Unit 2
Array, Function and String

Marks: 14 (R-2, U-4, A-8)

Course Outcome: Implement arrays and functions in javascript.

Unit Outcome:

1. Create array to solve the given problem.
2. Perform the specified string manipulation operation on the given string.
3. Develop javascript to implement the given function.
4. Develop javascript to convert the given Unicode to character form.
5. Develop javascript to convert the given character to Unicode and vice-versa.

Topics and Sub-topics:

2.1 Array-Declaring an array, initializing an array, defining an array element, looping an array, adding an array element, sorting an array element, combining an array element into a string, changing elements of an array, Objects as Associative array.

2.2 Function-Defining a function, writing a function, adding an argument, scope of variable and arguments

2.3 Calling a function- calling a function with or without an argument, calling function from HTML, function calling another function, returning the value from a function.

2.4 String- manipulate a string, joining a string, retrieving a character from given position, retrieving a position of character in a string, dividing a text, copying a sub-string, converting string to number and numbers to string, changing the case of string, finding Unicode of a character.

2. Introduction

- Array, Function and String are the basic concepts in JavaScript.
- Array in JavaScript is used to store multiple values in a single variable.
- JavaScript function is the block of code designed to perform a particular task.
- A function is a group of reusable code which can be called anywhere in the JavaScript program. This eliminates the need of writing the same code again and again.
- The string object in JavaScript let you works with the series of characters. JavaScript strings are used for storing and manipulating text.

2.1 Array

- Array is used to store a set of values (different types) in a single variable name.
- Arrays are the fundamentals part of most programming languages and scripting languages.
- Using array, we can store multiple values under a single name.

2.1.1. Declaring an Array

In JavaScript, objects and array are handled almost identically, because arrays are merely a special kind of object.

There are two ways to declare the arrays:

1) By using new Array() constructor:

The Array() constructors creates the Array objects..

You can declare an array with the "new" keyword to instantiate the array in memory.

You can invoke this constructor in three distinct ways:

A. Call it with no arguments:

```
var a= new Array();
```

This method is creating an empty array with no elements and is equivalent to the array literal [].

B. Call it with a single numeric argument, which specifies a length:

```
Var a = new Array(10);6
```

This technique creates an array with specified length.

C. Explicitly specify two or more array elements or a single non-numeric element for the element for the array:

Var a = new Array(5,4,3,2,1,"testing", "testing");

In this form, the constructor arguments become the elements of the new array.

2) By using 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= [];

Var x=[5];

2.1.2 Initializing an Array:

Initialization is the process of assigning a value when either a variable or an array is declared.

When initializing an array, you place the value within the parentheses of the Array() constructor.

The following example initializes the products array with the value 'BMW', which is assigned to the first element of this array:

var products = new Array('BMW')

In the real world, an array usually has more than one array element, with each element having its own value. Therefore, you'll find yourself having to initialize the array with more than one value.
Here's how this is done:
var products = new Array('BMW', 'Maruti', 'Mahindra')

while Initializing array with declaration then all elements must specify in parenthesis and elements should be separated by a comma.

Code:

```
<html>
<head>
<title>Array</title>
</head>
```

```
<body>
<script>
 var products = new Array('BMW', 'Maruti', 'Mahindra');
document.write("Length of array is :"+products.length);
</script>
</body>
</html>
```

Output:

Length of array is : 3

2.1.3 Defining Array Elements

Array is used to store a set of values in a single variable name. In order to differentiate between these set of values. Array make use of index.

Once array is declared we can define its elements as follows:

var cars = new Array(3);

cars[0] = "BMW";

cars[1] = "Maruti";

cars[2] =2546 ;

As JavaScript automatically changes the value of length when you add elements to an array that you created with the Array keyword. Array indicates in JavaScript always start at 0 , not 1, so the length property is always one greater than the largest index in the array.

Accessing an array value is quite easy. We use array index to access a value. If we want to access val 1 from the above syntax , we use Array[0], So,

Array[0] holds "BMW"

Array[1] holds "Maruti"

Array[2] holds "Honda"

Code:

```
<html>
<head>
<title> Array</title>
</head>
```

```
<body>
<script>
var cars = new Array(3);
cars[0] = “BMW”;
cars[1]=”Maruti”;
cars[2]=”Honda”;
document.write(cars[0]);
document.write(“<br>”+ cars[1]);
document.write(“<br>”+ cars[2]);
</script>
</body></html>
```

Output:

BMW

Maruti

Honda

2.1.4 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 JavaScript.

To identify how many loops, we must know the total number of elements present inside the array which can find out using array_name.length.

Example:

```
<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

2.1.4 Adding an Array Element

On some occasions your JavaScript will need to increase the size of the array while your JavaScript is running.
There are two methods to add the elements into the 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

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");

Example:

```
<html>
<head>
<title> Array</title>
<body>
<script>
  var fruits = new Array(3);
 fruits[0] = "Banana";
  fruits[1] = "Orange";
 fruits[2] = "Apple";
 fruits[3] = "Mango";
for(i=0;i<fruits.length;i++)
{
    document.write(fruits[i] +" "+"<br>");
}
```

```
fruits.push("Lemon");
fruits.unshift("Pineapple");

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

Output

Banana
Orange
Apple
Mango
Pineapple Banana Orange Apple Mango Lemon

2.1.5 Sorting Array Elements:

The index of the array elements determines the order in which values appear in an array when a for loop is used to display the array. Sometimes you want values to appear in sorted order, which means that strings will be presented alphabetically and numbers will be displayed in ascending order. You can place an array in sorted order by calling the sort() method of the array object. the index of the element to which the value is assigned.
Here's what you need to do to sort an array:
1. Declare the array.
2. Assign values to elements of the array.
3. Call the sort() method.

Example:

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

```
var products = new Array();
products[0] = 'Soap ';
products[1] = 'Water';
products[2] = 'Pizza';
products[3] = 'Fan';
products.sort();
for (var i = 0; i < products.length; i++)
{
document.write(products[i] + '<br>');
}
</script>
</body>
</html>
```

Output:

Fan
Pizza
Soap
Water

2.1.6 Reversing an Array Element:

The reverse() method reverses the elements in an array.

- You can use it to sort an array in descending order.
- Syntax: array.reverse();

Example:

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

```
document.write(fruits+"<br>");
</script>
</body>
</html>
```

Output:

Apple,Banana,Chikoo,Mango,Orange,Watermelon

Watermelon,Orange,Mango,Chikoo,Banana,Apple

2.1.7 Combining an Array Element into String

Sometimes there is need to combine all elements of an array into a single string.

Let car is an array consist of three elements as follows:

var car = new Array(3);

car[0] = “BMW”;

car[1] = “Maruti”;

car[2] = “Honda”;

Array elements can be combined in two ways:

1)Using the concat() method:

The concat() method separates each value with a comma. It is used to join one or more 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

2. Using the join() method:

- 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(,).
Example:

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

var products = new Array();
products[0] = 'Car ';
products[1] = 'Water';
products[2] = 'Soap';
products[3] = 'Pizza';
var str = products.concat();
document.write(str);
document.write('<br>');
var str = products.join(' ');
document.write(str);
</script>
</body>
</html>
```

Output:

Car,Water,Soap,Pizza

Car Water Soap Pizza

2.1.8 Changing Elements of an Array:

JavaScript provides **shift()** method to remove the first element of an array.

The shift() method removes the first element of the array then moves the other tasks up on the list and returns the removed item.

Example:

```
<html>
<head>
<title> Array</title>
</head>
<body>
 <script>
    Var car = new Array(3);
    car[0]=”BMW”;
   car[1]=”Honda”;
   car[2]=”Maruti”;
  car.shift();
 for (i=0;i<car.length;i++)
{
   document.write(items[i]+””);
}
</script>
</body>
</html>
```

Output:
Honda Maruti

The **pop()** method remove an item from the end of an array

Example:

```
<html>
<head>
<title> Array</title>
</head>
<body>
 <script>
    Var car = new Array(3);
    car[0]=”BMW”;
   car[1]=”Honda”;
   car[2]=”Maruti”;
  car.pop();
 for (i=0;i<car.length;i++)
{   document.write(items[i]+””);
}</script>
</body>
</html>
```

Output:

BMW Honda

2.1.9 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).

Example:

```
<html>
```

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

Output:

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

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.

Example:

```
<html>
<body>

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

</body>
</html>
```

Output:

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

2.1.10 Object 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:

```
<html>
<body>
<script>
var object1 = new Object;
object1.name = “abc";
object1.nationality = "Indian";
document.write(" property name: " + object1["name"] );
document.write("<br>");
document.write(" property nationality: " + object1["nationality" ] );
</script>
</body>
</html>
```

OUTPUT :
property name: Girija
property nationality: Indian

Methods of Array Object:

Sr. No.	Methods	Description
1	concat	Returns a new array consisting of a combination of two array
2	indexOf	Returns the index of the first occurrence of a value in an array.
3	Join	Joins all elements of an array into a string
4	lastIndexOf	Returns the index of the last occurrence of a specified value in an array
5	Pop	Removes the last element of an array
6	Push	Add new elements to the end of an array and returns the new length
7	Reverse	Reverses the order of the elements in an array
8	Shift	Removes the first element of an array and return that element
9	Slice	Selects a part of an array, and returns the new array
10	Sort	Sorts the elements of an array
11	Splice	Adds/Remove elements from array
12	toString	Converts an array to a string, and returns the result
13	unshift	Adds new element at the beginning of an array
14	valueOf	Returns the primitive value of an array

Example: Using array methods:

```
<html>
<body>
<script>
   var arr = new Array();
   arr.push(1,2,3);
  arr.push(5,6);
 document.write("After the Join method" +arr.join(","));
 arr.pop();
 document.write("<br>After the pop method" +arr.join(","));
 arr.shift();
 document.write("<br>After the shift method" +arr.join(","));

arr.reverse();
document.write("<br>After the reverse method"+arr.join(","));

arr.unshift(1);
document.write("<br>After the unshift method" +arr.join(","));
```

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

Output:

After the Join method1,2,3,5,6
After the pop method1,2,3,5
After the shift method2,3,5
After the reverse method5,3,2
After the reverse method1,5,3,2

2-Dimensional 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"]      ];
```

Example: 2-D array

```
    <script>
var branch = [
  ['Computer Engg', "CO"],
  ['Information Technology', "IF"],
  ['Electronics and Telecommunication', "EJ"],
  ['Civil Engineering', "CV"],
  ['Chemical Engg', "CE"],
  ['Instrumnet Engg',"IE"]
];
// display now
```

```
for(i = 0; i < branch.length; i++)
document.write(branch[i][0] + ',' + branch[i][1] + '<br>' );
</script>
```

Output:

Computer Engg,CO
Information Technology,IF
Electronics and Telecommunication,EJ
Civil Engineering,CV
Chemical Engg,CE
Instrumnet Engg,IE

Multi-Dimensional Array:

Multidimensional arrays are not directly provided in JavaScript. If we want to use anything which acts as a multidimensional array then we need to create a multidimensional array by using another one-dimensional array. So multidimensional arrays in JavaScript is known as arrays inside another array. We need to put some arrays inside an array, then the total thing is working like a multidimensional array. The array, in which the other arrays are going to insert, that array is use as the multidimensional array in our code. To define a multidimensional array, its exactly the same as defining a normal one-dimensional array.

Example:

```
<script>
var my_ans = new Array(); // declaring array
my_ans.push({0:45,1:55,2:65,3:45});
my_ans.push({0:145,1:155,2:165,3:"VP"});
my_ans.push({0:245,1:255,2:265});
my_ans.push({0:"aaa",1:"bbb",2:"ccc",3:"ddd"});

// displaying the array data //
```

```
for(i=0;i<4;i++)
{
 document.write("key : " + i + "  =>value: " + my_ans[i][0] +
 ',' +my_ans[i][1] + ',' +my_ans[i][2]+ ',' +my_ans[i][3] + "<br>");
}
</script>
```

Output:

key: 0 =>value: 45,55,65,45
key: 1 =>value: 145,155,165, VP
key : 2 =>value: 245,255,265, undefined
key: 3 =>value: aaa,bbb,ccc,ddd

2.2 Function

Functions are building blocks of any programming language. Function is a block of statements that perform certain tasks.

Functions are of the two types:

1. Built-in functions are the functions that are already defined in JavaScript. Examples are written (), prompt () etc.

2. User-Defined functions are defined by the user.

2.2.1 Defining a function

- A function must be defined before it can be called in a JavaScript statement. If you think about it, this makes sense, because the browser must learn the definition of the word (the function name) before the browser sees the word (the function call) in a statement.
- The best place to define a function is at the beginning of a JavaScript that is inserted in the <head> tag, because then all subsequent JavaScript on the web page will know the definition of that function. The browser always loads everything in the <head> tag before it starts executing any JavaScript.
- A function definition consists of four parts: the name, parentheses, a code block, and an optional return keyword.

Function Name

The function name is the name that you've assigned the function. It is placed at the top of the function definition and to the left of the parentheses. Any name will do, as long as it follows certain naming rules. The name must be

- Letter(s), digit(s), or underscore character
- Unique to JavaScript on your web page, as no two functions can have the same name

The name cannot

- Begin with a digit
- Be a keyword
- Be a reserved word

Parentheses

- Parentheses are placed to the right of the function name at the top of the function definition.
- Parentheses are used to pass values to the function; these values are called arguments.

Code Block

- The code block is the part of the function definition where you insert JavaScript statements that are executed when the function is called by another part of your JavaScript application.
- Open and close French braces define the boundaries of the code block. Statements that you want executed must appear between the open and close French braces.

Return (Optional)

- The return keyword tells the browser to return a value from the function definition to the statement that called the function.

2.2.2 Writing function definition

We can write function with argument and without argument in JavaScript.

Syntax for function without argument:

```
function function_name(parameters...)

{
  Statements ....
```

}

Example:

```
function hellojavascript()
{
  alert("hello");
}
```

2.2.3 Adding an Argument

- A function typically needs data to perform its task. Sometimes you provide the data when you define the function, such as the salary and percentage increase in salary instead of writing this data into the function definition.
- Other times, the data is known only when you run your JavaScript.
- For example, we could ask the user to enter the salary and percentage increase in salary instead of writing this data into the function definition.
- Data that is needed by a function to perform its task that is not written into the function definition must be passed to the function as an argument.
- An argument is one or more variables that are declared within the parentheses of a function definition.

Syntax:

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

Example:

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

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

2.2.4 Scope of Variable and Arguments

- The scope of a variable means how the different parts of the program can access that variable. In JavaScript there are two types of Scope namely: Local Scope and Global Scope.
- A variable can be declared within a function this is called a local variable, because the variable is local to the function. Other parts of your JavaScript don't know that the local variable exists because it's not available outside the function.
- But a variable can be declared outside a function. Such a variable is called a
- global variable because it is available to all parts of your JavaScript—that is, statements within any function and statements outside the function can use a global variable.
- JavaScript developers use the term scope to describe whether a statement of a JavaScript can use a variable. A variable is considered in scope if the statement can access the variable. A variable is out of scope if a statement cannot access the variable.

Example of Local Variable:

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

Example of Global Variable:

```
var carName = "Volvo";

// code here can use carName

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

2.3 Calling a Function

- You call a function any time that you want the browser to execute statements contained in the code block of the function.
- A function is called by using the function name followed by parentheses. If the function has arguments, values for each argument are placed within the parentheses.
- You must place these values in the same order that the arguments are listed in the function definition. A comma must separate each value.

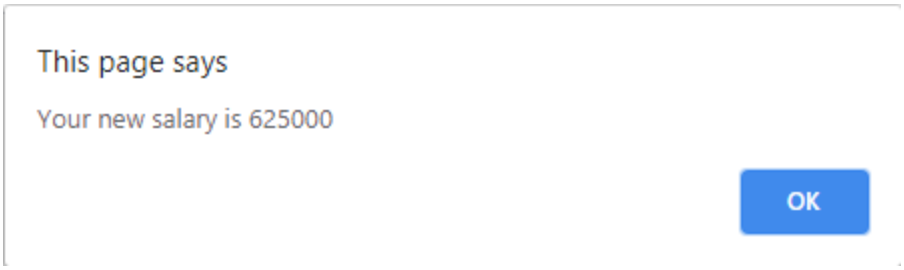
2.3.1 Calling function without argument:

- Here is an example of how to define and call a function that does not have any arguments.
- The function definition is placed within the <head> tag and the function call is placed within the <body> tag.
- When the function is called, the browser goes to the function definition and executes statements within the code block of the function.

Example:

```
<html>
<body>
<script>
function IncreaseSalary()
{
var salary = 500000 * 1.25;
alert('Your new salary is ' + salary);
}
IncreaseSalary();
</script>
</body>
</html>
```

Output:



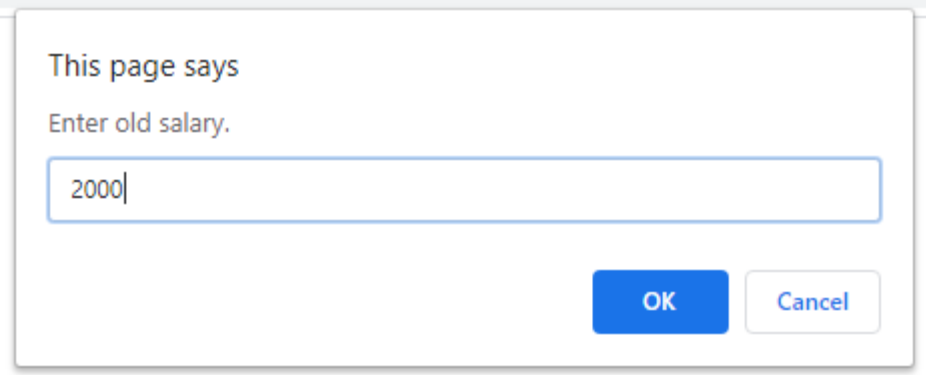
2.3.2 Calling function with arguments:

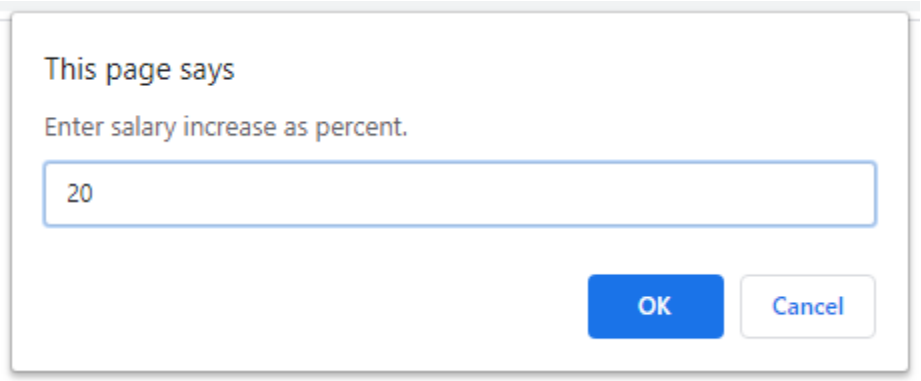
- The Salary and Increase variables are then used within the parentheses of the function call, which tells the browser to assign these values to the corresponding arguments in the function definition. The function calculates and displays the new salary.

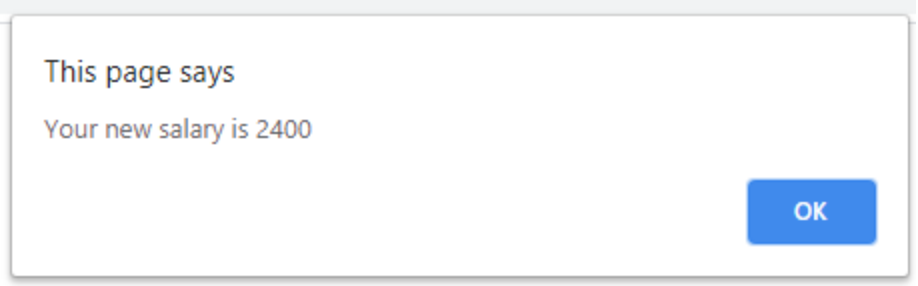
Example:

```
<html>
<body>
<script>
function IncreaseSalary(OldSalary, PercIncrease)
{
var NewSalary =
OldSalary * (1 + (PercIncrease / 100))
alert("Your new salary is " + NewSalary)
}
var Salary = prompt('Enter old salary.', ' ')
var Increase =
prompt('Enter salary increase as percent.', ' ')
IncreaseSalary(Salary, Increase)
</script>
</body>
</html>
```

Output:







2.3.3. Calling Function from HTML

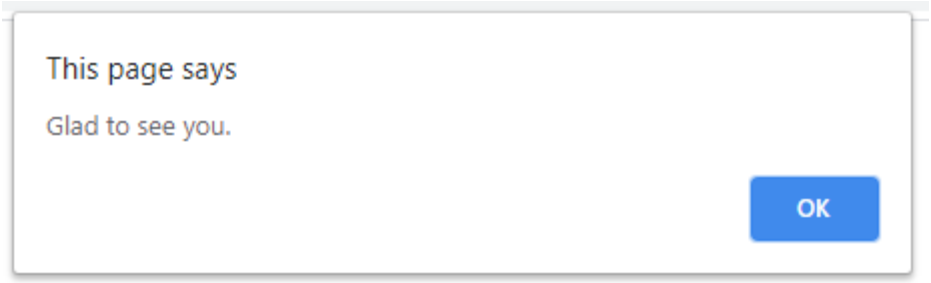
- A function can be called from HTML code on your web page. Typically, a function will be called in response to an event, such as when the web page is loaded or unloaded by the browser.
- You call the function from HTML code nearly the same way as the function is called from within a JavaScript, except in HTML code you assign the function call as a value of an HTML tag attribute.

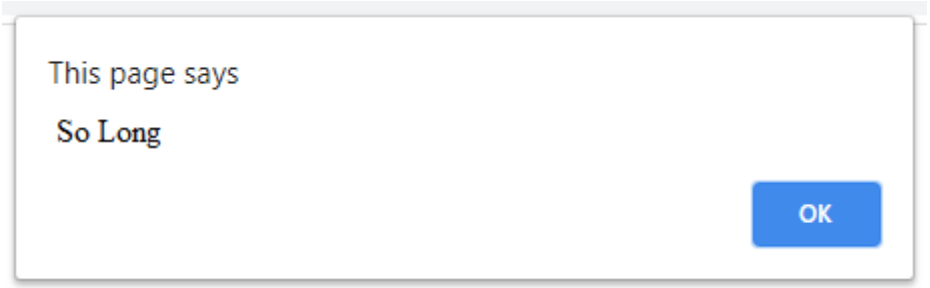
```
<html>
<script >
```

```
function WelcomeMessage()
{
alert('Glad to see you.')
}
function GoodbyeMessage()
{
alert('So long.')
}

</script>
<body onload="WelcomeMessage()"
onunload="GoodbyeMessage()">
</body>
</html>
```

Output:





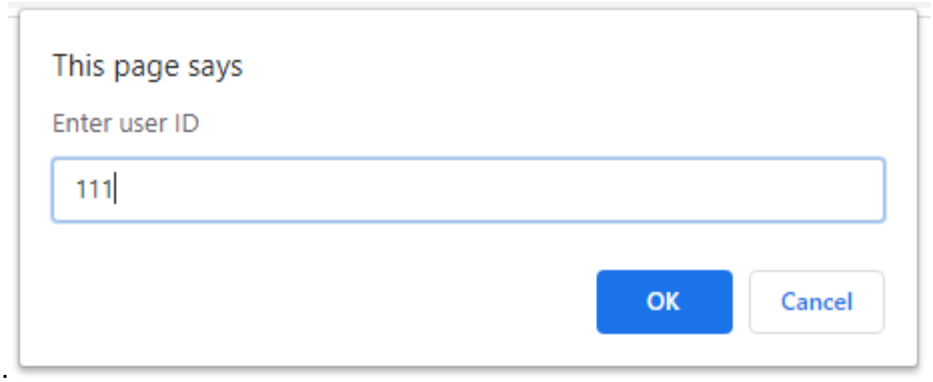
2.3.4 Function Calling another Function

JavaScript developers typically divide an application into many functions, each of which handles a portion of the application.

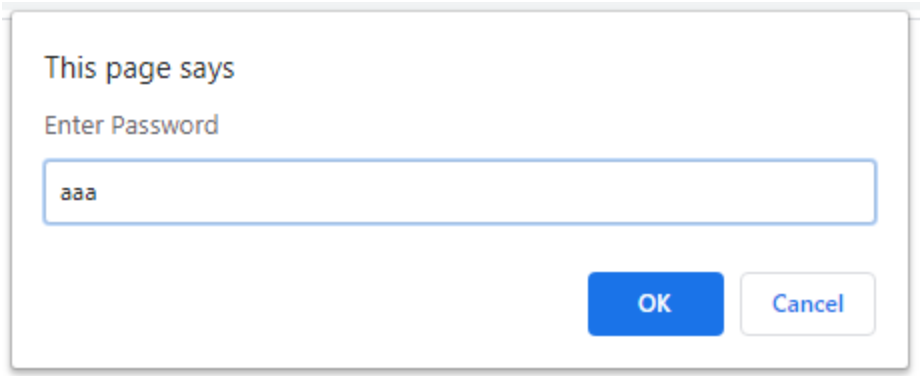
```
<html>
<head>
<title> function calling another function </title>
<script>
function Logon()
{
var userID;
var password;
var valid;
userID=prompt('Enter user ID', ' ');
password=prompt('Enter Password', ' ');
valid=validateLogon(userID, password);
      if(valid === true)
      {
       alert("Valid Logon");
      }
      else
      {
       alert("InValid Logon");
      }
}
function validateLogon(id, pwd)
{
var ret_val;
if(id === '111' && pwd === 'aaa')
{
      ret_val=true;
}
else
{
ret_val=false;
}
return ret_val;
```

```
}
</script>
</head>
<body onload="Logon()">
</body>
</html>
```

Output:



..



2.3.5 Returning a value from functions:

- A function can be designed to do something and then report back to the statement that calls after it's finished
- A function reports back to the statement that calls the function by returning a value to that statement using the return keyword, followed by a return value in a statement.

```
<html>
<bod>
<script>
var x = myFunction(4, 3);
function myFunction(a, b)
{
 return a * b;
 }
   document.write(x);
</script>
</body>
</html>
```

Output:
12

2.4 String

- A string value is chain of zero or more Unicode characters.
- You use string data type to represent text in JavaScript.
- The String object is used to storing and manipulating text. A string is simply storing the series 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";

B) By string object (using new keyword)

Syntax:

var stringname=new String("string literal");

2.4.1 Manipulating string:

In JavaScript there are various properties and methods associated with string objects.

We can make use of these to perform some manipulating on string.

String Properties:

Sr. No.	Property	Description
1	constructor	This property returns a reference to the string function that created the object
2	length	Returns the length of a string.
3	prototype	Allows you to add new properties and methods to a String object.

Code: Length Property

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

Output: str.length is:23

Code: constructor Property

```
<html>
  <head>
    <title>JavaScript String constructor Method</title>
  </head>

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

Output:

str.constructor is:function String() { [native code] }

Code: prototype property

// without using prototype property

```
<html>
<body>
    <h1>without using prototype property</h1>
     <script>
    function Student()
     {
     this.name = 'Pallavi';
     this.gender = 'F';
    }

    var studObj1 = new Student();
    studObj1.age = 20;
    document.write(studObj1.age+"<br>");

    var studObj2 = new Student();
    document.write(studObj2.age);
    </script>
</body>
</html>
```

Output:

With Prototype property:

```
<html>
```

```
<body>
    <h1>Prototype</h1>
     <script>
    function Student()
    {
     this.name = 'Pallavi';
     this.gender = 'M';
    }
    Student.prototype.age = 20;
    var studObj1 = new Student();
     document.write(studObj1.age+"<br>");
    var studObj2 = new Student();
    document.write(studObj2.age);
    </script>
</body>
</html>
```

Output:

Prototype

20
20

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.
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.
toUowerCase()	It converts the given string into uppercase letter.
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.

2.4.2 Joining a String:

- When you concatenate a string, you form a new string from two strings by placing a copy of the second string behind a copy of the first string.
- The new string contains all the characters from both the first and second strings.
- You use the concatenation operator (+) to concatenate two strings, as shown here

NewString = FirstString + SecondString

```
<html>
<bod>
<script>
var s1="Rahul";
var s2="Patil";
```

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

Output:
RahulPatil
RahulPatil

2.4.3 Retrieving a character from given position

- The charAt() method requires one argument, which is the index of the character that you want to copy.

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

Output:

v

2.4.4 Retrieving a position of characters in a string

- To find a character or characters in string we have two methods, indexOf() and search()
- You can determine the index of a character by calling the indexOf() method of the string object.
- The indexOf() method returns the index of the character passed to it as an argument.
- The search () method searches a string for a specified value and returns the position of the match.

```
<html>
<body>
<script>
var s1="JavaScript is a scripting language";
var n=s1.indexOf("a");
document.writeln(n+"<br>");
document.writeln(s1.search("scripting"));
var m=s1.lastIndexOf("a");
document.writeln("<br>"+m);
</script>
</body>
</html>
```

Output:
1
16
31

2.4.5 Dividing Text

- The split() method creates a new array and then copies portions of the string, called a *substring,* into its array elements.
- You must tell the split() method what string (*delimiter*) is used to separate substrings in the string.
- You do this by passing the string as an argument to the split() method.

```
<html>
<body>
<script>
var str="CO IF EJ";
document.write(str.split(" "));
</script>
</body>
</html>
```

Output:
CO,IF,EJ

2.4.6 Copying a Substring:

- Now you've learned how to divide a string into many substrings by using the split() method and a string called a *delimiter*.
- This is useful when you are separating a string containing data elements into individual data elements.
- However, the split() method isn't of much use to you if you need to copy one substring.
- For this, you'll need to use one of two other methods: substring() and substr().

1. The substring()is a method of a string object that copies a substring from a string based on a beginning and an end position that is passed as an argument to the substring() method.
Syntax:
String.substring(startindex, endindex);

2. In the real world, you probably won't know the starting position and end position of characters for your substring, because a user can enter any length string into your application. You can overcome this problem by using the substr() method The substr() method returns a substring. You must tell it the starting position of
the first character that you want to include in the substring and how many characters you want copied into the substring from the starting position. Both positions are passed as arguments to the substr() method.

Syntax:
String.substr(startindex,length);

```
<html>
<body>
<script>
var str="JavaScript";
document.write(str.substr(0,6));
document.write("<br>");
document.writeln(str.substring(4,9));
</script>
</body>
</html>
```

Output:

JavaSc
Scrip

2.4.7 Converting String to number:

- If you need to convert string values to number values, you can do so by converting a number within a string into a numeric value that can be used in a calculation.
- You do this by using the **parseInt()** method and **parseFloat()** method of the string object. The parseInt() method converts a number in a string to an integer numeric value, which is a whole number. You write the parseInt() method this way:
  var num = parseInt(StringName)

- The parseFloat() method is used similarly to the parseInt() method, except the parseFloat() method is used with any number that has a decimal value.
- **Number( ):** converts a string into number.

```
var StrPrice = '10.95'
  var NumPrice = parseFloat(StrCount)
```

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

Output:

Addition=162.75
SUM=162.75

2.4.8 Converting number to string

- As you can probably guess, you need to convert a numeric value to a string before the number can be used in the string.
- You do this by calling the toString() method of the number object.
- The toString() method can be used to convert both integers and decimal values (floats).
- Here's how to convert a number value to a string:
  Var NumCount = 100
  var StrCount = NumCount.toString()

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

Output:
Addition=5080

2.4.9 Changing the Case of the string

- JavaScript provides two methods to change the case of string by using toLowerCase() and toUpperCase() method.

1. toLowerCase(): this method converts all the string character to lowercase.It does not take any argument.

2.toUpperCase(): this method converts all the string character to uppercase. It does not take any argument.

```
<html>
<body>
<script>
var str = "JavaScript";
```

```
document.writeln(str.toLowerCase());
document.writeln("<br>"+str.toUpperCase());
</script>
</body>
</html>
```

Output:

javascript
JAVASCRIPT

2.4.9 Finding a Unicode of a character

- Unicode is a standard that assigns a number to every character, number, and symbol that can be displayed on a computer screen, including characters and symbols that are used in all languages.
- You can determine the Unicode number or the character that is associated with a Unicode number by using the **charCodeAt()** method and **fromCharCode()** method. Both are string object methods.
- The charCodeAt() method takes an integer as an argument that represents the index of the character in which you're interested. If you don't pass an argument, it defaults to index 0.
- The charCodeAt() method returns the Unicode number of the string:
  var UnicodeNum = StringName.charCodeAt()
- If you need to know the character, number, or symbol that is assigned to a Unicode number, use the fromCharCode() method. The fromCharCode() method requires one argument, which is the Unicode number.

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

Output:

97
HELLO
IFMMP

Code:charCodeAt()

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

Output:

97

2.4.10 JavaScript String replace() Method

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

Output:

JavaScript

2.4.11 JavaScript String search() Method

The search() method searches a string for a specified value, and returns the position of the match.

The search value can be string or a regular expression.

This method returns -1 if no match is found.

Syntax: string.search(*searchvalue*);

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

Output:

16

2.4.12 JavaScript String match() Method

The **string.match()** is an inbuilt function in JavaScript which is used to search a string for a match against an any regular expression and if the match will found then this will return the match as an array otherwise it returns null.

Syntax:

string.match(regExp);

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

Output:

Java

22.4.13 JavaScript String slice() Method

The slice() method returns the selected elements in an array, as a new array object.

The slice() method selects the elements starting at the given *start* argument, and ends at, *but does not include*, the given *end* argument

Syntax: *array*.slice(*start*, *end*);

Where,

Start: Optional. An integer that specifies where to start the selection (The first element has an index of 0). Use negative numbers to select from the end of an array. If omitted, it acts like "0"

end: Optional. An integer that specifies where to end the selection. If omitted, all elements from the start position and to the end of the array will be selected. Use negative numbers to select from the end of an array

Code:

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

Output:

JavaScript
Script

Code: Write a javascript function to insert a string within a string at a particular position.

```
<html>
<body>
<button onclick = "myfunction()">click</button>
<script language="javascript" type="text/javascript">
function myfunction()
 {
var s1 = "client  scripting";
var s2 = " side";
var output = [s1.slice(0, 6), s2, s1.slice(6)].join('');
document.write(output);
}
</script>
</body>
</html>
```

Output: