UNIT-3

JAVASCRIPT: FUNCTIONS: Introduction, Program Modules in JavaScript, Programmer – Defined Functions, Function Definitions, Random Number Generation, Example: Game of Chance, Another Example: Random Image Generator, Scope Rules, JavaScript Global Functions, Recursion, Recursion vs. Iteration.

JAVASCRIPT: ARRAYS: Introduction, Arrays, Declaring and Allocating Arrays, Examples Using Arrays, Random Inage Generator Using Arrays, References and Reference Parameters, Passing Arrays to Functions, Sorting Arrays, Searching Arrays: Linear Search and Binary Search, Multidimensional Arrays, Building an Online Quiz.

JAVASCRIPT: OBJECTS: Introduction, Introduction to Object Technology, Math Object, Date Object, Boolean and Number Objects, document Object, window Object, Using Cookies.

JAVASCRIPT: FUNCTIONS

1. INTRODUCTION:

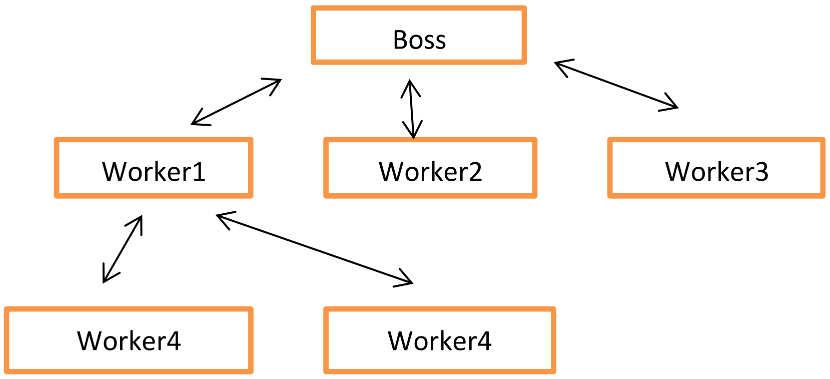
Most computer programs that solve real-world problems are much larger than the programs presented in olden days. Experience has shown that the best way to develop and maintain a large program is to construct it from small, sample pieces, or modules. This technique is called divide and conquer. Here describes many key features of javascript that facilitate the design, implementation, operation and maintenance of larger scripts.

2. Program modules in javascript:

Functions are one of the fundamental building blocks in JavaScript. A function in JavaScript is similar to a procedure—a set of statements that performs a task or calculates a value, but for a procedure to qualify as a function, it should take some input and return an output where there is some obvious relationship between the input and the output.

Modules in javascript are called **functions.** The prepackaged functions that belong to javascript objects are often called METHODS. Javascript provides several objects that have a rich collection of methods for performing common mathematical calculations, string manipulatiopns, date and time mainipulations, and manipulations of collections of data called Arrays.

A function is invoked by a function call. The function call specifies the function name and provides information(arguments) that the called function needs to perform its task. For example. A boss (the calling, or caller) asks a worker(called function) to perform a task and treturn the results when the task is done. The worker may call other worker functions, and the boss will be unaware of this situation.



In above figure, shows the boss function communicating with several worker functions in a hierarchical manner. Note that worker1 acts as a "boss" function to worker4 and worker5, and worker 4 and worker5 report back to worker1. Relationship among functions may be other than the hierarchical structure shown in above.

Functions are invoked by writing the name of the function followed by a left parenthesis, followed by the arguments of the function.

3. Programmer –Defined functions:

The programmer can write functions to define specific tasks that may be used at many points in a script. These functions are referred to as programmer-defined functions. All variables declared in function definitions are called as local variables. A function's parameter are also considered to be local variables. When a function is called, the arguments in the function call are assigned to the corresponding parameters in the function definition.

Another use for functions is "software reusability", with good function naming and definition. Javascript already provides function parseInt to convert a string to an integer and function parseFloat to convert a string to a floating point number. Another advantage of function is to avoid code repeating.

4. Function Definitions:

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, {...}.

Syntax: function **name of the function(parameters)**

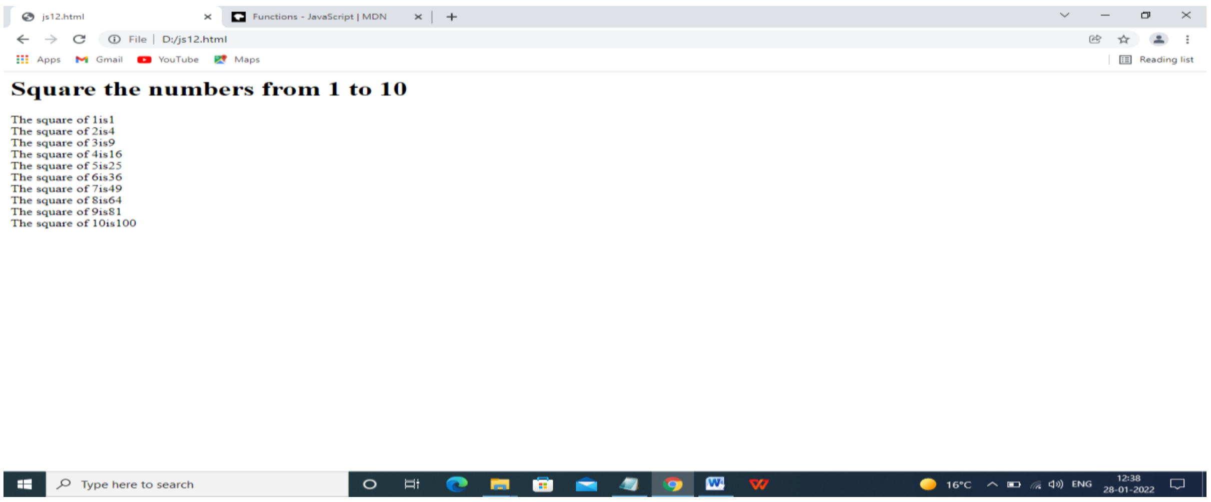
```
{
    Block of code;
}
```

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

EX: Consider a script that uses a function square to calculate the squares of the integers from 1 to10.

```
<html>
<head>
<script language="javascript">
document.writeln("<h1>Square the numbers from 1 to 10</h1>");
for(var x=1; x<=10;x++)
document.writeln("The square of " +x+ "is"+square(x)+"<br>");
function square(y)
{
return y * y;
}
</script>
</head><body></body></html>.
```

Output:



```
Square the numbers from 1 to 10

The square of 1is1
The square of 2is4
The square of 3is9
The square of 4is16
The square of 5is25
The square of 6is36
The square of 7is49
The square of 8is64
The square of 9is81
The square of 10is100
```

The for statement in lines (5 to 6) outputs HTML displays the results of squaring the integers from 1 to 10. Function square is invoked or called in line 6 with the expression square(x). when program control reaches this expression , the program calls function square(x) lines (7 to 10).

Function square(x) receives the copy of the value of x and stores it in the parameter y. Then square calculates y * y, The result is passed back to the point in line 6 where square was invoked. The return statement in square passes the result of the calculation y * y back to the calling function . Note that javascript keyword var is not used to declare variables in the parameter list of a function.

5. Random-number generation:

The element of chance can be introduced through the Math object's random method. Consider the following statement:

var randomValue = Math.random();

Method random generates a floating-point value from 0.0 up to, but not ncluding 1.o. If random truly produces values at random, then every value ftom 0.0 up to, but not including, 1.0 has an equal chance (or probability) of being chosen each time random is called.

The range of values produced directly by random is often different than what is needed in a specific application. For example, a program that simulates rolling a six-sided die would require random integers in the range from 1 to 6.

Ex:

```
<html>
  <head>
    <script language="javascript">
      var value;
      document.writeln("<table border= \"1\" width=\"50%\">");
      document.writeln(""<caption>random numbers</caption><tr>");
      for (var i=1;i<=20;i++)
   {
      value=Math.floor(1+Math.random() * 6);
      document.writeln("<td>"+value+"</td>");
     if(i % 5 == 0 &&  i != 20)
     document.writeln("</tr><tr>");
    }
      document.writeln("</tr></table>");
  </script>
   </head>
  <body></body></html>
```

Output:

In above program, use the multiplication operator (*) with random as follows:

Math.floor(1 + Math.random() * 6)

First, the preceding expression multiples the result of a call to Math.random() by 6 to produce a number in the range 0.0 up to, but not including, 6.0. This is called scaling the range of the random numbers. The number 6 is called the **scaling factor.** Next, we add 1 to the result to shift the range of numbers to produce a number in the range 1.0 up to, but not including 7.0.

Finally the method, Math.floor to round the result down to the closestinteger value in the range 1 to 6. Math method floor rounds its floating-point number argument to the closest integer not greater than the argument's value- for example 1.75 is rounded to 1, and -1.25 isrounded to -2.

6. **Random image generator:**

Here, we build a random number generator, a script that displays a randomly selected image every time the page that contains the script is loaded.

Ex:

```
 <html>
  <head>
    <script language="javascript">
     document.write("<img src= \ "" +
                          Math.floor( 1 + Math.random() * 7 ) +
                          " .gif\" width= \"105\" height=\" 100\" />");
    </script>
</head>
<body>
 <p>click refresh to run the script again</p>
</body>
</html>
```

Output:

In above program , for the script to function properly, the directory containing the file Randompicture.html must also contain seven images with integer file names(i.e,1.gif,2.gif,……..7.gif). The web page containing this script displays one of these seven images, selected at random, each time the page loads.

The statement document.write creates an image tag in the web page with the src attribute set to a random integer from 1 to 7. We use the Math.random method, a shifting value of 1 and scaling factor of 7. Thus, the script dynamically sets the source of the image tag to the name of one of the image files in the current directory.

7. Scope Rules:

Scope determines the accessibility of variables. We use identifiers as variable names. Each identifier in a program has a scope. The scope of an identifier for a variable or function is the portion of the program in which the identifier can be referenced.

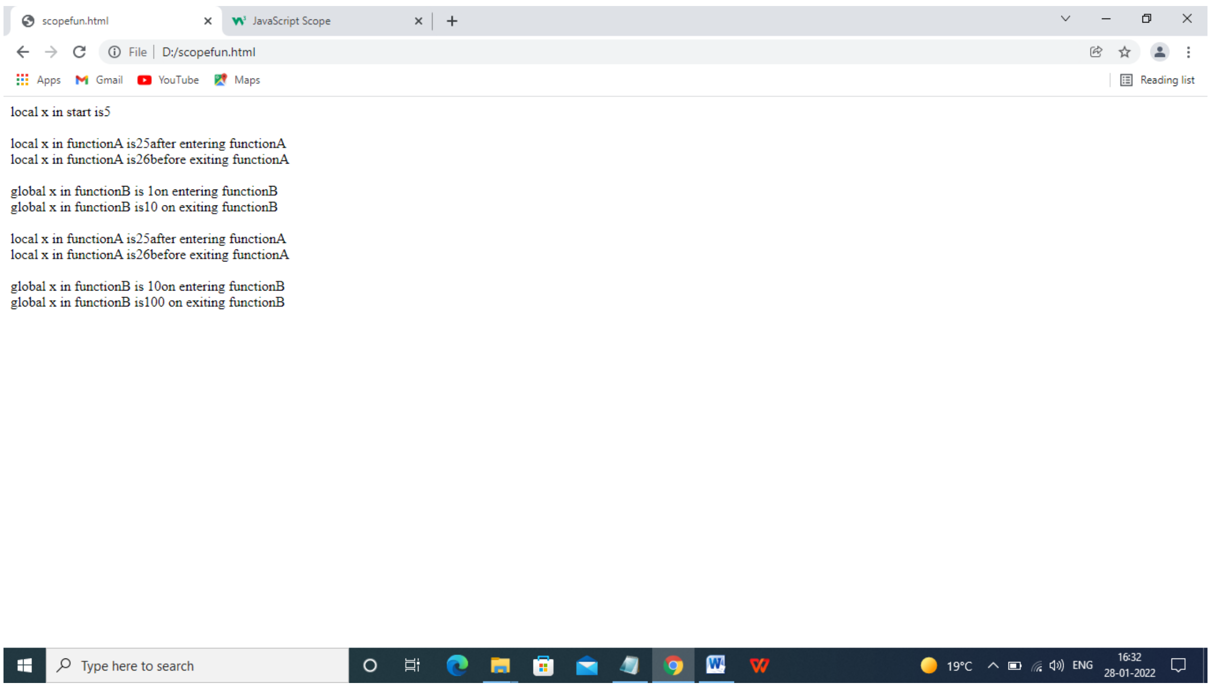
There are two types of scope of a variable in javascript. First one is **GLOBAL** scope and the next one is **LOCAL** scope.

The global variables or script-level variables are accessible in any part of a script and are said to have global scope. Identifiers declared inside a function have function scope and can be used only in that function. Function scope begins with the opening left brace ({) of the function in which the identifier is declared and ends at the terminating right brace(}) of the function. If a local variable in a function has the same name as a global variable, the global variable is “hidden” from the body of the function.

Ex:

```
<html>
<head>
<script language=”javascript”>
var x=1;
function start()
{
 var x=5;
Document.writeln(“ local x in start is” +x);
functionA();
functionB();
functionA();
functionB();
}
function functionA()
{
var x=25;
document.writeln(“ local x in functionA is “ +x+”after entering functionA”);
++x;
document.writeln(“ local x in functionA is “ +x+”before exiting functionA”);
}
function functionB()
{
document.writeln(“ global x in functionB is “ +x+” onentering functionB”);
x *= 10;
document.writeln(“ global x in functionB is “ +x+” on exiting functionB”);
}
</script>
<body onload=”start()”></body>
</html>
```

Output:



```
local x in start is5

local x in functionA is25after entering functionA
local x in functionA is26before exiting functionA

global x in functionB is 1on entering functionB
global x in functionB is10 on exiting functionB

local x in functionA is25after entering functionA
local x in functionA is26before exiting functionA

global x in functionB is 10on entering functionB
global x in functionB is100 on exiting functionB
```

Global variable x (line 4) is declared and initialized to 1. Function start() (line 5)declares a local variable (line 7) and it initializes to 5. The script defines two other functions functionA and function- that each take no arguments and return nothing. Each function is called twice from function start.

Function functionA defines local variable x (line 16) and initializes it to 25. The variable is incremented and output in a line of HTML text again before the function is exited.Function function does not declare any variables. When function is called, the global variable is output in a line of HTML text, multiplied by 10 and output in a line of HTML text again before the function is exited.

8. Javascript Global functions:

In javascript seven functions that are available globally. The global functions are summarized in below table.

GLOBAL FUNCTION	DESCRIPTION
escape	This function takes a string argument and returns a string in which all spaces,punctuation, accent characters and any other character that is not in the ASCII character set are encoded in a Hexadecimal format.
eval	This function allows javascript code to be stored as strings and executed dynamically.
isNaN	This function takes a numeric argument and returns true if the value of the argument is not a number; otherwise, it returns false.
parseFloat	This function takes a string argument and convert the beginning of the string into a floating-point value.
parseInt	This function takes a string argument and convert the beginning of the string into a integer value

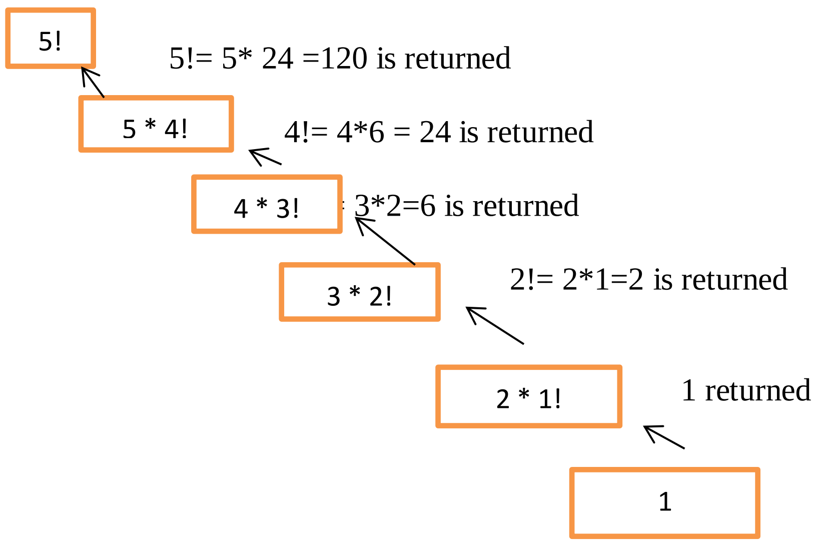
Actually , the global functions are all part of javascript GLOBAL object. The global object contains all the global variables in the script. Because the global functions and user defined functions are part of the global object.

9. RECURSION:

A recursive function is a function that calls itself, either directly or indirectly through another function. Recursive problem-solving approaches have a number of elements in common. A recursive function is called to solve a problem.

The below example shows, recursion to calculate and print the factorials of the integer 0 to 10.

Procession of recursive calls

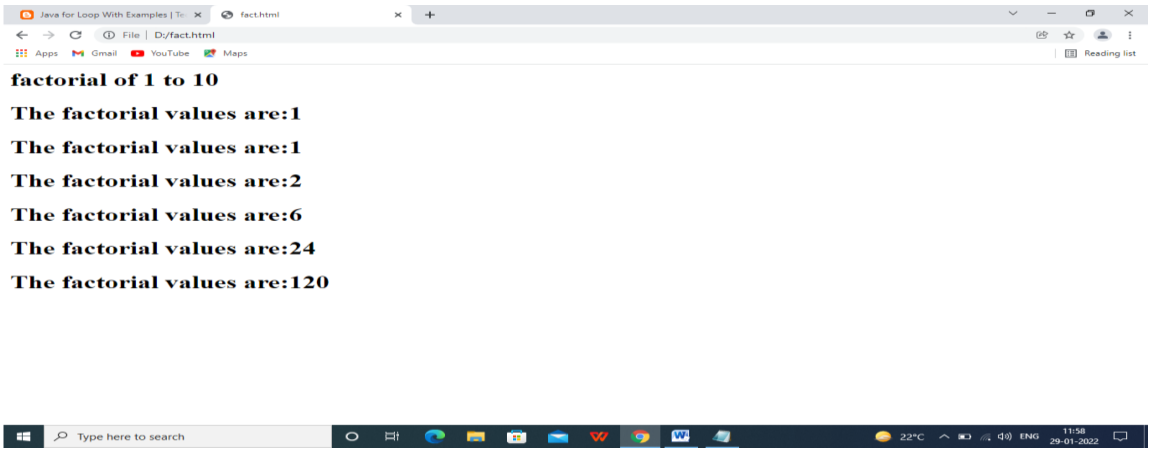


Ex:

```
1.  <html>
2.  <head>
3.  <script language="javascript">
4.  document.writeln("<h1> factorial of 1 to 10 </h1>");
5.  for( var i=0; i<=10;i++)
6.  document.writeln("<h1> The factorial values are:" +factorial(i) +"</h1>");
7.  function factorial(number)
8.  {
9.  If(number <= 1)
10. Return 1;
11. Else
12. Return number *factorial (number -1);
13. }
14. </script>
15. </head><body></body></html>
```

In this example , the recursive function factorial first tests (line 9) to see if a terminating condition is true. If number is indeed less than or equal to 1. Factorial returns 1. If number greater than 1, line 12 expresses the problem as the product of number and the value returned by a recursive call to factorial evaluating the factorial of number -1.

Output:



```
factorial of 1 to 10
The factorial values are:1
The factorial values are:1
The factorial values are:2
The factorial values are:6
The factorial values are:24
The factorial values are:120
```

10. Recursion Vs Iteration:

Both iteration and recursion are based on a control statement: iteration uses a repetition statement; recursion uses a selection statement. Both involve repetition. When comparing to iteration , recursion has many negatives.

Property	Recursion	Iteration
Definition	Function calls itself.	A set of instructions repeatedly executed.
Application	For functions.	For loops.
Termination	Through base case, where there will be no function call.	When the termination condition for the iterator ceases to be satisfied.
Usage	Used when code size needs to be small, and time complexity is not an issue.	Used when time complexity needs to be balanced against an expanded code size.
Code Size	Smaller code size	Larger Code Size.
Time Complexity	Very high(generally exponential) time complexity.	Relatively lower time complexity(generally polynomial-logarithmic).

JAVACSRIPT ARRAYS

1. Introduction:

Arrays are data structures consisting of related data items(collection of data items). Javascript arrays are dynamic entities in that they can change size after they are created.

2. Arrays:

An array is a group of memory locations that all have the same name and normally are of the same type. The name of the array followed by the position number of the element in square brackets[]. The first element in every array is the zeroth element. The position number in square brackets is called a subscript or index.

Syntax: var arrayname= list of elements;

Ex : var c={01,2,3,4,5,6,……};

In above example, the first element of array is referred to as c[0], the second element of array c is referred to as c[1], the seventh element of array c is referred to as c[6] and, in general, the ith element of array c is referred to as c[i-1].

An array with 12 elements as shown below:

```
c[0]  c[1]  c[2]  c[3]  c[4]  c[5]  c[6]  c[7]  c[8]  c[9]  c[10]  c[11]
 45   |0     6     72    15    0     62    -3    1     78    4      9     ↓
      |
      └──────→  Name of array

                                                        Position number of the
                                                        Element within array c
```

Every array in javascript knows its own length. The length of an can be determined by using **length** property. The array name is **C.** The length of array C is 12 and is determined by the following expression:

c.length

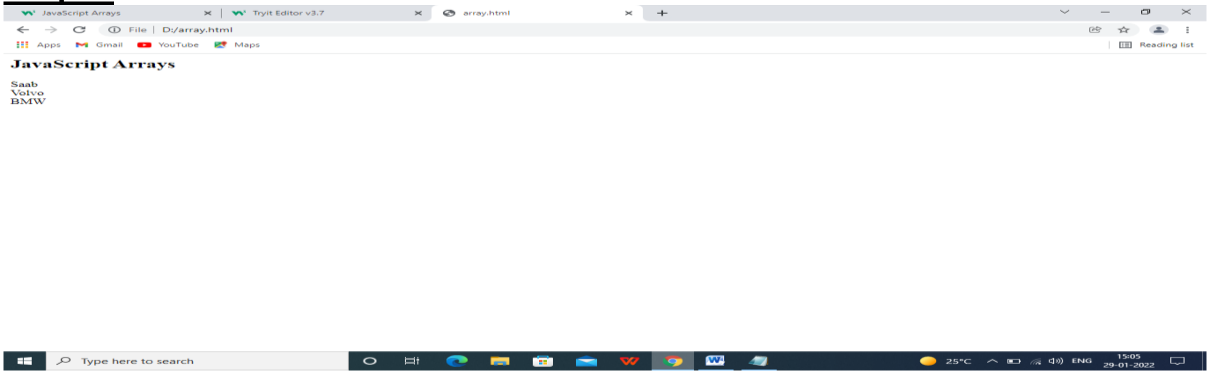
The array's 12 elements are referred to as c[0],c[1],c[2],………..c[11]. The value of c[0] is 45, the value of c[1] is 0,….. To calculate the sum of values contained in the first three elements of array c and store the result in variable sum, we would write:

Sum= c[0] + c[1] + c[2];

Example:

```
<html>
    <head></head>
       <body>
           <h2>JavaScript Arrays</h2>
           <script language="javascript">
           var cars = ["Saab", "Volvo", "BMW"];
          document.writeln(cars[0] +"<br>"+ cars[1]+ "<br>" +cars[2]);
           </script></body></html>
```

Output:



```
JavaScript Arrays

Saab
Volvo
BMW
```

3. Declaring and Allocating Arrays:

Arrays occupy space in memory . Actually an array in java script is an array object. The programmer uses **new** operator to allocate dynamically the number of elements required by each array. Operator **new** creates an object as the program executes by obtaining enough memory to store an object of the type specified to the right of new.

The process of creating new objects is also known as **instantiating an object**, and the operator **new** is known as the dynamic memory allocation operator. Arrays are allocated with new because arrays are considered to be objects, and all objects must be created with new.

Syntax: var arrayname=new Array();
Ex: var colors= new Array();

Ex:
```
  <html>
       <body>
          <script language=”javascript”>
            var i;
           var emp = new Array();
             emp[0] = "Arun";
             emp[1] = "Varun";
             emp[2] = "John";

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

Output:
```
 Arun
Varun
John
```

4. Examples using Arrays:

❖ **Creating and initializing Arrays:**

Ex: 1
```
 <html>
      <body>
           <script>
              var emp=new Array("Jai","Vijay","Smith");
              for (i=0;i<emp.length;i++)
            {
             document.write(emp[i] + "<br>");
            }
</script>
</body>
</html>
```

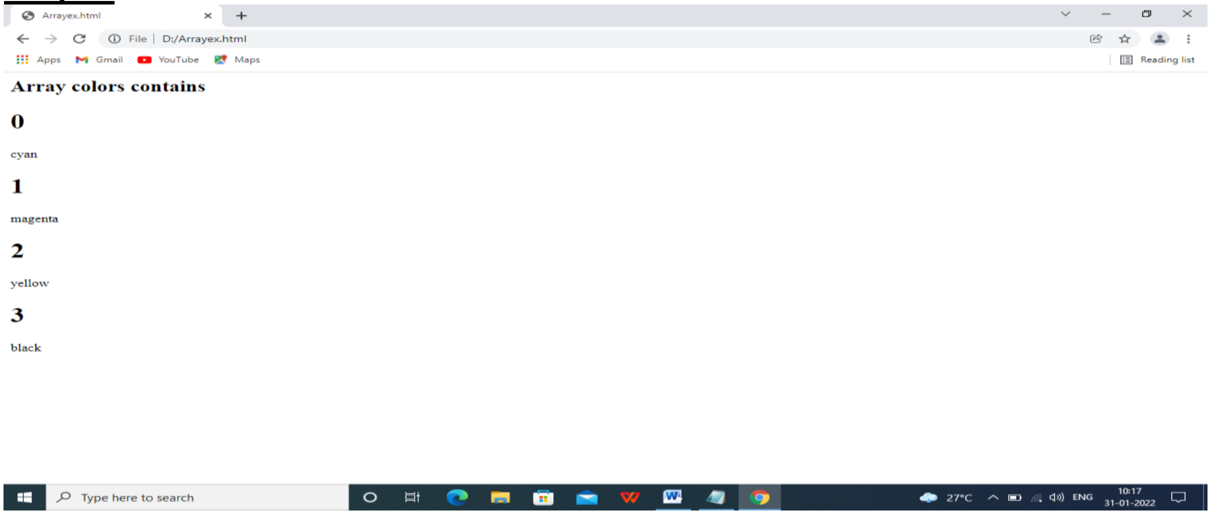
<u>Output:</u>

```
Jai
Vijay
Smith
```

<u>Ex: 2:</u>

```
1.<html>
 2.<head>
  3.<script language="javascript">
    4. function start()
   5.{
     6.   var colors=new Array("cyan","magenta","yellow","black");
       7.outputArray(" Array colors contains",colors);
   8.}
    9.function outputArray(header,theArray)
   10.{
      11. document.writeln("<h2>"+header+"</h2>");
       12.for(var i=0; i<theArray.length;i++)
       13.document.writeln("<h1>"+i+"</h1>"+theArray[i]);
   14.}
     15.</script>
 16.</head>
    17.<body onload="start()"></body>
  18.</html>
```

<u>Output:</u>



```
Array colors contains
0
cyan
1
magenta
2
yellow
3
black
```

In above example , the script creates one Array object to demonstrate initializing arrays with initializer lists (line 6), and displays each array in an HTMLdocument using the same function outputArray. When the web page loads,the script calls function start() in response to the <body>'s onload event.

5. Random Image generator using arrays:

We created a random image generator that required image files to be named1.jp,2.jpg….. Here we uses an array pictures to store the names of the image files as strings. The script generates a random integer and uses it as a subscript into the pictures array. The script outputs an HTML img element whose src attribute contains the image file name located in the randomly selected position in the pictures array

Ex:

```
1.<html>
 2.<head>
  3. <script language="javascript">
4.var pictures=["flower1","flower2","flower1","flower2"];
5. document.write("<img src=\""
                    +pictures[Math.floor(Math.random()*4)]+
                     ".gif\"width=\"105\" height=\"100\" />");
6.</script>
7.</head>
8.<body>
9.<p>click refreshto run the scriopt again</p>
10.</body>
11.</html>.
```

The script declares an Array pictures in line 4 and initializes it with the names of four image files. Line5 create the image tag that displays the random image on the webpage. Lne 5 opens the img tag and begins the src attribute. Line 5 generates a random integer from 0 to 3 as an index into the pictures array, the result of which is a randomly selected image file name.Line 5 completes the img tag with the extension of the image file and the width and height of the image.

6. Passing Arrays to Functions:

To pass an array argument to a function, specify the name of the array without brackets. Although entire arrays are passed by reference, individual numeric and Boolean array elements are passed by value exactly as simple numeric and Boolean variables are passed. Such simple single pieces of data are called scalars, or scalar quantities. To pass an array element to a function, use the subscripted name of the element as an argument in the function call.

Ex:

```
   1. <html>
2.<head>
3.<script language="javascript">
4.function start()
5.{
  6.var a=[1,2,3,4,5];
  7.outputArray(" The values of original array are:",a);
  8.modifyArray(a);
  9.outputArray("The values of modified array are:",a);
10.}
11.function outputArray(header,theArray)
12.{
 13.document.writeln(header +theArray.join(" ")+"<br>");
14.}
15,function modifyArray(theArray)
16.{
 17. for(var j in theArray)
   18.theArray[j] *= 2;
19.}
20.</script>
21.</head><body onload="start()"></body>
22.</html>
```

Output:

```
The values of original array are:1 2 3 4 5
The values of modified array are:2 4 6 8 10
```

The statements in line 7 invokes function outputArray to dislay the contents of array a before it is modified. Function outtputArray (11-14 lines) receives a string to output and the array to output. The stataement in lines 12-13 uses array method join() to create a string containing all the elements in theArray.

Line 8 invokes function modifyArray(lines 15-19) and passes it array a. The modifyArray function multiplies each element by 2. As the screen capture shows, the elements of a are indeed modified by modifyArray.

7. Sorting Arrays:

Sorting data is one of the most important computing functions.The Array object in javascript has a built –in method called **“sort()”** for sorting arrays.

By default sort method uses string comparisions to determine the sorting order of the array element. The Method sort takes as its optional argument the name of a function that compares its two arguments and returns one of the following:

- A negative value if the first argument is less than the second argument.
- Zero, if the arguments are equal.
- A positive value f the first argument is greater than the second argument.

Ex:

```
1.<html>
2.<head></head>
3.<body>
4.<h2>JavaScript Array Sort</h2>
5.<p>The sort() method sorts an array alphabetically:</p>
6.<script language="javascript">
7.var fruits = ["Banana", "Orange", "Apple", "Mango"];
8.document.writeln("before sorting the fruits are:" +fruits+"<br>");
9.fruits.sort();
10.document.writeln("After sorting the fruits are:"+fruits);
11.</script>
12.</body>
13.</html>
```

Output:

JavaScript Array Sort

The sort() method sorts an array alphabetically:

```
before sorting the fruits are:Banana,Orange,Apple,Mango
After sorting the fruits are:Apple,Banana,Mango,Orange
```

In above example, use array object in line 7. In line 8 display the order before sorting array elements. In line 9 uses a sort method fruits.sort(). Next displays the array objects after sorting.

8. Searching Arrays: Linear search and Binary search:

Often, a programmer will be working with large amounts of data stored in arrays. It may be necessary to determine whether an array contains a value that matches a certain **key value.** The process of locating a particular element value in an array is called SEARCHING. Two searching techniques are there in javascript.---the simple LINEAR SEARCH technique and the more efficient BINARY SEARCH technique.

- **Linear Search:**

Linear search is used to search a key element from multiple elements. Linear search is less used today because it is slower than binary search and hashing.

Algorithm:

- Step 1: Traverse the array
- Step 2: Match the key element with array element
- Step 3: If key element is found, return the index position of the array element
- Step 4: If key element is not found, return -1

- **Binary search:**

Binary Search is searching technique which works on Divide and Conquer approach. It used to search any element in a sorted array. As compared to linear, binary search is much faster with Time Complexity of O(logN) whereas linear search algorithm works in O(N) time complexity.Given a sorted array of numbers. The task is to search a given element in the array using Binary search.

Examples:
```
Input : arr[] = {1, 3, 5, 7, 8, 9}
x = 5
Output : Element found!

Input : arr[] = {1, 3, 5, 7, 8, 9}
x = 6
Output : Element not found!
```

Algorithm:

1. *BASE CONDITION:* If starting index is greater than ending index return false.
2. Compute the middle index.
3. Compare the middle element with number x. If equal return true.
4. If greater, call the same function with ending index = middle-1 and repeat step 1.

5. If smaller, call the same function with starting index = middle+1 and repeat step 1.

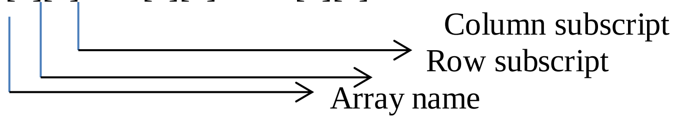
Basis of comparison	**Linear search**	**Binary search**
Definition	The linear search starts searching from the first element and compares each element with a searched element till the element is not found.	It finds the position of the searched element by finding the middle element of the array.
Sorted data	In a linear search, the elements don't need to be arranged in sorted order.	The pre-condition for the binary search is that the elements must be arranged in a sorted order.
Implementation	The linear search can be implemented on any linear data structure such as an array, linked list, etc.	The implementation of binary search is limited as it can be implemented only on those data structures that have two-way traversal.
Approach	It is based on the sequential approach.	It is based on the divide and conquer approach.
Size	It is preferrable for the small-sized data sets.	It is preferrable for the large-size data sets.
Efficiency	It is less efficient in the case of large-size data sets.	It is more efficient in the case of large-size data sets.
Worst-case scenario	In a linear search, the worst- case scenario for finding the element is O(n).	In a binary search, the worst-case scenario for finding the element is $O(\log_2 n)$.
Best-case scenario	In a linear search, the best-case scenario for finding the first element in the list is O(1).	In a binary search, the best-case scenario for finding the first element in the list is O(1).
Dimensional array	It can be implemented on both a single and multidimensional array.	It can be implemented only on a multidimensional array.

9. <u>Multi-Dimensional Arrays:</u>

Multidimensional arrays with two subscripts often are used to represent **tables** of values consisting of information arranged in rows and columns. Arrays that require two subscripts to identify a particular element called **two-dimensional arrays.** Javascript does not support multidimensional arrays directly.

Below representation shows two- dimensional array with three rows and four columns.

	column 0	column 1	column 2	column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]



Every element in array a is identified by an element name of the form a[i][j]; a is name of the array, and I and j are the subscripts that uniquely identify the row and coloumn, of each element in a.

❖ **Creating two-dimensional arrays:**

A multidimensional array in which each row has a different number of columns can be allocated dynamically, as follows:

```
var b;
  b=new Array(2); //allocating rows
  b[0]=new Array(5); //allocate column for row 0
  b[1]=new Array(3); //allocate column for row 1
```

Ex:

```
<html>
 <head></head>
 <body>
    <script language="javascript">
      var emp1=new Array(3);
      for(var i=0;i<3;i++)
{
   emp1[i]=new Array(3);
}
 emp1[0][0]=1;
 emp1[0][1]=2;
 emp1[0][2]=3;

emp1[1][0]=4;
 emp1[1][1]=5;
 emp1[1][2]=6;

 emp1[2][0]=7;
 emp1[2][1]=8;
 emp1[2][2]=9;
for(var i=0;i<3;i++)
{
   for(var j=0;j<3;j++)
{
  document.write(emp1[i][j] + " &nbsp");
}
  document.write("<br>");
}
</script>
</body>
</html>
```

Output:

```
 1 2 3
 4 5 6
 7 8 9
```

JAVA SCRIPT OBJECTS

1. Introduction:

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc. JavaScript is an object-based language. Everything is an object in JavaScript. JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.

2. Introduction to Object Technology:

There are 2 ways to create objects.

1. By object literal
2. By creating instance of Object directly (using new keyword)

1) JavaScript Object by object literal:

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

```
1. object={property1:value1,property2:value2.....propertyN:valueN}
```

As you can see, property and value is separated by : (colon).Let’s see the simple example of creating object in JavaScript.

```
Ex: <html>
     <body>
     <script language=”javascript”>
     emp={id:102,name:"Shyam Kumar",salary:40000}
     document.write(emp.id+" "+emp.name+" "+emp.salary);
     </script>
     </body>
     </html>
```

Output:

```
102 Shyam Kumar 40000
```

2. By creating instance of Object:

The syntax of creating object directly is given below:

```
1. var objectname=new Object();
```

Here, **new keyword** is used to create object.Let’s see the example of creating object directly.

Ex:
```
<html>
        <body>
           <script>
           var emp=new Object();
          emp.id=101;
          emp.name="Ravi Malik";
          emp.salary=50000;
          document.write(emp.id+" "+emp.name+" "+emp.salary);
</script> </body></html>
```

Output:

```
101vi Malik 50000
```

Math object:

The Math object's methods allow the programmer to perform many common mathematical calculations. An object's methods are called by writing the name of the object followed by a dot (**.**) and the name of the method. In parenthesis following the method name is the argument to the method.

Some Math object methods are shown in below table:

METHOD	DESCRIPTION	EXAMPLE
abs(x)	Absolute value of x	abs(7.2) is 7.2 abs(-5.6) is 5.6
ceil(x)	Rounds x to the smallest integer not less than x	ceil(9.2) is 10.0 ceil(-9.8) is -9.0
floor(x)	Rounds x to the largest integer not greater than x	floor(9.2) is 9.0 floor(-9.8) is 10.0
log(x)	Natural logarithm of x (base e)	log(2.718282)
max(x, y)	Larger value of x and y	max(2.3,12.7) is 12.7
min (x, y)	Smaller value of x and y	min(-2.3,-12.7) is -12.7
pow(x , y)	x raised to power y	pow (2.0,4.0) is 16
sqrt(x)	Square root of x	sqrt (900.0) is 30.0

Ex:

```
    <html>
   <body>
   <script language="javascript">
     document.writeln("The square root is:"+Math.sqrt(16.0)+"<br>");
    document.writeln("The minimum value is:"+Math.min(16.0,5.6)+"<br>");
    document.writeln("The maximum value  is:"+Math.max(16.0,5.6)+"<br>");
    document.writeln("The rounded value is:"+Math.ceil(-16.8)+"<br>");
    document.writeln("The floor value root is:"+Math.floor(16.8)+"<br>");
    document.writeln("The power value  is:"+Math.pow(2,4));
  </script>
  </body>
</html>
```

Output:

```
The square root is:4
The minimum value is:5.6
The maximum value is:16
The rounded value is:-16
The floor value root is:16
The power value is:16
```

3. **Date object:**

Javascript date object provides methods for date and time manipulations. Date and time processing can be performed based on the computer's local time zone or based on World Standard's Coordinated Universal Time (UTC)-formerly called Greenwich Mean Time (GMT).

The methods of the Date object are summarized in below:

METHOD	DESCRIPTION
getDate()	Returns a number from1 to 31 representing the day of the month in local time
getDay()	Returns a number from 0(Sunday) to 6(Saturday) representing the day of the week in local time
getFullYear()	Returns the year as a four-digit number in local time
getHours()	Returns a number from 0 to 23 representing hours since midnight in local time
getMilliseconds()	Returns a number from 0 to 999 representing the number of milliseconds in local time.(1 sec=1000 milliseconds)
getMinutes()	Returns a number from 0 to 59 representing the minutes for the time in local time.
getMonth()	Returns a number from 0(January) to 11(December) representing the month in local time
getSeconds()	Returns a number from 0 to 59 represenmting the seconds for the time in local time.
getTime()	Returns the number of milliseconds between January 1, 1970 and the time in the Date object.
toLocaleString()	Returns a string representation of the date and time in a form specific to the computer's locale.

Ex:

```
<html>
   <body>
      <script language="javascript">
        var current=new Date()
        document.writeln("The current date and time
                                    is:"+current.toLocaleString()+"<br>");
       document.writeln("The current date is:"+current.getDate()+"<br>");
       document.writeln("The current month is:"+current.getMonth()+"<br>");
       document.writeln("The current year is:"+current.getFullYear()+"<br>");
       document.writeln("The current second is:"+current.getSeconds()+"<br>");
       document.writeln("The current minute is:"+current.getMinutes()+"<br>");
       document.writeln("The current hour is:"+current.getHours()+"<br>");
       document.writeln("The current day  is:"+current.getDay()+"<br>");
   </script>
   </body>
</html>
```

Output:

```
The current date and time is:2/4/2022, 12:09:13 PM
Thecurrentdateis:4
Thecurrentmontis:1
Thecurrentyearis:2022
Thecurrentsecondis:13
Thecurrentminuteis:9
Thecurrenthouris:12
The current day is:5
```

4. Boolean and Number object:

- **Boolean object:**

Java script provides the Boolean and number objects as object wrappers for Boolean true/false values and numbers.. These wrappers define methods and properties useful in manipulating Boolean values and numbers.

When a javascript program requires a Boolean value, javascript automatically creates a boolean object to store the value. Java script programmers can create Boolean objects explicitly with the statement.

var b=new Boolean(booleanvlue);

The constructor argument Boolean value specifies whether the value of the Boolean object should be true or false. If booleanvalue is false, 0, null, Number ,NaN or an empty string(“”), or if no argument is supplied, the new Boolean object contains false. Otherwise the new Boolean object contains true.

METHOD	DESCRIPTION
toString()	Returns the string ‘true’ if the value of the Boolean object is true ; otherwise , returns the string ‘false”
valueOf()	Returns the value true if the Boolean object is true; otherwise, returns false.

Ex:

```
 <html>
      <body>
          <script language="javascript">
            var b=new  Boolean(true);
            var b1=new Boolean(false);
               document.writeln("The boolean value is:" +b.toString()+"<br>");
               document.writeln("The boolean value of is:" +b1.valueOf());
     </script>
     </body>
   </html>
```

Output:

```
Thebooleanvalueis:true
The boolean value of is:false
```

- **Number object:**

  Javascript automatically creates Number objects to store numeric values in a javascript program. Javascript programmers can create a Number object with the statement.

  var n= new Number(numeric value);

  The constructor argument numeric value is the number to store in the object. The methods or properties of number objects are summarized below:

METHOD or PROPERTY	DESCRIPTION
toString(radix)	Returns the string representation of the number. The optional radix argument specifies the number’s base.For example, radix 2 results in the binary

	representation of the number.
valueOf()	Returns the numeric value
Number.MAX_VALUE	This property represents the largest value that can be stored in a javascript program-approximately 1.79E+308
Number.MIN_VALUE	This property represents the smallest value that can be stored in a javascript program-approximately 2.22E-308
Number.NaN	This property represents not a number- avalue returned from an arithemetic exression that does not result in anumber(eg.. the expression parseInt("hello") cannot convert the string hello into a number, so parseInt would returnNumber.NaN.

Ex:

```
 <html>
<head></head>
<body>
  <script language="javascript">
     var n= new Number(123);
   document.writeln(" The number object method is:"+n.toString()+"<br>");
  document.writeln(" The number object method is:"+n.valueOf()+"<br>");
  document.writeln("The number object method
                              is:"+Number.MAX_VALUE+"<br>");
 document.writeln(" The number object method is:"+Number.MIN_VALUE+"<br>");
 document.writeln(" The number object method is:"+Number.NaN);
</script>
</body>
</html>
```

Output:

```
Thenumberobjectmethodis:123
Thenumberobjectmethodis:123
Thenumberobjectmethodis:1.7976931348623157e+308
Thenumberobjectmethodis:5e-324
The number object method is:NaN
```

5. **Document object:**

Javascript provides the document object for manipulating the document that is currently visible in the browser window. The document object has many useful properties and methods, such as document.write and document.writeln, which have both been used in prior javascript examples.

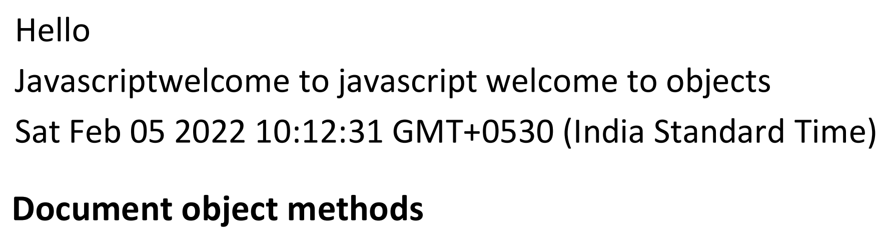
A more comprehensive list of properties and metjods can be found at the javascript shown below:

METHOD or PROPERTY	DESCRIPTION
write(string)	Writes the string to the HTML document as HTML code
writeln(string)	Writes the string to the HTML document as HTML code and adds a new line character at the end.
document.lastModified	This property is the date and time that this document was last modified.

Ex:

```
 <html>
  <head>
    <script language="javascript">
          document.writeln("Hello<br>");
          document.write("Javascript");
          document.writeln("welcome to javascript");
          document.writeln("welcome to objects<br>");
           var date=new Date(document.lastModified);
          document.writeln(date);
   </script>
</head>
  <body>
    <h1> Document object methods</h1>
</body>
</html>
```

Output:



```
Hello
Javascriptwelcome to javascript welcome to objects
Sat Feb 05 2022 10:12:31 GMT+0530 (India Standard Time)

Document object methods
```

6. Window object:

Javascript's window object provides methods for manipulating browser windows. The following script shows many of the commonly used properties and methods of the window object and uses them to create an interesting website that spans multiple browser windows.

The below table shows a list of some commonly used methods and properties of the window object.

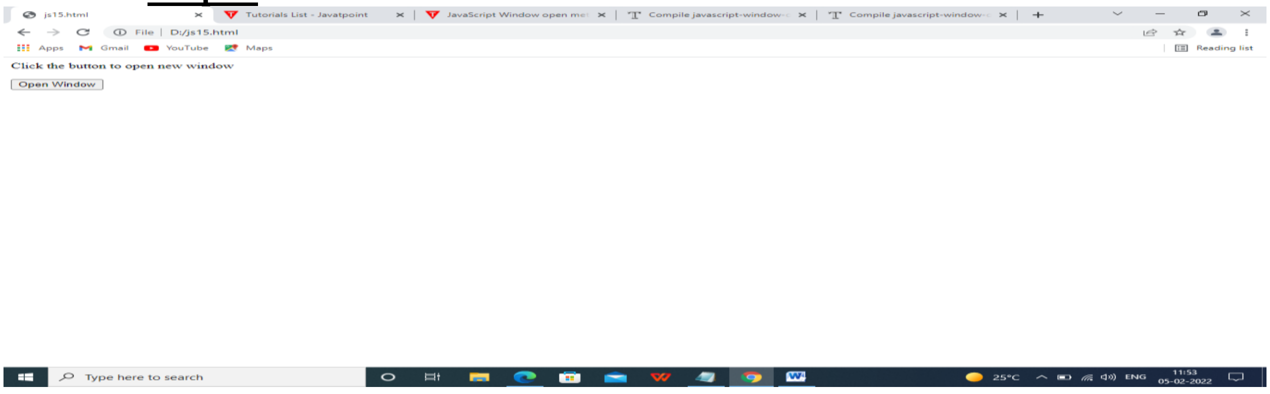
METHOD or PROPERTY	DESCRIPTION
open(url name,options)	Creates a new window with the URL of the window set to url, the name set to name, and the visible features set by the string passed in as option.
prompt(prompt,default)	Displays a dialog box asking the user for input. The txt of the dialog is prompt, and the default value is set to

	default.
close()	Closes the current window and delete its object from memory.
focus()	This method gives focus to the window.
blur()	This method takes focus away from the window.
window.document	This property contains the document object representing the document currently inside the window.
window.closed	This property contains a Boolean value that is set to true if the window is closed, and false if it is not.

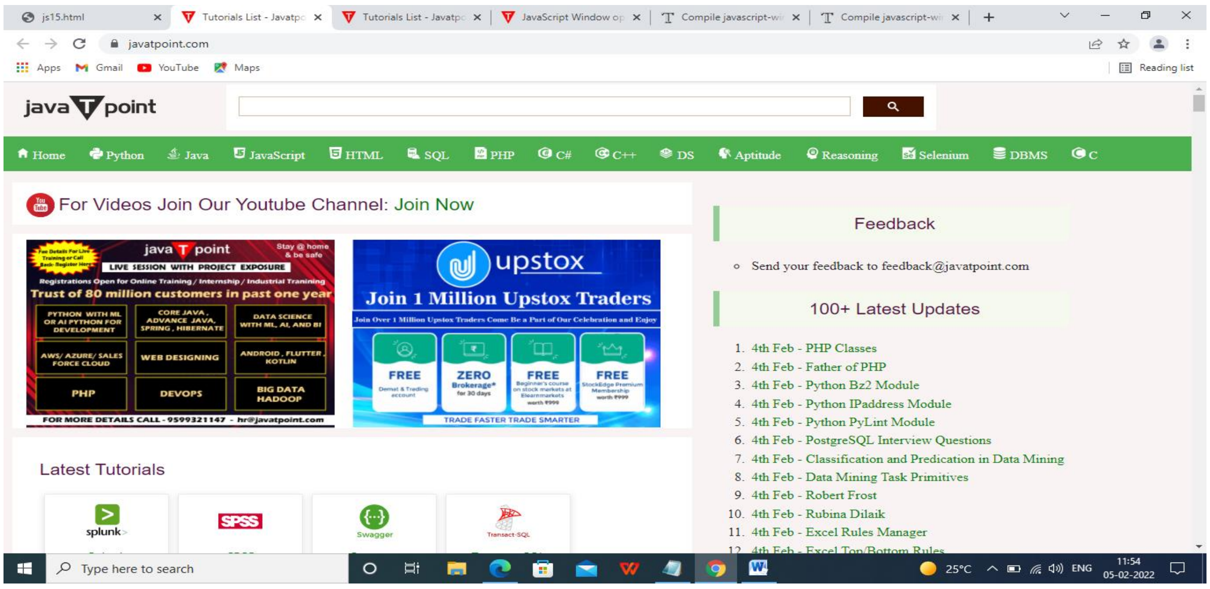
Ex 1:

```
<html>
<body>
Click the button to open new window <br><br>
<button onclick="window.open('https://www.javatpoint.com')"> Open Window
</button>

</body>
</html>
```

Output:



Click the button to open new window



For Videos Join Our Youtube Channel: Join Now

Feedback

- Send your feedback to feedback@javatpoint.com

100+ Latest Updates

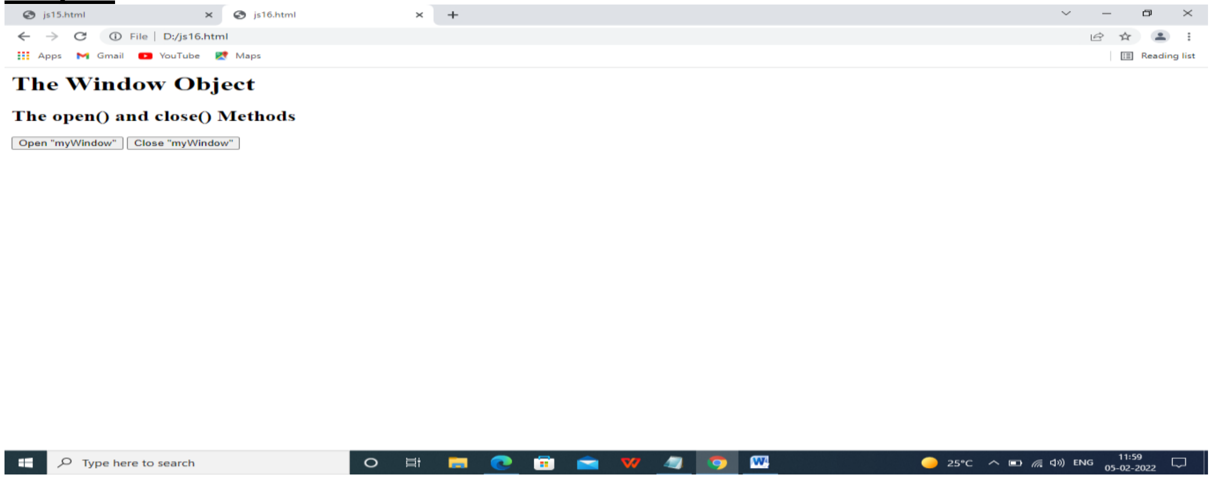
1. 4th Feb - PHP Classes
2. 4th Feb - Father of PHP
3. 4th Feb - Python Bz2 Module
4. 4th Feb - Python IPaddress Module
5. 4th Feb - Python PyLint Module
6. 4th Feb - PostgreSQL Interview Questions
7. 4th Feb - Classification and Predication in Data Mining
8. 4th Feb - Data Mining Task Primitives
9. 4th Feb - Robert Frost
10. 4th Feb - Rubina Dilaik
11. 4th Feb - Excel Rules Manager
12. 4th Feb - Excel Top/Bottom Rules

Latest Tutorials

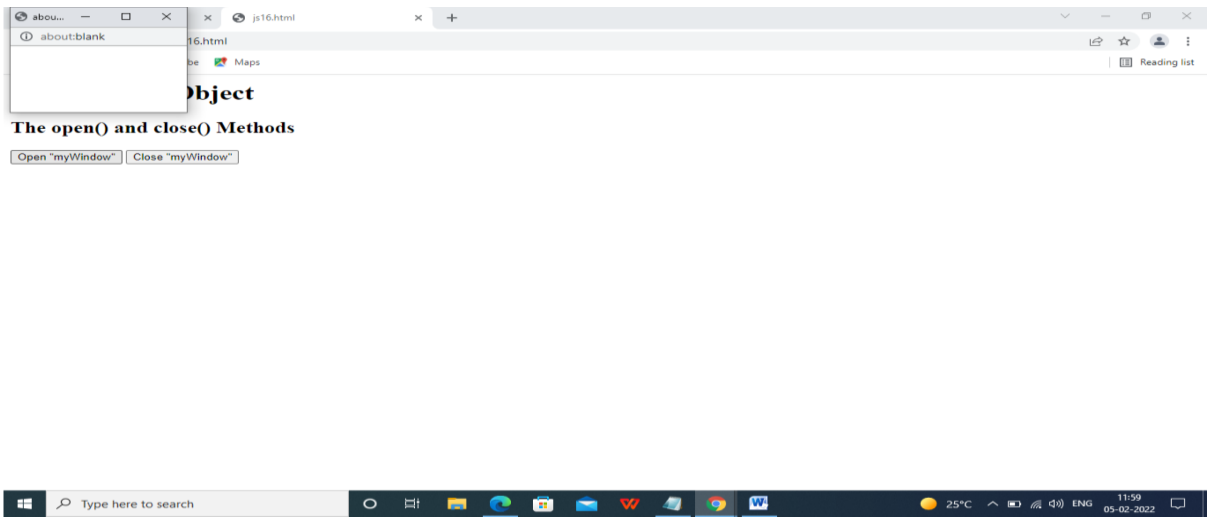
Ex 2:

```
<html>
<body>

<h1>The Window Object</h1>
<h2>The open() and close() Methods</h2>

<button onclick="openWin()">Open "myWindow"</button>
<button onclick="closeWin()">Close "myWindow"</button>

<script>
var myWindow;

function openWin() {
  myWindow = window.open("", "", "width=200,height=100");
}

function closeWin() {
  myWindow.close();
}
</script>
</body>
</html>
```

Output:



```
The Window Object
The open() and close() Methods
```



Object

The open() and close() Methods

Open "myWindow" Close "myWindow"

7. Using cookies:

Cookies provide web developers with a tool for personalizing web pages. A cookie is a piece of data that is stored on the user's computer to maintain information about the client during and between browser sessions. A website may store a cookie on the client's computer to record user preferences or other information that the web site can retrieve during the client's subsequent visits.

Cookies are accessible in javascript through the document object's "cookie" property. Javascript treats a cookie as a string of text. A cookie is a short text file that stores some data on a computer (about 4KB). They usually keep track of information such as preferencess for a website, prompting the user to improve the web page the next time they visit.

Using JavaScript, cookies can be created, retrieved, and modified directly, and the process is simple. The name, value and the length of the cookie can be restricted.

❖ Various types of cookies:

➢ There are three types of cookies:

1. **First Party Cookies** - These are cookies that are created by your website and can only be read by your website.
2. **Third-party cookies** - These cookies are produced by third-party advertising on your website. These cookies can only be read on any site that displays the same ad using the advertising code.
3. **Session cookies** - These cookies are saved on your browser. They are destroyed when the browser is closed.

❖ Creating a cookie:

You can make a cookie using the document.cookie property. In JavaScript, you can use this property to set up, read, and delete cookies.

Additionally, any cookies linked with the document are represented by this property. We create a cookie in the form of name=value using the document.cookie property.

You can only set one cookie at a time using this property.Take a look at the example below.

```
document.cookie = "UserName = fabuluosDesigns";
```

You must use an in-built javascript function called encodeURIComponent() to usespecial characters when creating cookies.

Before saving the cookie, this function encodes special characters like white spaces, semicolons, and others.

Take a look at the example below.

```
document.cookie = "UserName=" + encodeURIComponent("fabulous designs");
```

The cookie’s lifespan is limited to the duration of the current browser session, thus, it will be removed when the user quits the browser.

we created three functions: setCookie(), getCookie(), and checkCookie().

- **setCookie()** - Creates a cookie with an optional max-age attribute.
- **getCookie()** - This function, reads the value of a cookie.
- **checkCookie()** - Using getCookie(), this function checks whether the UserName is set or not. If set, it will display a greeting message. If it isn’t set, it will prompt the user to enter their user name and store it in the cookie using setCookie().

❖ Deleting a cookie:

To delete a cookie, simply rename it using the same name, specifying an empty value, orsetting its max-age attribute to 0.

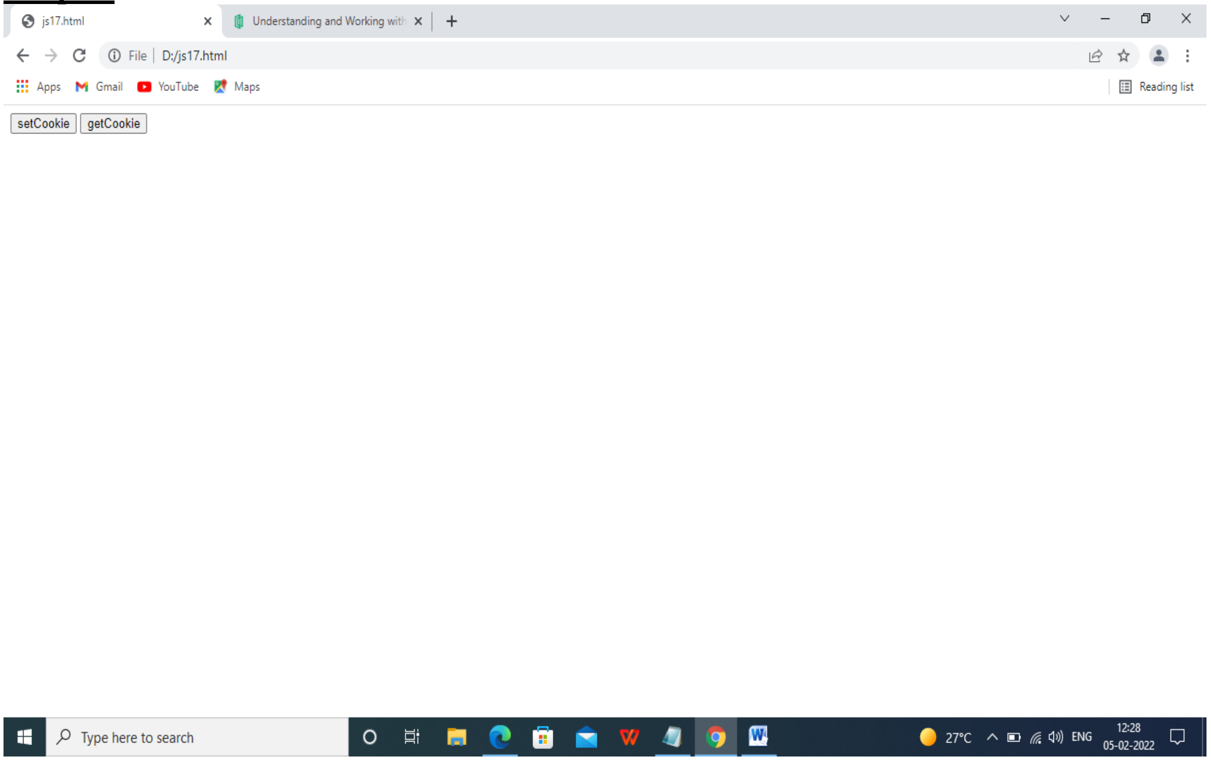
```
document.cookie = "UserName=; max-age=0";
```

Ex:

```
<html>
<head>
</head>
<body>
<input type="button" value="setCookie" onclick="setCookie()">
<input type="button" value="getCookie" onclick="getCookie()">
   <script language="javascript">
   function setCookie()
   {
      document.cookie="username=Duke Martin";
   }
   function getCookie()
   {
      if(document.cookie.length!=0)
      {
      alert(document.cookie);
```

```
        }
        else
        {
        alert("Cookie not available");
        }
    }
    </script>

</body>
</html>
```

Output:


js17.html
Understanding and Working with
File | D:/js17.html
Apps
Gmail
YouTube
Maps
Reading list
setCookie
getCookie
Type here to search
27°C
ENG
12:28
05-02-2022