.4
String

String

The **JavaScript string** is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

A) By string literal:

The string literal is created using double quotes.

The syntax of creating string using string literal is given below:

```
var stringname="string value";
```

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

String

The **JavaScript string** is an object that represents a sequence of characters.

There are 2 ways to create string in JavaScript

B) By string object (using new keyword)

Syntax:

```
var stringname=new String("string Object you can create str");
```

Example:

```
<script>
var stringname=new String("hello javascript string");
document.write(stringname);
</script>
```

String properties

Property	Description
length	Returns the length of a string.
prototype	Allows you to add new properties and methods to an String object.
constructor	This property returns a reference to the string function that created the object.

```
<script type = "text/javascript">
      var str = new String( "Vidyalankar Polytechnic" );
      document.write("str.length is:" + str.length);
</script>
```

String methods

Methods	Description
charAt()	It provides the char value present at the specified index.
charCodeAt()	It provides the Unicode value of a character present at the specified index.
concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndexOf()	It provides the position of a char value present in the given string by searching a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.

String methods

Methods	Description
match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.
toLowerCase()	It converts the given string into lowercase letter.

String methods

Methods	Description
toString()	It provides a string representing the particular object.
valueOf()	It provides the primitive value of string object.
split()	It splits a string into substring array, then returns that newly created array.
trim()	It trims the white space from the left and right side of the string.
fromCharCode()	The fromCharCode() method converts Unicode values into characters.

JavaScript String charAt(index) Method

```
<script>
var str="javascript";
document.write(str.charAt(2));
</script>
```

```
v
```

JavaScript String charCodeAt(index) method

```
<script>
var x="Javatpoint";
document.writeln(x.charCodeAt(3));
</script>
```

```
97
```

JavaScript String concat(str) Method

```
<script>
var s1="Vidyalankar ";
var s2="Polytechnic";
var s3=s1.concat(s2);
document.write(s3);
var s4=s1+s2;
document.write("<br>"+s4);
</script>
```

```
Vidyalankar Polytechnic
Vidyalankar Polytechnic
```

JavaScript String slice(beginIndex, endIndex) Method

```
<script>
var s1="Vidyalankar Polytechnic";
var s2=s1.slice(12,16);
document.write(s2);
</script>
```

```
Poly
```

JavaScript String lastIndexOf(str) Method

```
<script>
var s1="Vidyalankar Polytechnic, Mumbai";
var n=s1.indexOf("a");
var n1=s1.lastIndexOf("a");
document.write(n);
document.write("<br>"+n1);
</script>
```

```
4
29
```

V	i	d	y	a	l	a	n	k	a	r		P	o	l	y	t	e	c	h	n	i	c	,		M	u	m	b	a	i
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

JavaScript String trim() Method

```
<script>
var s1="     Vidyalankar     Polytechnic   ";
var s2=s1.trim();
document.write(s2);
</script>
```

```
Vidyalankar Polytechnic
```

JavaScript String split() Method

```
<script>
var str="CO IF EJ";
document.write(str.split(" ")); //splits the given string.
</script>
```

```
CO,IF,EJ
```

JavaScript String search() Method

```
<script>
var str="JavaScript is a scripting language.";
document.writeln(str.search("scripting"));
</script>
```

```
16
```

JavaScript String match() Method

```
<script>
varstr="JavaProgramming";
document.writeln(str.match("Java"));
</script>
```

```
Java
```

JavaScript String replace() Method

```
<script>
var str="JavaProgramming";
document.writeln(str.replace("Programming","Script"));
</script>
```

JavaScript

JavaScript String substr() Method

```
<script>
var str="JavaScript";
document.writeln(str.substr(0,6));
</script>
```

JavaSc

JavaScript String substring() Method

```
<script>
var str="JavaScript";
document.writeln(str.substring(4,9));
</script>
```

```
Scrip
```

JavaScript String slice() Method

```
<script>
var str = "JavaScript";
document.writeln(str.slice(0));
document.writeln("<br>"+str.slice(4));
</script>
```

```
JavaScript
Script
```

JavaScript String toString() and valueOf()

```
<script>
var str="JavaScript";
document.writeln(str.toString());
document.writeln("<br>"+str.valueOf());
</script>
```

```
JavaScript
JavaScript
```

JavaScript String toLowerCase() , toUpperCase()

```
<script>
var str = "JavaScript";
document.writeln(str.toLowerCase());
document.writeln("<br>"+str.toUpperCase());
</script>
```

```
javascript
JAVASCRIPT
```

JavaScript String fromCharCode()

```
<script>
  var res = String.fromCharCode(72, 69, 76, 76, 79);
   var res1 = String.fromCharCode(73, 70, 77, 77, 80);
  document.write(res);
    document.write("<br>"+res1);
  </script>
```

```
HELLO
IFMMP
```

Converting string to Number

- ✓ parseInt(): converts a string into an integer.
- ✓parseFloat(): converts a string into a decimal points. (floating point)
- ✓Number( ): converts a string into number.

Example

```
<script>
 var a=50;
 var b="67";
 var c="45.75";
 var ans=a + parseInt(b)+parseFloat(c);
 document.write("Addition="+ans);
 var sum=a+ Number(b)+parseFloat(c);
  document.write("<br>"+"SUM="+sum);
</script>
```

```
Addition=162.75
SUM=162.75
```

Converting Numbers into string

✓ toString(): convert integer value and decimal value into a string.

Example:

```
<script>
 var a=50;
 var b=80
 var ans=a + b.toString();
 document.write("Addition="+ans);
 </script>
```

```
Addition=5080
```

Thank
You