3.1 Features of JavaScript

Following are some features of JavaScript

1. **Browser Support :** For running the JavaScript in the browser there is no need to use some plug-in. Almost all the popular browsers support JavaScripting.
2. **Structure Programming Syntax :** The Javascript supports much commonly the structured language type syntax. Similar to C-style the programming blocks can be written.
3. It automatically inserts the semicolon at the end of the statement, hence there is no need to write semicolon at the end of the statement in JavaScript.
4. **Dynamic Typing :** It supports dynamic typing, that means the data type is bound to the value and not to the variable. For example one can assign integer value to a variable say 'a' and later on we can assign some string value to the same variable in JavaScript.
5. **Run Time Evaluation :** Using the **eval** function the expression can be evaluated at run time.
6. **Support for Object :** JavaScript is object oriented scripting language. However handling of objects in JavaScript is somewhat different that the conventional object oriented programming languages. JavaScript has a small number of in-built objects.
7. **Regular Expression :** JavaScript supports use of regular expressions using which the text-pattern matching can be done. This feature can be used to validate the data on the web page before submitting it to the server.
8. **Function Programming :** In JavaScript functions are used. One function can accept another function as a parameter. Even, one function can be assigned to a variable just like some data type. The function can be run without giving the name.

3.2 JavaScript Syntax and Types

The JavaScript can be directly embedded within the HTML document or it can be stored as external file.

Directly embedded JavaScript

The syntax of directly embedding the JavaScript in the HTML is

```
<script type="text/javascript">
...
```

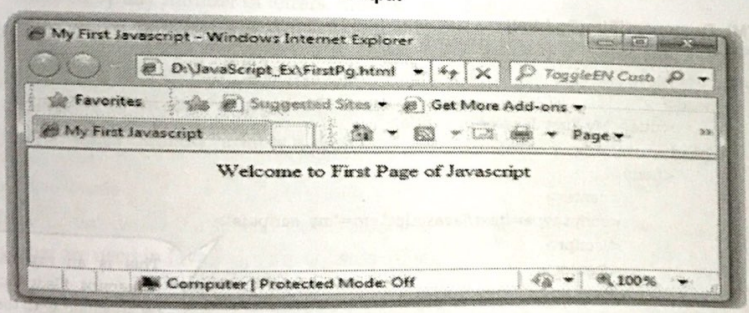
```
        ...
        ...
</script>
```

Example 3.2.1 *Write a JavaScript to display Welcome message in JavaScript*

Solution :

```
<!DOCTYPE html >
<html >
   <head>
      <title> My First Javascript </title>
   </head>
   <body>
      <center>
      <script type="text/javascript">
      /*This is the First JavaScript*/
      document.write(" Welcome to First Page of Javascript");
      </script>
      </center>
   </body>
</html>
```

Output



Script Explanation :

In above script

(1) We have embedded the javaScript within

```
   <script type="text/javascript">
   ...
   </script>
```

(2) A comment statement using /* and */. Note that this type of comment will be recognized only within the <script> tag. Because, JavaScript supports this kind of comment statement and not the XHTML document.

(3) Then we have written document.write statement, using which we can display the desired message on the web browser.

Indirectly Embedding JavaScript

If we want to embed the JavaScript indirectly, that means if the script is written in some another file and we want to embed that script in the HTML document then we must write the script tag as follows -

```
<script type="text/javascript" src="MyPage.js">
    ...
    ...
    ...
</script>
```

Javascript is which is to be embedded is in the file MyPage.js

We will follow the following steps to use the external JavaScript file.

Step 1 : Create an XHTML document as follows -

XHTML Document[FirstPg.html]

```
<!DOCTYPE html >
<html>
    <head>
        <title> My First Javascript </title>
    </head>
        <body>
            <center>
            <script type="text/javascript" src="my_script.js">
            </script>
            </center>
        </body>
    </html>
```

This is an external javascript file,it can be specified with the attribute **src**

Step 2 :

JavaScript[my_script.js]

```
/*This is the First JavaScript*/
        document.write(" Welcome to First Page of Javascript");
```

Step 3 : Open the HTML document in Internet Explorer and same above mentioned output can be obtained.

Advantages of indirectly embedding of JavaScript

1. Script can be hidden from the browser user.
2. The layout and presentation of web document can be separated out from the user interaction through the JavaScript.

Disadvantages of indirectly embedding of JavaScript

1. If small amount of JavaScript code has to be embedded in XHTML document then making a separate JavaScript file is meaningless.
2. If the JavaScript code has to be embedded at several places in XHTML document then it is complicated to make separate JavaScript file and each time invoking the code for it from the XHTML document.

3.3 Keywords, Identifiers, and Comments

1. Identifiers

Identifiers are the names given to the variables. These variables hold the data value. Following are some conventions used in JavaScript for handling the identifiers -

1. Identifiers must begin with either letter or underscore or dollar sign. It is then followed by any number of letters, underscores, dollars or digits.
2. There is no limit on the length of identifiers.
3. The letters in the identifiers are case-sensitive. That means the identifier INDEX, Index, index, inDex are considered to be distinct.
4. Programmer defined variable names must not have upper case letters.

2. Reserved words

Reserved words are the special words associated with some meaning. Various reserve words that are used in JavaScript are enlisted as below –

break	Continue	delete	for	in	return	throw	var	with
case	default	else	function	instanceof	switch	try	void	
catch	do	finally	if	new	this	typeof	while	

3. Comments

JavaScript supports following comments

1. The // i.e a single line comment can be used in JavaScript.
2. The /* and */ can be used as a multi-line comment.

3. The XHTML <!--> and <--> is also allowed in JavaScript

4. Semicolon

While writing the JavaScript the web programmer must give semicolon at the end of the statements

3.4 Data Types

JavaScript defines two entities primitives and objects. The primitives are for storing the values whereas the object is for storing the reference to the actual value.

There are following primitive types used in JavaScript

1. Number 2. String 3. Boolean
4. Undefined 5. Null

There are three type of predefined objects in JavaScript

1. Number 2. String 3. Boolean

These objects are called **wrapper objects**. These wrapper objects provide properties and methods which can be used by primitive types.

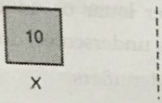


Fig. 3.4.1 (a) Representation of primitive type

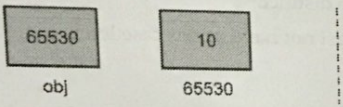


Fig. 3.4.1 (b) Representation of object

3.5 Variables

In JavaScript we can declare the variable using the reserved word var. The value of this variable can be any thing; it can be numeric or it can be string or it can be a Boolean value.

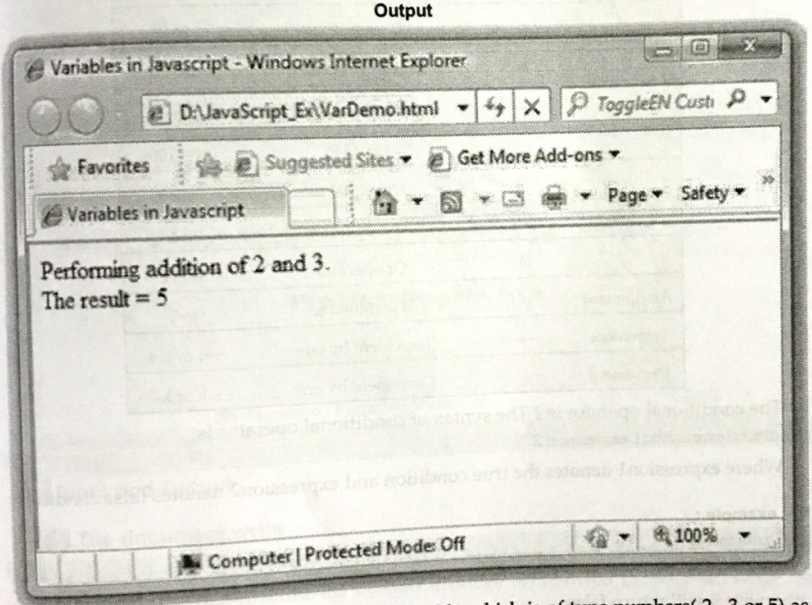
JavaScript[VarDemo.html]

```
<!DOCTYPE html >
<html >
 <head>
 <title> Variables in Javascript </title>
</head>
```

```
<body>
<script type="text/javascript">
var a,b,c;
var string;
a=2;
b=3;
c=a+b;
string ="The result = ";
document.write("Performing addition of 2 and 3.  "+"<br/>");
document.write(string);
document.write(c);
</script>
</body>
</html>
```

Variable declaration is done using **var**. Note that there is no data type required for handling variables.

Output



Note that using **var** we can define the variable which is of type numbers(2 , 3 or 5) as well as the string "The result".

3.6 Operators

Various operators used by JavaScript are as shown in following table -

Type	Operator	Meaning	Example
Arithmetic	+	Addition or unary plus	c = a+b
	-	Subtraction or unary minus	d = -a
	*	Multiplication	c=a*b
	/	Division	c=a/b
	%	Mod	c=a%b
Relational	<	Less than	a<4
	>	Greater than	b>10
	<=	Less than equal to	b<=10
	>=	Greater than equal to	a>=5
	==	Equal to	x==100
	!=	Not equal to	m!=8
Logical	&&	And operator	0&&1
	||	Or operator	0 ||1
Assignment	=	Is assigned to	a=5
Increment	++	Increment by one	++i or i++
Decrement	--	Decrement by one	--k or k--

The conditional operator is ? The syntax of conditional operator is

```
Condition?expression1:expression 2
```

Where expression1 denotes the true condition and expression2 denotes false condition.

For example :

```
a>b?true:false
```

This means that if a is greater than b then the expression will return the value true otherwise it will return false.

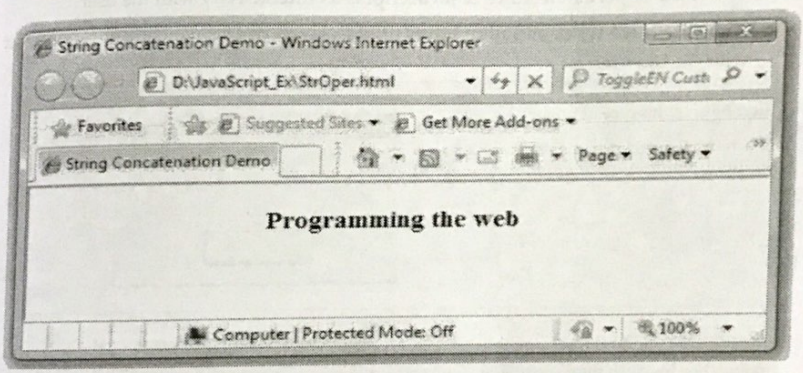
3.6.1 String Concatenation Operator

Two string can be concatenated using the + operator. A variable can be concatenated with the string using + operator also.

JavaScript[StrOper.html]

```
<!DOCTYPE html >
<html >
<head>
 <title>String Concatenation Demo</title>
</head>
<body>
 <center>
 <script type="text/javascript">
  var first_string;
  first_string="Programming";
  document.write("<h3>"+first_string+" the web"+"</h3>");
 </script>
 </center>
</body>
</html>
```

Output



3.7 Input and Output

3.7.1 The document.write

- For displaying the message on the web browser the basic method being used is **document.write**. To print the desired message on the web browser we write the message in double quotes inside the document.write method.
- **For example**

```
document.write("Great India");
```

- We can pass the variable instead of a string as a parameter to the document.write.

For example

```
var my_msg="Great India";
document.write(my_msg);
```

Note that the variable my_msg should not be passed in double quotes document.write otherwise the result the string "my_msg" will be printed on the browse instead of the string "Great India" .

- We can combine some HTML tags with the variable names in document.write s that the formatted display can be possible on the web browser.

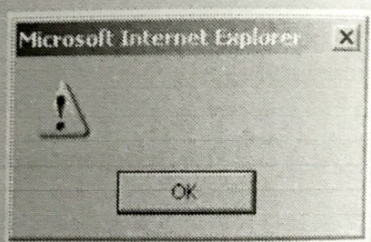
For example - if we want to print the message "Great India" on the web browser i bold font then we can write following statements-

```
var my_msg="Great_India";
document.write("<strong>"+my_msg+"</strong>");
```

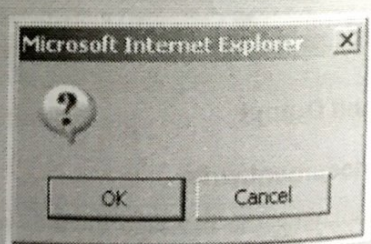
3.7.2 Popup Box

- One of the important features of JavaScript is its **interactivity** with the user.
- There are **three types of popup boxes** used in JavaScript by which user can interac with the browser.

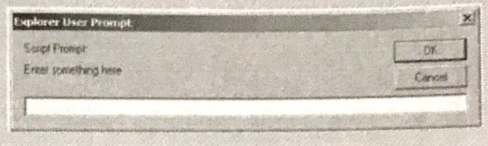
alert box : In this type of popup box some message will be displayed.



confirm box: In this type of popup box in which the message about confirmation will be displayed. Hence it should have two buttons **Ok** and **Cancel.**



Prompt box is a type of popup box which displays a text window in which the user can enter something. Hence it has two buttons **Ok** and **Cancel**.

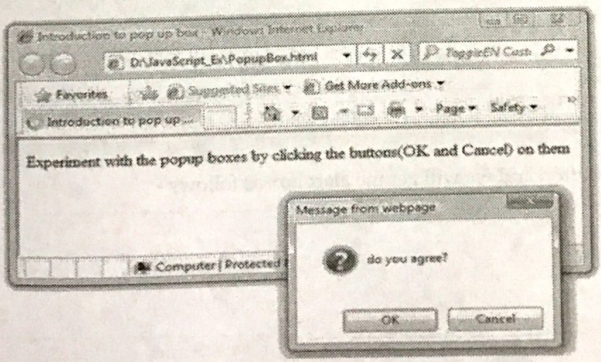


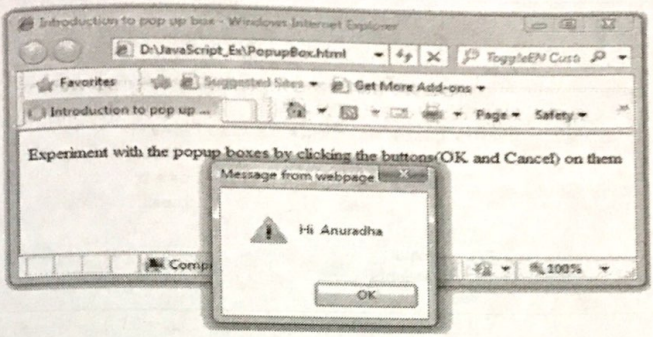
JavaScript[PopupBox.html]

```
<!DOCTYPE html>
<html >
<head>
  <title>Introduction to pop up box</title>
</head>
<body>
<p>Experiment with the popup boxes by clicking the buttons(OK and Cancel) on them</p>

  <script type="text/javascript">
   if(confirm("do you agree?"))
     alert("You have agreed");
   else
     input_text=prompt("Enter some string here...","");
   /*the value entered in prompt box is returned
   and stored in the variable text */
     alert("Hi "+input_text);
  </script>
</body>
</html>
```

Output





Thus alert, confirm and prompt boxes cause the browser to wait for user response. The user responds by clicking the button on these popup boxes.

Script Explanation

On loading this script using web browser first of all a confirm box will be displayed. If we click on OK button an alert box will appear. Otherwise a prompt box will be displayed. Again if we click on the OK button of prompt box again an alert box will appear which will display the string which you have entered on prompt box. If you run the above script you will get all these effects.

3.8 Conditions and Loops

Various control structures used in JavaScript are

Statement	Syntax	Example

if-else	if(condition)
	statement
	else
	statement
		if(a>b)
		document.write(" a is greater than b");
		else
		document.write("b is greater than a");

while	while(condition)
	{
	statements
	}
		while(i<5)
		{
		i=i+1;
		document.write("value of i"+i)
		}

do..while	do
	{
	}while(condition);
		do
		{
		i=i+1;
		document.write("value of i"+i)
		}while(i<5);

for	for(initialization;test condition;stepcount)	for(i=0;i<5;i++) {

	{
	statements
	}
		document.write(i);
		}

switch...case	switch(expression)
	{
	case 1: statements
	break;
	case 2: statements
	break;
	...
	default:
	statements
	}
		switch(choice)
		{
		case 1: c=a+b;
		break;
		case 2:c=a-b;
		break;
		}

break :
Similar to C or C++, the break statement is used to break the loop.
	break;	for(i=10;i>=0;i--)
		{
		if(i==5)
		break;
		}

continue:
The continue statement is used in a loop in order to continue(skip). The keyword **continue** is used to make use of continue statement in a loop.
	continue;	for(i=10;i>=0;i--)
		{
		if(i==5)
		{
		x=i;
		continue;
		}

Control structure is essential part of JavaScript. The Control Structures in JavaScript are just similar to that of C or C++. Various control structures are -

1. if statement 2. While 3. Do-while 4. for
5. switch case 6. break 7. Continue

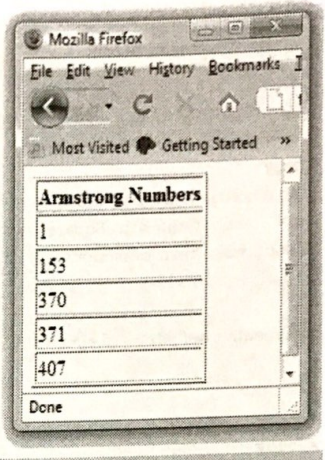
Let us discuss various examples that make use of control structures.

Example 3.8.1 *Develop a javascript to generate 'ARMSTRONG NUMBERS' between the range 1 to 100. [Eg : 153 is an Armstrong number, since sum of the cube of the digits is equal to the number i.e. [13+53+33=153]*

Solution :

```
<html>
<body>
<table border="1" allign="center">
<th>Armstrong Numbers</th>
<script type="text/javascript">
var num,i,temp,sum;
```

```
var n=0;
i=1;
do
{
 num=i;
 sum=0;
 while(num>0)
 {
 n=num%10;
 n=parseInt(n);
 num=num/10;
 num=parseInt(num);
 sum=sum+(n*n*n);
 }
 if(sum==i)
 {
 document.write("<tr><td>"+i+"</td></tr>");
 }
 i++;
}while(i<=1000);
</script>
</table>
</body>
</html>
```



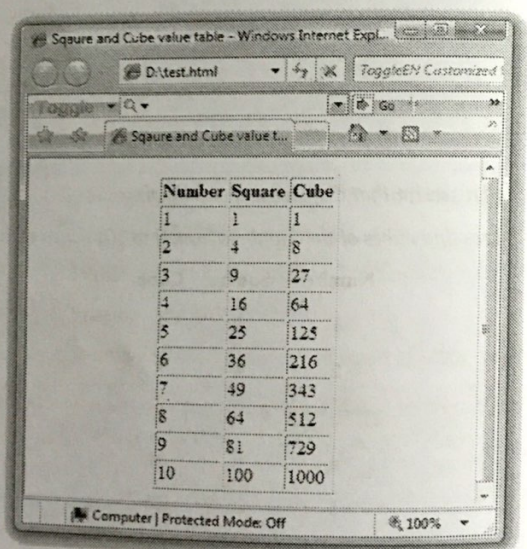
Example 3.8.2 *Write a javascript that displays as per following :*

(Calculate the squares and cubes of the numbers from 0 to 10)

Number	Square	Cube
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

Solution :

```
<html>
<head>
<title>Sqaure and Cube value table</title>
</head>
<body>
<table border=1 align="center">
<th>Number</th><th>Square</th><th>Cube</th>
<script type="text/javascript">
for (i=1; i<=10; i++)
{
document.write("<tr><td>"+i+"</td><td>"+(i*i)+"</td><td>"+(i*i*i)+"</td></tr>");
}
</script>
</table>
</body>
</html>
```



Number	Square	Cube
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

Example 3.8.3 *Write a script that reads an integer and displays whether it is a prime number or not.*

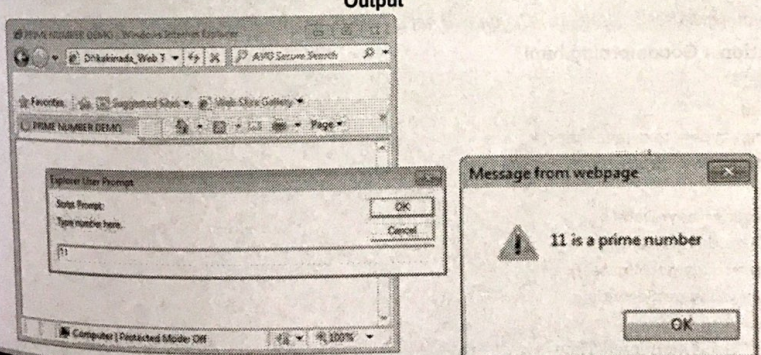
Solution :

```
<html>
 <head>
```

```
<title>PRIME NUMBER DEMO</title>
</head>
<body>
<script type="text/javascript">
var num=prompt("Type number here.","");
var b;
var flag=1;
for(i=2;i<num;i++)
{
  b=num%i;
  if(b==0)
  {
   flag=0;

   break;
  }
}
if(flag==0)
  alert(num+" is not a prime number");
else
  alert(num+" is a prime number");
</script>
</body>
</html>
```

Output



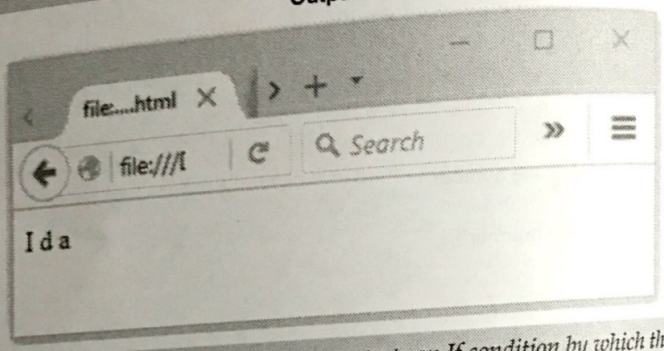
Example 3.8.4 *Write a JAVAScript to print characters of a string at odd positions. (for example for the string India, I, d and a should get printed).*

Solution :

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

```
<script>
var str="India";
var k=str.length;
for(i=0;i<=k;i=i+2)
{
n=str.charAt(i);
document.write(n);
document.write(" ");
}
</script>
</body>
</html>
```

Output



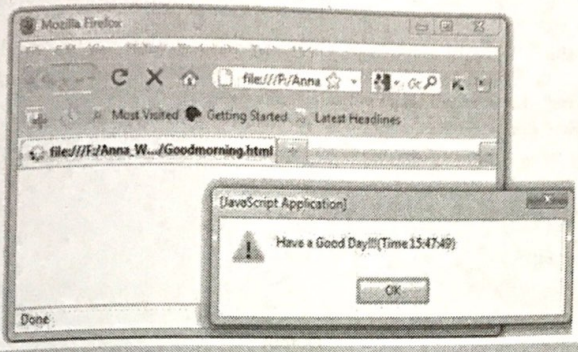
Example 3.8.5 *Develop a JavaScript page to demonstrate an If condition by which the time on your browser is less than 10; you will get a "Good morning" greeting.*

Solution : Goodmorning.html

```
<html>
<head>
<script type="text/javascript">
function GreetMsg()
{
 var today=new Date();
 var h=today.getHours();
 var m=today.getMinutes();
 var s=today.getSeconds();
 if(h<10)
  alert("Good Morning!!!(Time "+h+":"+m+":"+s+")");
 else
  alert("Have a Good Day!!!(Time "+h+":"+m+":"+s+")");
}
</script>
</head>
<body onload="GreetMsg()">
```

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

Output



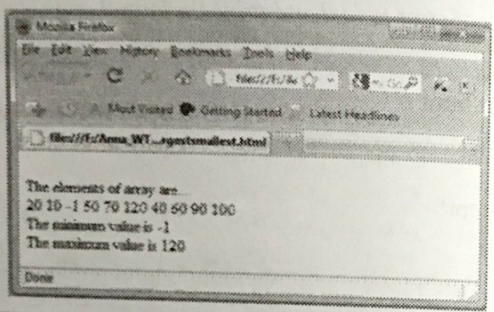
Example 3.8.6 *Write a JavaScript to find and print the largest and smallest values among 10 elements of an array.*

Solution : largestsmallest.html

```
<html>
 <head>
 <script type="text/javascript">
 function fun()
 {
  a=new Array(10);
  a[0]=20;
  a[1]=10;
  a[2]=-1;
  a[3]=50;
  a[4]=70;
  a[5]=120;
  a[6]=40;
  a[7]=60;
  a[8]=90;
  a[9]=100;
  document.write("<br/>The elements of array are...<br/>");
  max_val=a[0];
  for(i=0;i<10;i++)
   document.write(" "+a[i]);
  min_val=a[0];
  max_val=a[0];
  for(i=0;i<10;i++)
```

```
 {
   if(a[i]<min_val)
     min_val=a[i];
   if(a[i]>max_val)
     max_val=a[i];
 }
 document.write("<br/>The minimum value is "+min_val);
 document.write("<br/>The maximum value is "+max_val);
}
 </script>
<title>Finding largest and smallest Element</title>
</head>
 <body onload=fun()>
 </body>
</html>
```

Output



Example 3.8.7 *Write a JavaScript to find the product of first 15 even numbers.*

Solution :

```
<html>
<body>
<script type="text/javascript">
prod=1;
for(i=1;i<15;i++)
{
    if(i%2)
    {
        prod=prod*i;
    }
}
document.write("Product of odd numbers between 1 to 15 is "+prod);
</script>
</body>
</html>
```

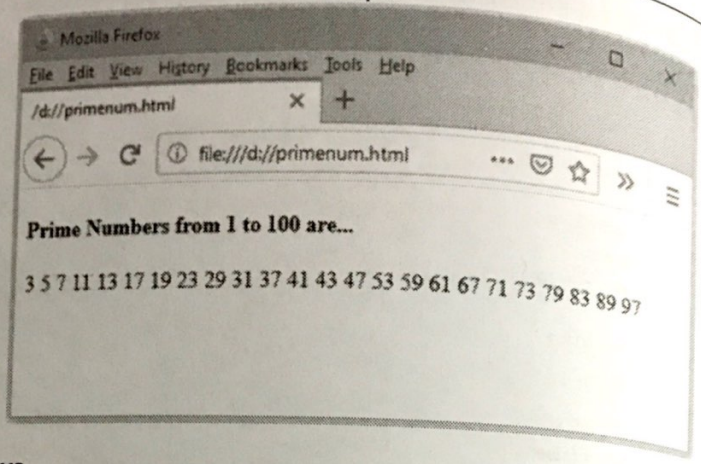
Example 3.8.8 *Write a JavaScript program to print prime numbers from 1 to 100.*

Solution :

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">

function primeNumber(from, to){
        var flag=false;
        for(i = from; i <= to; i++)
       {
        for( j = 2; j < i; j++)
        {
        if(i % j == 0)
            {
                flag = false;
                break;
            }
        else
            {
                flag = true;
            }
        }
        if(flag)
        {
        document.write(" "+i);
        }
      }
    }
</script>
</head>
<body>
<h4> Prime Numbers from 1 to 100 are...</h4>
<script type="text/javascript">
primeNumber(1, 100)
</script>
</body>
</html>
```

Output



3.9 Arrays

- Arrays is a collection of similar type of elements which can be referred by a common name.
- Any element in an array is referred by an array name followed by "[" followed by position of the element followed by].
- The particular position of element in an array is called array index or subscript.

For example –

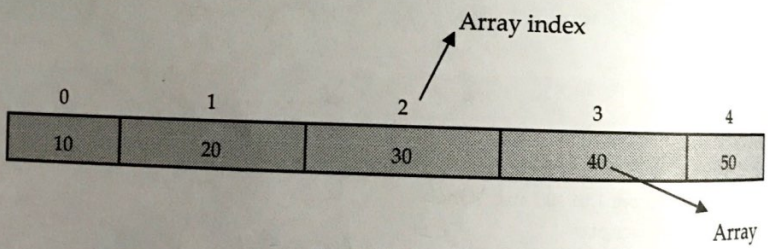


Fig. 3.9.1 Arrays

- Normally the first element in an array is stored at 0th location, however we can start storing the element from any position.

3.9.1 Array Declaration

- In JavaScript the array can be created using **Array** object.
- Suppose, we want to create an array of 10 elements then we can write,

```
var ar = new Array(10);
```

- Using new operator we can allocate the memory dynamically for the arrays.
- In the brackets the size of an array is mentioned and the var ar denotes the name of the array. Thus by the above sentence an array ar will be created in which we can store 10 elements at the most. Sometimes the above statement can be written like this

```
var ar;
   ar=new Array(10);
```

3.9.2 Array Initialization

Let us see how to store some elements in an array.

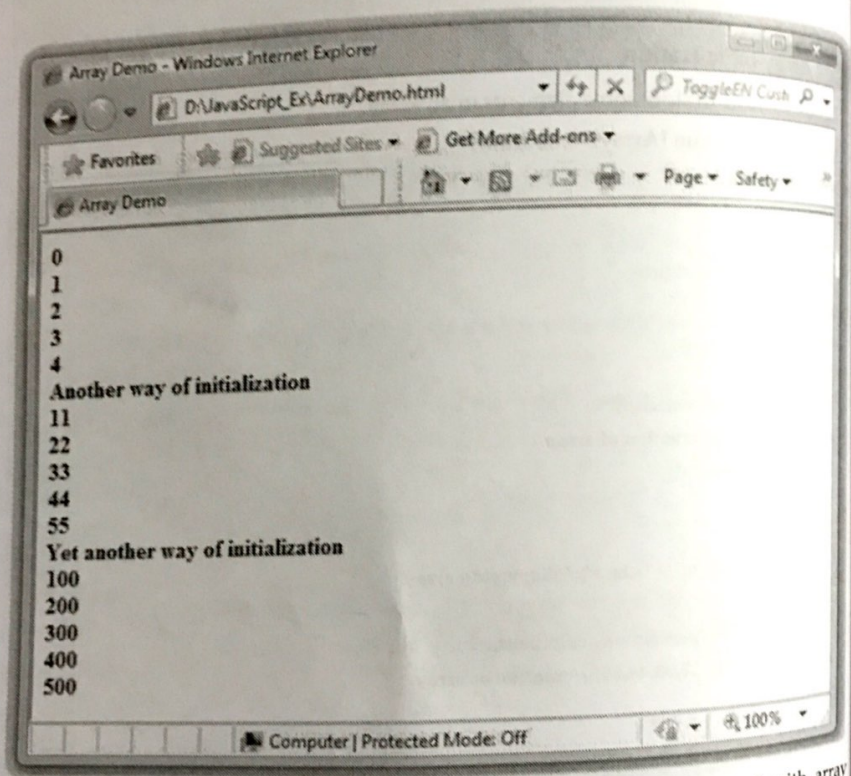
- **JavaScript Program [ArrayDemo.html]**

```
<!DOCTYPE html>
<html>
<head>
  <title>Array Demo</title>
</head>
<body>
<strong>
  <script type="text/javascript">
   a=new Array(5);//creation of array
   for(i=0;i<5;i++)
   {
    a[i]=i;
    document.write(a[i]+"<br>");//displaying array
   }
   document.write("Another way of initialization"+"<br>");
   b=new Array(11,22,33,44,55);//creation of array
   for(i=0;i<5;i++)
   {
    document.write(b[i]+"<br>");//displaying array
   }
   document.write("Yet another way of initialization"+"<br>");
   var c=[100,200,300,400,500];//creation of array
   for(i=0;i<5;i++)
   {
    document.write(c[i]+"<br>");//displaying array
```

```
    }
</script>
</strong>
</body>
</html>
```

As you can notice that, in above JavaScript an array can be initialized in three differen ways which is shown by boldface. Hence an output of above script will be

Output



There is one control structure in JavaScript which is closely associated with array elements and such a control structure is **for...in**. Let us see a simple JavaScript which makes use of **for-in** control structure to display elements of an array.

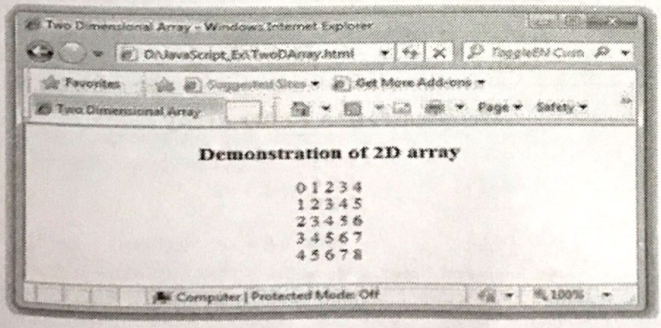
3.9.3 Two Dimensional Array

Actually JavaScript does not support the multidimensional arrays. Hence for defining the 2D array we make use of single dimensional array, how? Here it is -

JavaScript[TwoDArray.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Two Dimensional Array</title>
</head>
<body>
 <center>
 <h3> Demonstration of 2D array</h3>
 <script type="text/javascript">
  a=new Array();//creation of rows of array
  for(i=0;i<5;i++)
  a[i]=new Array();//creating columns of array
  for(i=0;i<5;i++)
  {
   for(j=0;j<5;j++)
   {
    a[i][j]=i+j;
    document.write(a[i][j] +"  ");
   }
   document.write("<br>");
  }
 </script>
</center>
</body>
</html>
```

Output

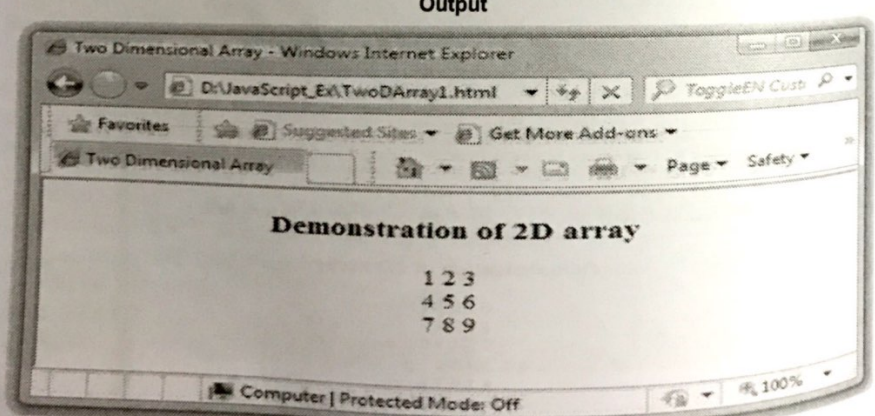


There is another way by which the two dimensional array can be initialized. This method is represented in the following Script

JavaScript[TwoDArray1.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Two Dimensional Array</title>
</head>
<body>
 <center>
 <h3> Demonstration of 2D array</h3>
 <script type="text/javascript">
  var a=[ [1,2,3],
          [4,5,6],
          [7,8,9]
        ]; //creation of array
  for(i=0;i<3;i++)
  {
  for(j=0;j<3;j++)
  {
  document.write(a[i][j] +"  ");
  }
  document.write("<br>");
  }
 </script>
</center>
</body>
</html>
```

Output



```
sum=0;
for(i=1;i<n;i++)
{
if(i%2==0)
  sum=sum+i;
}
document.write("<br>"+" Sum of first "+n+" even numbers is "+sum+"<br>");
}
</script>
</head>
<body>
<script type="text/javascript">
  var n=prompt("Enter the value of n","");
  EvenNumSum(n);
</script>
</body>
</html>
```

3.11 JavaScript Objects and DOM

3.11.1 Definition of DOM

- The Document Object Modeling (DOM) is for defining the standard for accessing and manipulating HTML,XML and other scripting languages.
- It is the **W3C recommendation** for handling the structured documents.
- Normally the structured information is provided in XML, HTML and many other documents.
- Hence DOM provides the standard set of programming interfaces for working with XML and XHTML and JavaScript.

What is DOM?

Document Object Model (DOM) is a set of platform independent and language neutral Application Programming Interface (API) which describes how to access and manipulate the information stored in XML,HTML and JavaScript documents.

3.11.2 DOM Tree

- Basically DOM is an **Application Programming Interface (API)** that defines the interface between HTML document and application program. That means, suppose application program is written in Java and this Java program wants to access the

elements of HTML web document then it is possible by using a set of Application Programming Interfaces (API) which belongs to the DOM.

- The DOM contains the **set of interfaces** for the document tree node type. These interfaces are similar to the Java or C++ interfaces. They have **objects, properties and methods** which are useful for respected node type of the web document.
- The documents in DOM are represented using a tree like structure in which every element is represented as a node. Hence the tree structure is also referred as DOM tree.
- **For example :** Consider following XHTML document.

```
<html>
    <head>
      <title>This is My Web Page </title>
    </head>
    <body>
      <h1>Hello Friends </h1>
      <h2>How are you?</h2>
      </h3>See you</h3>
    </body>
    </html>
```

- The DOM tree will be

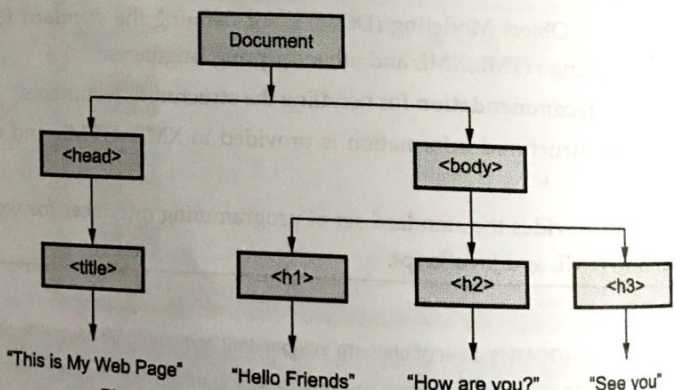


Fig. 3.11.1 DOM structure for simple web document

- We can describe some basic terminologies used in DOM tree as follows -
  1. Every element in the DOM tree is called **node.**
  2. The topmost single node in the DOM tree is called the **root.**
  3. Every child node must have a parent **node.**
  4. The bottommost nodes that have no children are called **leaf nodes.**
  5. The nodes that have the common parent are called **siblings.**

3.11.3 Using DOM Methods

We can access or change the contents of document by using various methods. Some commonly used properties and methods of DOM are as follows :

Methods

Method	Meaning
getElementById	This method is used to obtain the specific element which is specified by some id within the script.
createElement	This method is used to create an element node.
createTextNode	Useful for creating a text node.
createAttribute	Useful for creating attribute.
appendChild	For adding a new child to specified node, this method is used.
removeChild	For removing a child node of a specific node, this method is used.
getAttribute	This method is useful for returning the specified attribute value.
setAttribute	This method is useful for setting or changing the specified attribute to the specified value.

Properties

Property	Meaning
attributes	This property is used to get the attribute nodes of the node.
parentNode	This DOM property is useful for obtaining the parent node of the specific node.
childNodes	This DOM property is useful for obtaining the child nodes of the specific node.
innerHTML	It is useful for getting the text value of a node.

3.11.3.1 Accessing Elements using DOM

- There are several ways by which we can access the elements of the web document.
- To understand these methods of accessing we will consider one simple web document as follows.

```
<html >
<head>
   <title>This is My Web Page </title>
</head>
<body>
```

```
<form name="form1">
    <input type="text" name="myinput"/>
</form>
</body>
</html>
```

Method 1

- Every XHTML document element is associated with some address. This address called **DOM address.**
- The document has the collection of **forms** and **elements**. Hence we can refer the te box element as

```
var Dom_Obj=document.forms[0].elements[0];
```

- But this is not the suitable method of addressing the elements. Because if we chan the above script as

```
...
<form name="form1">
    <input type="button" name= "mybutton"/>
    <input type="text" name="myinput"/>
</form>
...
```

then index reference gets changed. Hence another approach of accessing the elem is developed.

Method 2

- We can access the desired element from the web document using JavaScript me **getElementById**. The element access can be made as follows -

```
var Dom_Obj=document.getElementById("myinput");
```

- But if the element is in particular group, that means if there are certain elemer the form such as radio buttons or check boxes then they normally appear groups. Hence to access these elements we make use of its index. Consid following code sample

```
<form id="Food">
    <input type="checkbox" name="vegetables" value="Spinach" />Spinach
    <input type="checkbox" name="vegetables" value="FenuGreek" />FenuGreek
    <input type="checkbox" name="vegetables" value="Cabbage" />Cabbage
</form>
```

- For getting the values of these checkboxes we can write following code.

```
var Dom_obj=document.getElementById("Food");
for(i = 0 ; i < Dom_Obj.vegetables.length ; i++)
```

```
  document.write(parseInt("100"));
 </script>
</body>
</html>
```

The output of above code will be 100.

Similar to the **parseInt** function the **parseFloat** function is used to obtain the real value from the string.

4. eval

The eval function is used to evaluate the expression.

The syntax is

```
eval(string);
```

Example

globlafun4.html

```
<html>
 <body>
  <script type="text/javascript">
   document.write(eval("2+3*5"));
  </script>
 </body>
</html>
```

The output of above code will be **17.**

3.13 Javascript Validations

- Various control objects are placed on the form. These control objects are called **widgets.**
- These widgets used in JavaScript are – Textbox, Push button, Radio button, Checkbox and so on.
- In JavaScript the **validation of these widgets** is an important task.

Let us understand the validation of form elements with the help of examples –

Example 3.13.1 *Write a JavaScript for password verification.*

Solution :

```
<!DOCTYPE html>
<html>
<head>
 <title>Demo of onlclick Tag Attibute</title>
<script type="text/javascript">
```

```
function my_fun()
{
var mypwd=document.getElementById("pwd");
var my_re_pwd=document.getElementById("re_pwd");
if(mypwd.value=="")
{
 alert("You have not entered the password");
 mypwd.focus();
 return false;
}
if(mypwd.value!=my_re_pwd.value)
{
 alert("Password is not verified, Re-enter both the passwords");
 mypwd.focus();
 mypwd.select();
 return false;
}
else
{
 alert("Congratulations!!!");
 return true;
}
}
</script>
</head>
<body>
 <center>
 <form id="form1">
  <label> Enter your password
  <input type="password" value="" id="pwd" />
  </label>
  <br/><br/>
  <label> Re-Enter the password
  <input type="password" value="" id="re_pwd" onblur="my_fun();"/>
  </label><br/>
  <input type="submit" value="Submit" name="submit" onsubmit="my_fun();"/>
  <input type="reset" value="Reset" name="reset"/><br/>
 </form>
 </center>
</body>
</html>
```

```
Course:
<select name="course">
<option value="">Select an Option:</option>
<option value="Computer">Computer</option>
<option value="Mechanical">Mechanical</option>
<option value="E&Tc">E&Tc</option>
</select>
<br/><br/>
<input type="button" name="SubmitButton" value="Submit"
onClick="ValidateForm(this.form)">
<input type="reset" value="Reset">
</form>
</body>
</html>
```

3.4 Regular Expressions

- **Definition :** Regular Expression is a special text string that defines the search pattern. It is a logical expression.
- **For example** – For counting specific characters in a string or to replace some substring by another substring we need to create a regular expression.
- We can create a regular expression pattern using forward slash /. For instance -

  re = /abc/
- Regular expression is a powerful way for searching and replacing the characters in the string.
- The words of regular expression are called **special characters.**
- Various special characters that can be used in Regular expression along with their meanings are written in the following table :

Special Character	Meaning
.	Any character except newline
A	The character a
ab	The string ab
a|b	a or b
a*	0 or more a's
\	Escapes a special character
[ab-d]	One character of: a, b, c, d

[^ab-d]	One character except: a, b, c, d
[\b]	Backspace character
\d	One digit
\D	One non-digit
\s	One whitespace
\S	One non-whitespace
\w	One word character
\W	One non-word character
*	0 or more
+	1 or more
?	0 or 1
{2}	Exactly 2
{2, 5}	Between 2 and 5
{2,}	2 or more
(...)	Group of pattern
^	Start of string
$	End of string
\b	Word boundary
\n	Newline
\r	Carriage return
\t	Tab
\0	Null character

Methods that use regular expressions

Method	Description
exec	A RegExp method that executes a search for a match in a string. It returns an array of information or null on a mismatch.

test	A RegExp method that tests for a match in a string. It returns true or false.
match	A String method that executes a search for a match in a string. It returns an array of information or null on a mismatch.
matchAll	A String method that returns an iterator containing all of the matches, including capturing groups.
search	A String method that tests for a match in a string. It returns the index of the match, or –1 if the search fails.
replace	A String method that executes a search for a match in a string, and replaces the matched substring with a replacement substring.
split	A String method that uses a regular expression or a fixed string to break a string into an array of substrings.

3.14.1 Finding Non Matching Characters

We can find the non matching characters from the given text by placing ^ as the first character within a square [ ].

Example 3.14.1 *Write a Java program that checks whether the string entered by the user contains digit or not.*

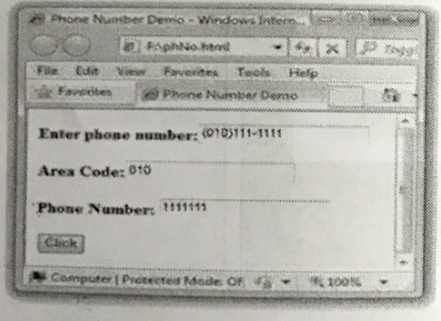
Solution :

```
<!DOCTYPE html>
<html>
<head>
   <script type="text/javascript">
      function TestString(str)
      {
        re=/[^[0-9]]/; // [0-9] indicates any digit
        if(re.test(str))
         {
         alert("The string does not contain
```

Solution : HTML Document[phNo.html]

```
<html>
<head>
  <title>Phone Number Demo</title>
  <script type="text/javascript">
  function TestString()
   {
   var str=document.form1.input.value;
   var i=str.match(/("\d{3}")"\" "\d{3}"-"\d{4}/);
   var temp1=str.split("(");
   var temp2=temp1[1].split(")")
   document.form1.area.value=temp2[0];
   var temp3=temp2[1].split(" ");
      var temp4=temp3[1].split("-")
   document.form1.phnum.value=temp4[0]+temp4[1];
   }
  </script>
</head>
 <body>
   <form name="form1">
   <b>Enter phone number:</b><input type="text" name="input" value=""><br/><br/>
   <b>Area Code:</b><input type="text" name="area" value=""><br/><br/>
   <b>Phone Number:</b> <input type="text" name="phnum" value=""><br/><br/>
   <input type="button" value="Click"onclick=TestString(input)>
 </body>
</html>
```

Output



3.15 Event Handling with Javascript

- **Definition of Event :** Event is an activity that represents a change in the environment. For example mouse clicks, pressing a particular key of keyboard represent the events. Such events are called intrinsic events.

- **Definition of Event Handler :** Event handler is a script that gets executed in response to these events. Thus event handler enables the web document to respond the user activities through the browser window.
- JavaScript support this special type of programming in which events may occur and these events get responded by executing the event handlers. This type of programming is called event-driven programming.
- Events are specified in lowercase letters and these are case-sensitive.
- **Event Registration :** The process of connecting event handler to an event is called event registration. The event handler registration can be done using two methods -
  - Assigning the tag attributes
  - Assigning the handler address to object properties.

Internet Programming

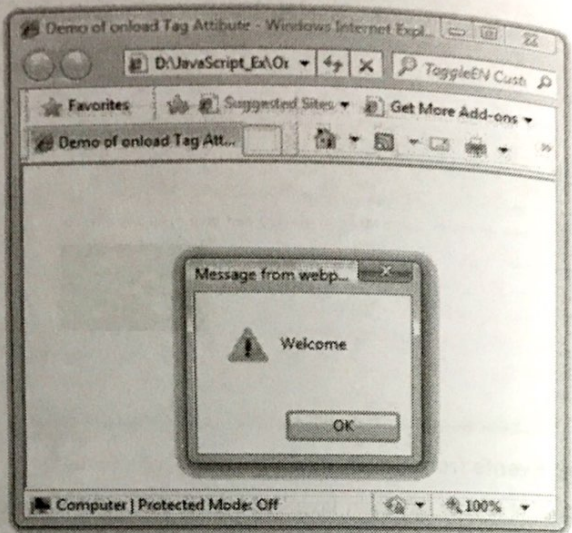
On occurrence of events the tag attribute must be assigned with some user defined functionalities. This will help to execute certain action on occurrence of particular event.

Commonly used events and tag attributes are enlisted in the following table -

Events	Intrinsic event attribute	Meaning	Associated tags
blur	onblur	Losing the focus.	<button> <input> <a> <textarea> <select>
change	onchange	On occurrence of some change.	<input> <textarea> <select>
click	onclick	When user clicks the mouse button.	<a> <input>
dblclick	ondblclick	When user double clicks the mouse button.	<a> <input> <button>
focus	onfocus	When user acquires the input focus.	<a> <input> <select> <textarea>

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

Output



3.16 Callbacks in Javascript

In JavaScript functions are objects. We can pass objects to the function as a parameter. That means we can pass function as a parameter to another function.

The mechanism of passing function as a parameter to another function is called callbacks. For instance –

```
function print(callback) {
  callback();
}
```

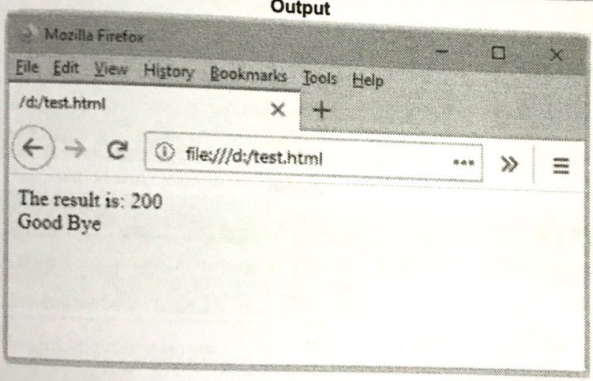
The print( ) function takes another function as a parameter and calls it inside. This is valid in JavaScript and we call it a "callback". So a function that is passed to another function as a parameter is a callback function.

Following JavaScript illustrates the use of callback function

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function mul(a,b,callback)
        {
```

```
        ans = a*b
        document.write("The result is: "+ans);
        callback();
    }
    function display(){
        document.write("<br/>Good Bye");
    }
  </script>
</head>
<body>
<script>
  mul(10,20,display);
</script>
</body>
</html>
```

Output



Script Explanation : In above script

(1) We have written a function named **mul**. To this function, a callback function named **display** is passed as a parameter.

(2) After execution of **mul** function, the callback function **display** function gets called, due to **callback()**.

(3) Thus the mechanism of callback function allows to call the functions subsequently.

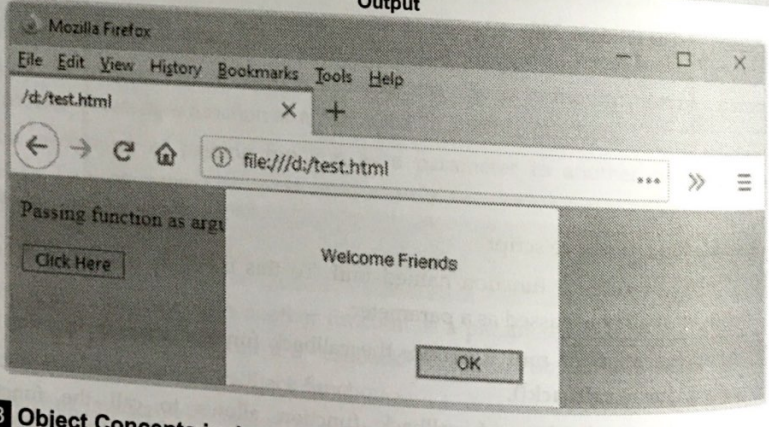
3.17 Function as Arguments in JavaScript

We can pass a function as a argument to another function in the same way as a variable is passed as an argument.

Following JavaScript demonstrates how to pass a function as an argument to another function

```
<!DOCTYPE html>
<html>
  <body>
    <p>
      Passing function as arguments.
    </p>
      <button onclick = "OuterFunction(InnerFunction)">
        Click Here
      </button>
    <script>
      function InnerFunction(value){
        return "Welcome Friends";
      }
      function OuterFunction(func){
        alert(func());
      }
    </script>
  </body>
</html>
```

Output



3.18 Object Concepts in JavaScript

In JavaScript object is a collection of properties. These properties are nothing but the members of the classes from Java or C++. For instance - in JavaScript the object Date() is used which happens to the member of the class in Java.

3.18.1 Math Objects

- For performing the mathematical computations there are some useful methods available from **math** object.
- For example if we want to find out minimum of two numbers then we can write -

```
document.write(math.min(4.5,7.8));
```

The above statement will result in 4.5. Thus using various useful methods we can perform many mathematical computations.

- Here are some commonly used methods from **math** object.

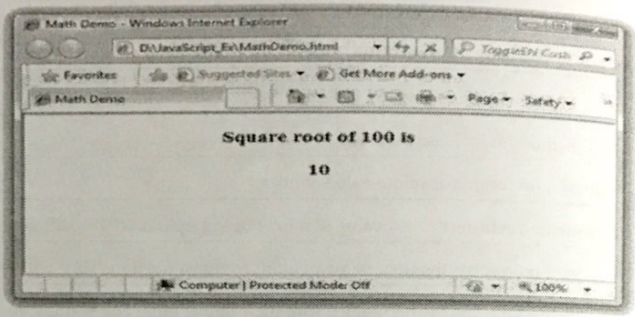
Method	Meaning
sqrt(num)	This method finds the square root of num.
abs(num)	This method returns absolute value of num.
ceil(num)	This method returns the ceil value of num. For example ceil(10.3) will return 11.
floor(num)	This method returns the floor value of num. For example floor(10.3) will return 10.
log(num)	This method returns the natural logarithmic value of num. For example log(7.9) will return 2.
pow(a,b)	This method will compute the ab. For example pow(2,5) will return 32.
min(a,b)	Returns the minimum value of a and b.
max(a,b)	Returns the maximum value of a and b.
sin(num)	Returns the sine of num.
cos(num)	Returns the cosine of num.
tan(num)	Returns the tangent of num.
exp(num)	Returns the exponential value i.e. enum .

JavaScript Program[MathDemo.html]

```
<!DOCTYPE html>
<html >
<head>
 <title>Math Demo</title>
</head>
<body>
 <center>
  <h3>Square root of 100 is </h3>
  <script type="text/javascript">
```

```
  var num=100;
  document.write("<h3>"+Math.sqrt(num)+"</h3>");
  </script>
 </center>
</body>
</html>
```

Output



3.18.2 Number Objects

Various properties of number object are -

Property	Meaning
MAX_VALUE	Largest possible number gets displayed.
MIN_VALUE	Smallest possible number gets displayed.
NaN	When not a number then NaN is displayed.
PI	The value of PI gets displayed.
POSITIVE_INFINITY	The positive infinity gets displayed.
NEGATIVE_INFINITY	The negative infinity gets displayed.

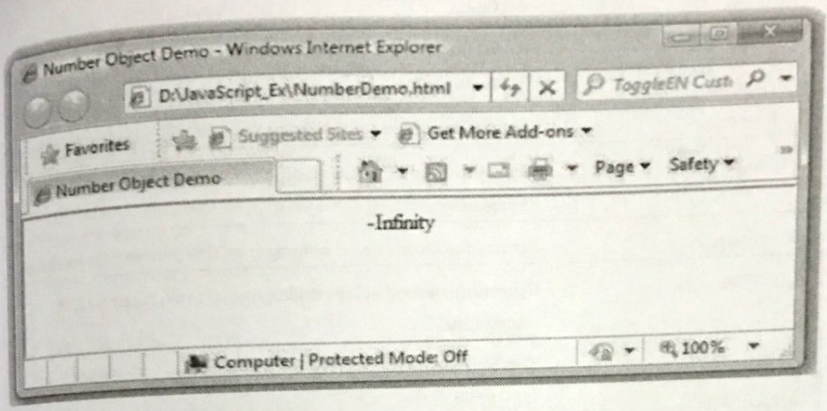
Using **Number.***property_name* we can display the property value. Following JavaScript uses the property of negative infinity.

JavaScript[NumberDemo.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Number Object Demo</title>
</head>
<body>
 <center>
  <script type="text/javascript">
```

```
document.write(Number.NEGATIVE_INFINITY);
</script>
</center>
</body>
</html>
```

Output



3.18.3 Date Objects

- This object is used for obtaining the date and time.
- This date and time value is based on computer's local time (system's time) or it can be based on GMT(Greenwich Mean Time).
- Nowadays this GMT is also known as UTC i.e. Universal Co-ordinated Time. This is basically a world time standard.
- Following are the commonly used methods of Date object.

Method	Meaning
getTime()	It returns the number of milliseconds. This value is the difference between the current time and the time value from 1st January 1970.
getDate()	Returns the current date based on computers local time.
getUTCDate()	Returns the current date obtained from UTC.
getDay()	Returns the current day. The day number is from 0 to 6 i.e. from Sunday to Saturday.
getUTCDay()	Returns the current day based on UTC. The day number is from 0 to 6 i.e. from Sunday to Saturday.

getHours()	Returns the hour value ranging from 0 to 23, based on local time.
getUTCHours()	Returns the hour value ranging from 0 to 23, based on UTC timing zone.
getMilliseconds()	Returns the milliseconds value ranging from 0 to 999, based on local time.
getUTCMilliseconds()	Returns the milliseconds value ranging from 0 to 999, based on UTC timing zone.
getMinutes()	Returns the minute value ranging from 0 to 59, based on local time.
getUTCMinutes()	Returns the minute value ranging from 0 to 59, based on UTC timing zone.
getSeconds()	Returns the second value ranging from 0 to 59, based on local time.
getUTCSeconds()	Returns the second value ranging from 0 to 59, based on UTC timing zone.
setDate(value)	This function helps to set the desired date using local timing or UTC timing zone.
setHour(hr,minute,second,ms)	This function helps to set the desired time using local or UTC timing zone. The parameters that can be passed to this function are hour,minute,seconds and milliseconds. Only hour parameter is compulsory and rest all are the optional parameters.

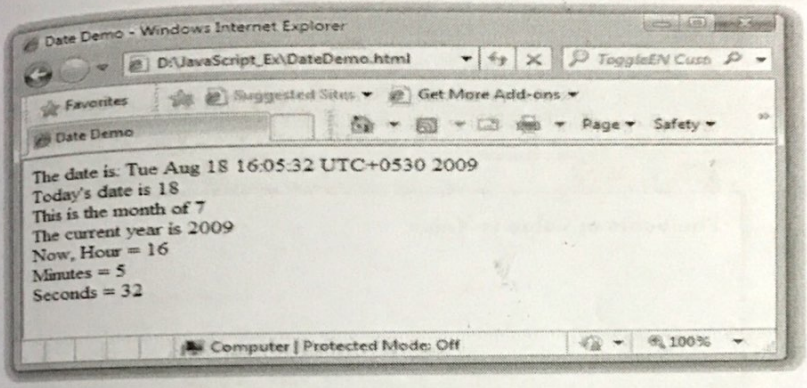
In the following web document we are making use of **Date()** object and some useful methods of it.

JavaScript[DateDemo.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>Date Demo</title>
</head>
<body>
  <script type="text/javascript">
  var my_date=new Date();
  document.write("The date is: "+my_date.toString()+"<br>");
  document.write("Today's date is "+my_date.getDate()+"<br>");
  document.write("This is the month of "+my_date.getMonth()+"<br>");
  document.write("The current year is "+my_date.getFullYear()+"<br>");
  document.write("Now, Hour = "+my_date.getHours()+"<br>");
```

```
document.write("Minutes = "+my_date.getMinutes()+"<br>");
document.write("Seconds = "+my_date.getSeconds()+"<br>");
</script>
</body>
</html>
```

Output



Script Explanation :

1. In the above script we have created an instance *my_date* of the object **Date().**
2. Then using **my_date.toString()** method we can display the current date along with the time.
3. Then using the functions like getDate(), getMonth(), getFullYear() we are displaying the date, month and year separately.
4. Similarly using the functions getHours(), getMinutes() and getSeconds() we can display the current hour, minute and seconds separately.

3.18.4 Boolean Objects

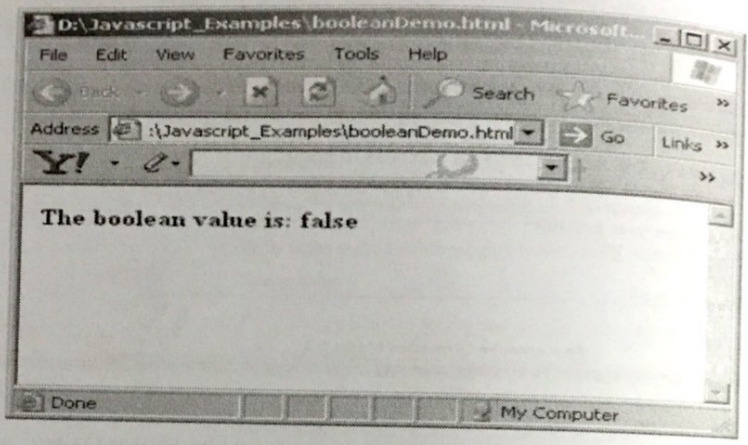
- This object is the simplest kind of object which is used especially when we want to represent **true** and **false** values.
- Here is a simple javascript in which the Boolean type variable is used -

Javascript Program

```
<html>
<body>
<script type="text/javascript">
```

```
var temp=new Boolean(false);
document.write("<b>"+"The boolean value is:  ");
document.write(temp.toString());
</script>
</body>
</html>
```

Output



3.18.5 String Objects

- String is a collection of characters.
- In JavaScript using **string** object many useful string related functionalities can be exposed off.
- Some commonly used methods of string object are concatenating two strings, converting the string to upper case or lower case, finding the substring of a given string and so on.
- Here is a listing of some methods of string.

Method	Meaning
concat(str)	This method concatenates the two strings. For example s1.concat(s2) will result in concatenation of string s1 with s2.
charAt(index_val)	This method will return the character specified by value index_val.
substring(begin,end)	This method returns the substring specified by begin and end character.
toLowerCase()	This function is used to convert all the uppercase letters to lower case.

toUpperCase()	This function is used to convert all the lowercase letters to upper case.
valueOf()	This method returns the value of the string.

There is one important property of string object and that is **length**. For example

```
var my_str="Hello";
var len;
len=my_str.length;
```

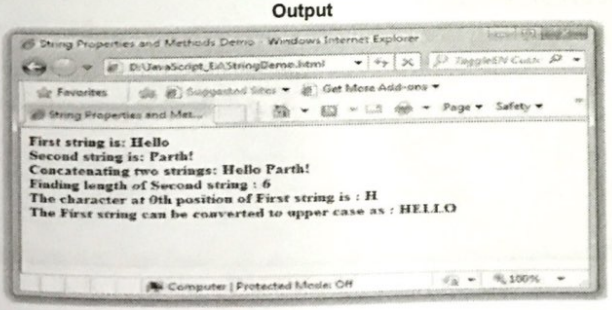
Length of the string "Hello" will be stored in the variable len

Here is sample program in which some methods of string object are used.

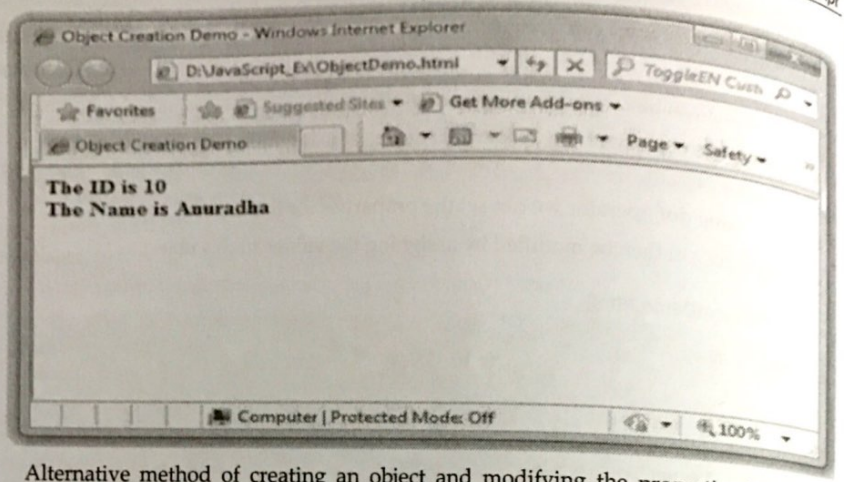
JavaScript[StringDemo.html]

```
<!DOCTYPE html>
<html>
<head>
  <title>String Properties and Methods Demo</title>
</head>
<body>
<strong>
  <script type="text/javascript">
   var s1="Hello ";
   var s2="Parth!";
   document.write("First string is: "+s1+"<br>");
   document.write("Second string is: "+s2+"<br>");
   document.write("Concatenating two strings: "+s1.concat(s2)+"<br>");
   document.write("Finding length of Second string : "+s2.length+"<br>");
   document.write("The character at 0th position of First string is : "+s1.charAt(0)+"<br>");
   document.write("The First string can be converted to upper case as : "+s1.toUpperCase());
  </script>
 </strong>
</body>
</html>
```

Output



Output



Alternative method of creating an object and modifying the properties is as given below -

```
var Myobj={name:"Anuradha",id:10};
```

Then using **document.write** statement we can display the property values as follows -

```
document.write("The ID is "+Myobj.id+"<br/>");
document.write("The Name is "+Myobj.name);
```

The property of created object can be deleted using an expression **delete**. Normally the property of an object is deleted in order to free the allocated memory so that this memory can be reused by other process.

3.19 JSON

- JSON stands for JavaScript Object Notation.
- Using JSON we can store and retrieve data. This text based open standard format.
- It is extended from JavaScript language.

Features of JSON

1. It is **text based, lightweight data interchange format.**
2. It is **language independent.**
3. It is **easy** to **read** and **write.**
4. It is **easy** for **machines** to **parse** and generate.
5. It uses the conventions that are **familiar** to the languages like C, C++, Java, JavaScript, Perl, Python and so on.

3.19.1 Syntax

JSON is built on **two structures :**

1. A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
2. An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

JSON Object

- JSON object holds the Key value pair.
- Each key is represented as string and value can be of any datatype.
- The key and value are separated by colon.
- **Syntax**

```
{ string : value, .....}
```

- **For example**

```
"Age":38
```

- Each key value pair is separated by comma.
- The object is written within the { }curly brackets.

1) JSON object containing value of different data types

```
{
  "student":
  {
    "name":    AAA",      ← String value
    "roll_no":  10,       ← Numeric value
    "Indian":   true      ← Boolean value
  }
}
```

2) JSON Nested Object

```
{
  "student":
  {
    "name":    "AAA",
    "roll_no":  10,
    "address":
      {
      "Street": "Shivaji Nagar",
```

```
        "City": "Pune",
        "Pincode": 411005
        }
    }
}
```

3) Creation of JSON Object with JavaScript

In JavaScript we can write the JSON object and store it in some variable. For example

```
var stud = { "name":"AAA", "roll_no":10, "city":"Pune" };
```

Here the object named stud is created.

Example

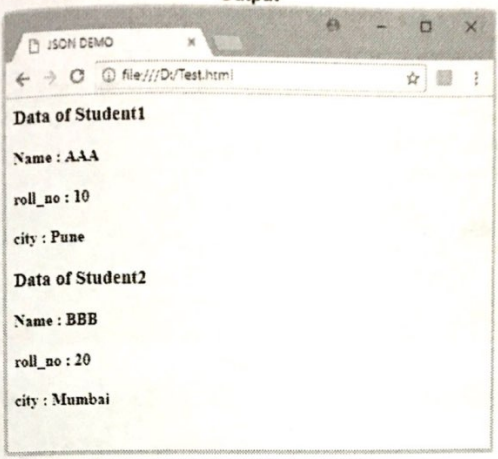
Let us understand how to create an JSON object and obtain its value, and get it displayed on the web browser using JavaScript. The code is as follows -

Test.html

```
<html>
  <head>
    <title>JSON DEMO</title>
    <script language = "javascript" >
      var stud1 = { "name":"AAA", "roll_no":10, "city":"Pune"};
      document.write("<h3>Data of Student1</h3>");
      document.write("<h4>Name : " + stud1.name+"</h4>");
      document.write("<h4>roll_no : " + stud1.roll_no+"</h4>");
      document.write("<h4>city : " + stud1.city+"</h4>");

      var stud2 = { "name":"BBB", "roll_no":20, "city":"Mumbai"}
      document.write("<h3>Data of Student2</h3>");
      document.write("<h4>Name : " + stud2.name+"</h4>");
      document.write("<h4>roll_no : " + stud2.roll_no+"</h4>");
      document.write("<h4>city : " + stud2.city+"</h4>");
    </script>
  </head>
  <body>
  </body>
</html>
```

Output



JSON with Arrays

JSON array represents the collection of values. It is denoted within [ ]. The elements of array are separated by comma.

Syntax

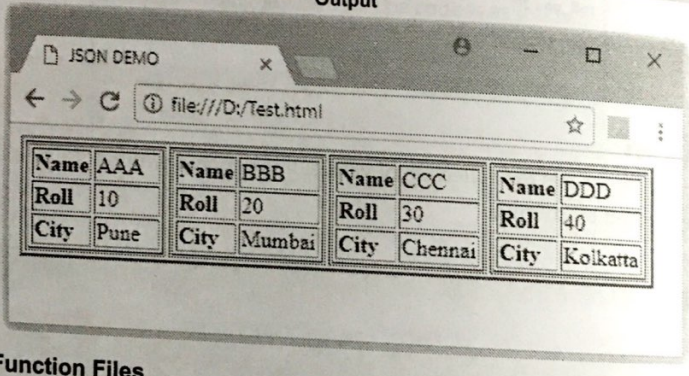
```
[ value, .....]
```

Example

```
<html>
  <head>
   <title>JSON DEMO</title>
   <script language = "javascript" >
    var obj = {"Student":[
        { "name":"AAA", "roll_no":10, "city":"Pune"},
        { "name":"BBB", "roll_no":20, "city":"Mumbai"},
        { "name":"CCC", "roll_no":30, "city":"Chennai"},
        { "name":"DDD", "roll_no":40, "city":"Kolkatta"}
        ]};
var i = 0
      document.writeln("<table border = '2'><tr>");
   for(i = 0;i<obj.Student.length;i++)
   {
        document.writeln("<td>");
        document.writeln("<table border = '1' width = 100 >");
        document.writeln("<tr><td><b>Name</b></td><td width = 50>"+
```

```
        obj.Student[i].name+"</td></tr>");
        document.writeln("<tr><td><b>Roll</b></td><td width = 50>"
obj.Student[i].roll_no
        +"</td></tr>");
      document.writeln("<tr><td><b>City</b></td><td width = 50>" + obj.Student[i].city
      +"</td></tr>");
      document.writeln("</table>");
      document.writeln("</td>");
    }
  </script>
 </head>
 <body>
 </body>
</html>
```

Output



3.19.2 Function Files

Using JSON, it is possible to define a function in a separate external JS file and we can access this functionality from HTML document.

Following example illustrates this idea

Example Code

Step 1 : We can write a JavaScript file containing an array of Student's information. This file is saved with .js extension. The code for it is as follows -

myFile.js

```
Student([
      { "name":"AAA", "roll_no":10, "city":"Pune"},
      { "name":"BBB", "roll_no":20, "city":"Mumbai"},
      { "name":"CCC", "roll_no":30, "city":"Chennai"},
      { "name":"DDD", "roll_no":40, "city":"Kolkatta"}
])
```

Step 2 : Now we will invoke this file in our JSON script to access the elements of an array. The code is as follows -

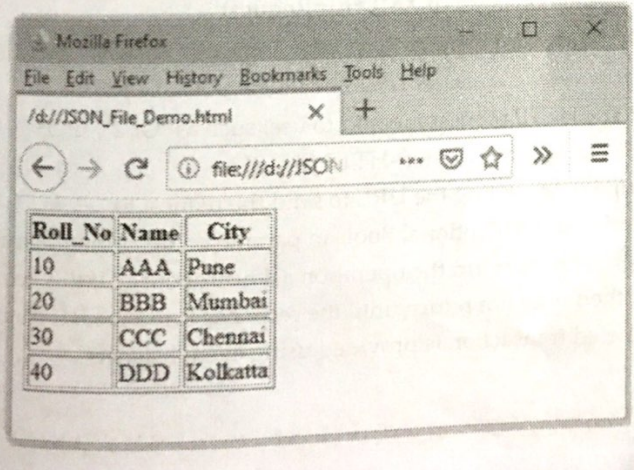
JSON_File_Demo.html

```
<!DOCTYPE html>
<html>
<body>

<div id="myID"></div>
<script>
function Student(arr) {
 var out = "";
 var i;
 out+='<table border=1>'
 out+='<tr><th>Roll_No</th><th>Name</th><th>City</th></tr>'
 for(i = 0; i<arr.length; i++) {
 out += '<tr><td>'+arr[i].roll_no + '</td><td>' +arr[i].name + '</td><td>' +arr[i].city +
'</td></tr>';
 }
 out+='</table>'
 document.getElementById("myID").innerHTML = out;
}
</script>
<script src="myFile.js"></script> // Invoking external file

</body>
</html>
```

Output



Script Explanation :

In above script,

1. We have written the Student function in a file named **MyFile.js**
2. This function defines an array of three fields Roll_No, Name and City. There are four elements in that array
3. In our HTML file, the JSON code refers to the external JS file as shown below

```
<script src="myFile.js"></script>
```

4. Then the elements of array are accessed and displayed on the browser as follows

```
for(i = 0; i<arr.length; i++) {
    out += '<tr><td>'+arr[i].roll_no +
    '</td><td>' +arr[i].name +
    '</td><td>' +arr[i].city + '</td></tr>';
}
```

3.19.3 HTTP Request

- JSON is most commonly used in asynchronous HTTP requests. This is where an application pulls data from another application via an HTTP request on the web.
- **XMLHttpRequest** is an API that provides scripted client functionality for transferring data between a client and a server.
- It allows you to get data from an external URL without having to refresh the page

XMLHttpRequest includes a number of methods and attributes.

- We use two functions of XMLHttpRequest - **Open()** and **send()**.
- The **open()** to initialize the request, and **send()** to send the request.
- The Syntax of method **open of XMLHttpRequest** is

```
XMLHttpRequest.open(method, url[, async[, user[, password]]])
```

where

- **Method :** The HTTP request method to use, such as "GET", "POST", "PUT", "DELETE", etc. Ignored for non-HTTP(S) URLs.
- **url :** A string representing the URL to send the request to.
- **async (Optional) :** An optional Boolean parameter, defaulting to true, indicating whether or not to perform the operation asynchronously. If this value is false, the send() method does not return until the response is received. If true, notification of a completed transaction is provided using event listeners.

- Following code illustrates this idea

Step 1 : Create a text file as

myfile.txt

```
[
    { "name":"AAA", "roll_no":10, "city":"Pune"},
    { "name":"BBB", "roll_no":20, "city":"Mumbai"},
    { "name":"CCC", "roll_no":30, "city":"Chennai"},
    { "name":"DDD", "roll_no":40, "city":"Kolkatta"}
]
```

Step 2 : The JSON code to invoke above text file using Http Request is as follows -

JSON_HttpReq_Demo.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>

<div id="myID"></div>
<script>

var xmlhttp = new XMLHttpRequest();
var url = "file:myfile.txt";

xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var stud_arr = JSON.parse(this.responseText);
        Student(stud_arr);
    }
};

xmlhttp.open("GET", url, true);
xmlhttp.send();

function Student(arr) {
```

```
var out = "";
var i;
out+='<table border=1>'
out+='<tr><th>Roll_No</th><th>Name</th><th>City</th></tr>'
for(i = 0; i<arr.length; i++) {
out += '<tr><td>'+arr[i].roll_no + '</td><td>' +arr[i].name + '</td><td>' +arr[i].city +
'</td></tr>';
}
out+='</table>'
document.getElementById("myID").innerHTML = out;
}
</script>
</body>
</html>
```

The output as obtained in previous section is displayed on the browser.

❑❑❑