Objects are Variables

JavaScript variables can contain single values:

Example

```
var person = "John Doe";
```

Objects are variables too. But objects can contain many values.

The values are written as **name : value** pairs (name and value separated by a colon).

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Example

```
<!DOCTYPE html>
<html>
<body>
<script>
let school = {
    name : "City School",
    location : "Delhi",
    established : 1995,
    20 : 1000,
    displayinfo : function() {
        document.write(this.name + " was established in "
                    + this.established + " at " + this.location + '<br>')
    }
}

school.displayinfo()
// Outputs : 1000
// using the dot notation will result in an error
document.write(school['20']) ;

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

OBJECT

JavaScript is an Object Oriented Programming (OOP) language. A programming language can be called object-oriented if it provides four basic capabilities to developers –

- **Encapsulation** – the capability to store related information, whether data or methods, together in an object.
- **Aggregation** – the capability to store one object inside another object.
- **Inheritance** – the capability of a class to rely upon another class (or number of classes) for some of its properties and methods.
- **Polymorphism** – the capability to write one function or method that works in a variety of different ways.

Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object, otherwise the attribute is considered a property.

Object Properties

Object properties can be any of the three primitive data types, or any of the abstract data types, such as another object. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that are used throughout the page.

The syntax for adding a property to an object is –

```
objectName.objectProperty = propertyValue;
```

For example – The following code gets the document title using the **"title"**property of the **document** object.

```
var str = document.title;
```

Object Methods

Methods are the functions that let the object do something or let something be done to it. There is a small difference between a function and a method – at a function is a standalone unit of statements and a method is attached to an object and can be referenced by the **this** keyword.

Methods are useful for everything from displaying the contents of the object to the screen to performing complex mathematical operations on a group of local properties and parameters.

For example – Following is a simple example to show how to use the **write()** method of document object to write any content on the document.

```
document.write("This is test");
```

User-Defined Objects

All user-defined objects and built-in objects are descendants of an object called **Object**.

The new Operator

The **new** operator is used to create an instance of an object. To create an object, the **new** operator is followed by the constructor method.

In the following example, the constructor methods are Object(), Array(), and Date(). These constructors are built-in JavaScript functions.

```
var employee = new Object();

var books = new Array("C++", "Perl", "Java");

var day = new Date("August 15, 1947");
```

The Object() Constructor

A constructor is a function that creates and initializes an object. JavaScript provides a special constructor function called **Object()** to build the object. The return value of the **Object()** constructor is assigned to a variable.

The variable contains a reference to the new object. The properties assigned to the object are not variables and are not defined with the **var** keyword.

Example 1

```
<html>

   <head>

      <title>User-defined objects</title>

      <script type = "text/javascript">

         var book = new Object();   // Create the object

         book.subject = "Perl";     // Assign properties to the object

         book.author  = "Mohtashim";

      </script>

   </head>


   <body>

      <script type = "text/javascript">

         document.write("Book name is : " + book.subject + "<br>");

         document.write("Book author is : " + book.author + "<br>");

      </script>

   </body>
```

```
</html>
```

Example 2

```
<html>
   <head>
   <title>User-defined objects</title>
      <script type = "text/javascript">
         function book(title, author) {
            this.title = title;
            this.author  = author;
         }
      </script>
   </head>

   <body>
      <script type = "text/javascript">
         var myBook = new book("Perl", "Mohtashim");
         document.write("Book title is : " + myBook.title + "<br>");
         document.write("Book author is : " + myBook.author + "<br>");
      </script>
   </body>
</html>
```

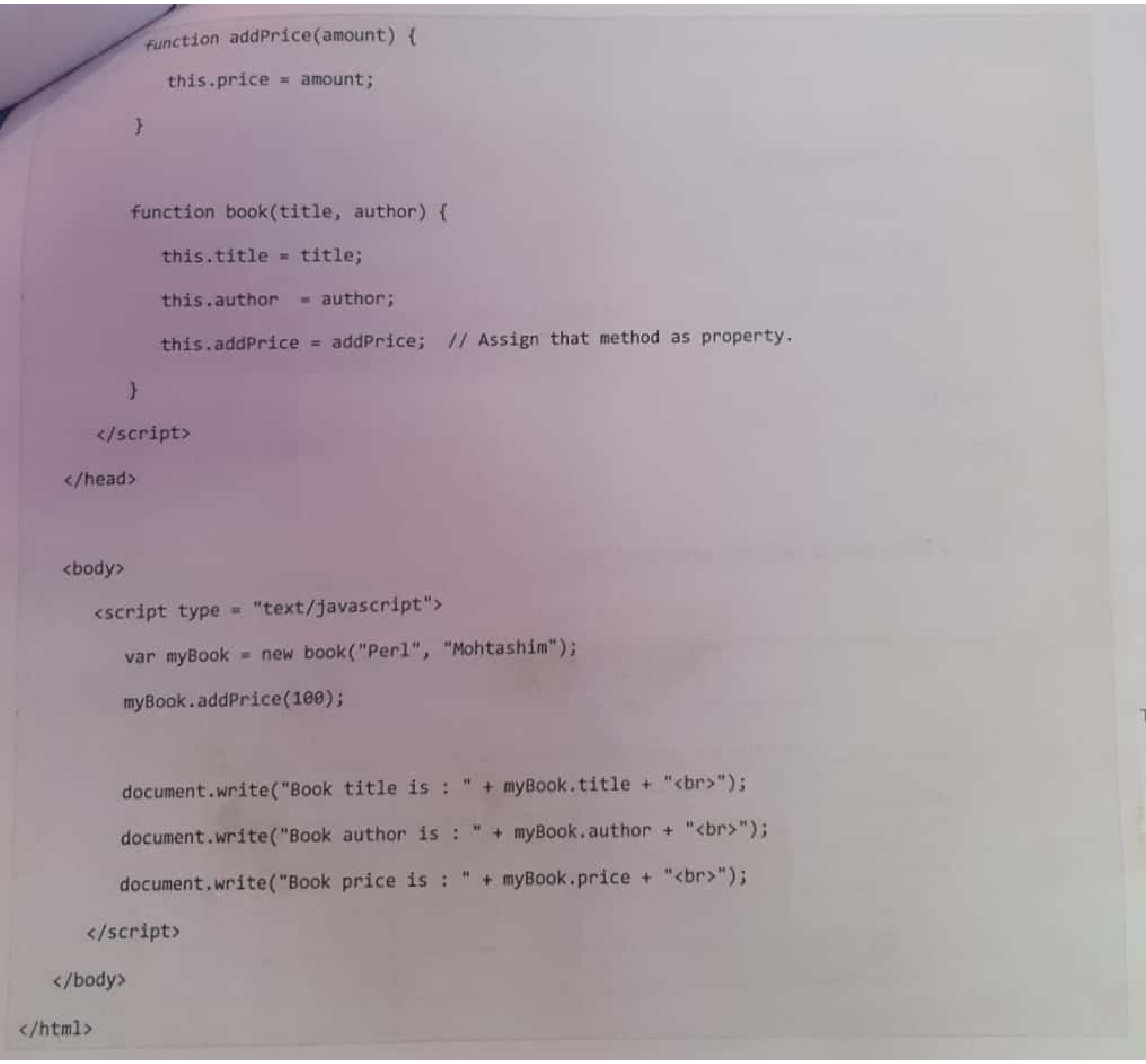
Defining Methods for an Object

The previous examples demonstrate how the constructor creates the object and assigns properties. But we need to complete the definition of an object by assigning methods to it.

Example

```
<html>

   <head>
   <title>User-defined objects</title>
      <script type = "text/javascript">
         // Define a function which will work as a method
```



```
         function addPrice(amount) {
            this.price = amount;
         }

         function book(title, author) {
            this.title = title;
            this.author  = author;
            this.addPrice = addPrice;  // Assign that method as property.
         }
      </script>
   </head>

   <body>
      <script type = "text/javascript">
         var myBook = new book("Perl", "Mohtashim");
         myBook.addPrice(100);

         document.write("Book title is : " + myBook.title + "<br>");
         document.write("Book author is : " + myBook.author + "<br>");
         document.write("Book price is : " + myBook.price + "<br>");
      </script>
   </body>
</html>
```

1.22 *JavaScript Objects*

JavaScript is an object-based language. Everything is an object in JavaScript. Your webpage document is an object. Any table, form, button, image or link on your page is also an object. It is possible to use **built-in objects** available in JavaScript. It is also possible for a JavaScript programmer to define his own objects and variable types.

A JavaScript object is an entity having state and behavior (properties and method). Therefore, each object has **Properties** and **Methods** associated with it.

Properties of an Object

Property is the value that is tagged to the object. For example, **Length** property of the **string** object returns the length of the string, that is in other words the number of characters present in the string.

Syntax of using the length property of the string object is :

```
variablename.length
```

where **variablename** is the name of the variable to which the string is assigned and length is the keyword.

Example

```
<html>
   <body>
         <script  type="text/javascript">
           var mess="Welcome"
           document.write(mess.length)
         </script>
   </body>
</html>
```

Output :

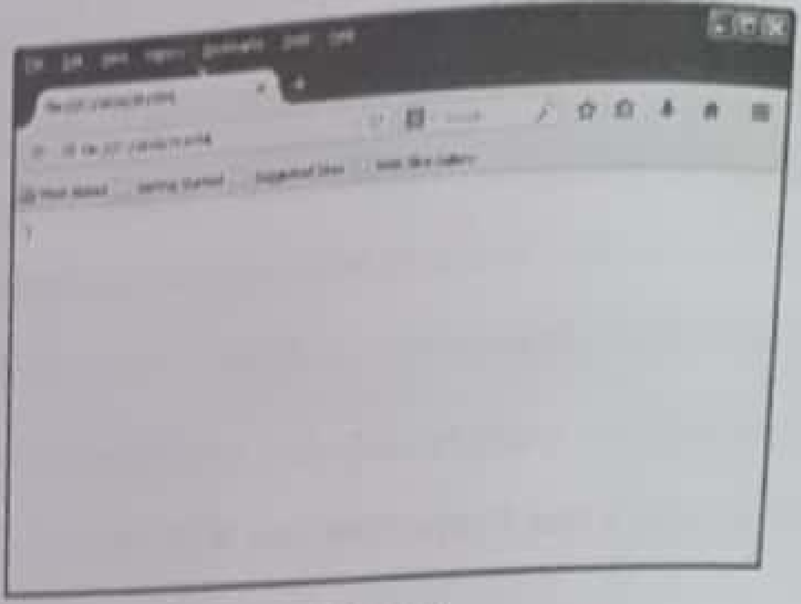


FIGURE 1.12

Note :

Property of an object is the value associated with the object.

Method of an object

Method of an object refers to the actions that can be performed on the objec
example, in String Object there are several methods available in JavaScript

```
toUpperCase()
toLowerCase()
substring()
```

Example

Consider the following code :

```
<html>
   <body>
         <script  type="text/javascript">
           var mess="Welcome"
           document.write(mess.toUpperCase())
         </script>
   </body>
</html>
```

Output :

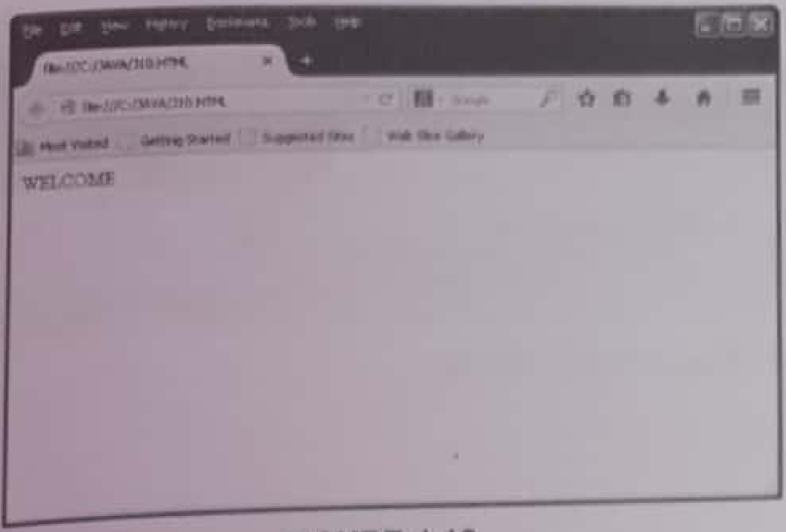

WELCOME

FIGURE 1.13

In this example, the method **toUpperCase** is applied to the string object **mess** which has value initialized as **Welcome** all letters get converted as upper case and hence the output is as above.

.22.1 *Creating Objects in JavaScript*

There are three ways to create objects in JavaScript.

.22.1.1 *By Object Literal*

The syntax of creating object using object literal is

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

Note that property and value is separated by : (colon).

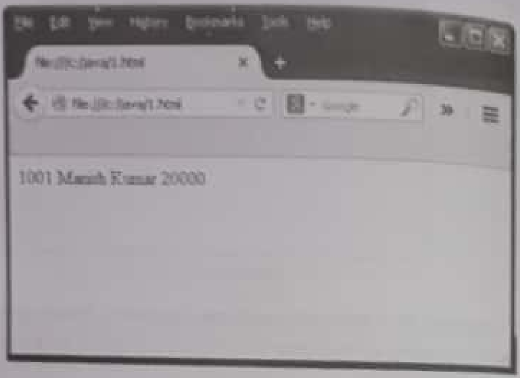
Example

Consider the following code :

```
<HTML>
   <BODY>
         <script type="text/javascript">
            emp={id:1001,name:"Manish Kumar",salary:20000}
            document.write(emp.id+" "+emp.name+" "+emp.salary);
          </script>
      </BODY>
   </HTML>
```

Scanned by CamScanner

Output :



```
1001 Manish Kumar 20000
```

FIGURE 1.14

1.22.1.2 By Creating Instance of Object

The syntax of creating object directly is

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

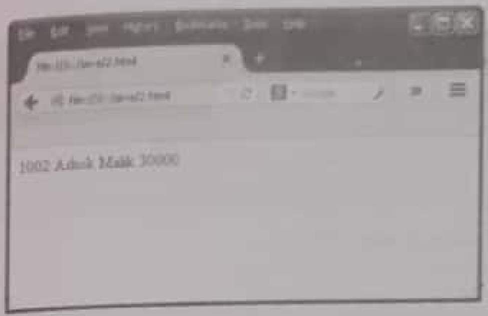
Here, **new keyword** is used to create object.

Example

Consider the following code :

```
<HTML>
  <BODY>
        <script  type="text/javascript">
           var emp=new Object();
           emp.id=1002;
           emp.name="Ashok Malik";
           emp.salary=30000;
           document.write(emp.id+" "+emp.name+" "+emp.salary
         </script>
    </BODY>
    </HTML>
```

Output :



```
1002 Ashok Malik 30000
```

FIGURE 1.15

1.22.1.3 By Using an Object Constructor

Here, you need to create function with argument. Each argument value can be assigned in the current object by using **this** keyword. The **this keyword** refers to the current object.

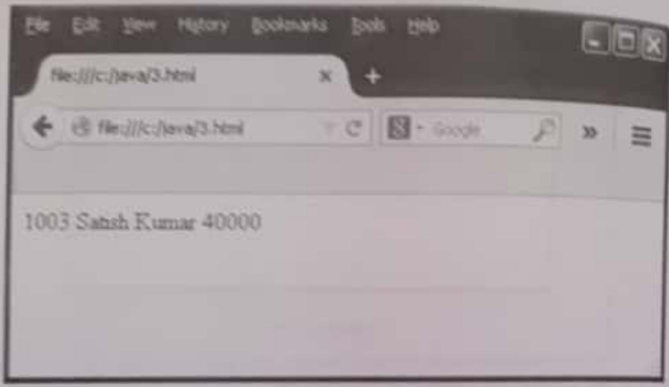
Example

Consider the following code :

```
<HTML>
  <BODY>
        <script  type="text/javascript">
           function emp(id,name,salary)
           {
              this.id=id;
              this.name=name;
              this.salary=salary;
           }
           e=new emp(1003,"Satish Kumar", 40000);

           document.write(e.id+" "+e.name+" "+e.salary);
         </script>
    </BODY>
  </HTML>
```

Output :



```
1003 Satish Kumar 40000
```

FIGURE 1.16

1.22.2 Use of Predefined Object and Methods

In JavaScript, it is possible to use predefined objects. Core predefined objects are

(i) Math

(ii) Number

(iii) String

(iv) Boolean

(v) Date

(vi) Array

1.22.2.1 Math Object

The predefined Math object has properties and methods for mathematical constants and functions. For example, the Math object's **PI** property has the value of pi (3.141 which you would use in an application as

```
Math.PI
```

Similarly, all methos of **Math** can be called by using Math as an object, without creating it. For example, if you want to find the square root of a number 25, you would write

```
Math.sqrt (25);
```

Note that all trigonometric methos of **Math** take arguments in radians.

The following table summarizes the Math Object's methods.

TABLE

Method	**Description**
abs (x)	Returns Absolute value of x
sin(x), cos(x), tan(x)	Standard trigonometric functions; argument in radians
acos(x), asin(x), atan(x), atan2(x)	Inverse trigonometric functions; return values in radians
exp(x)	Returns the value of e^x
log(x)	Returns natural logarithm, base e, of x
ceil(x)	Returns least integer greater than or equal to x i.e. round upwards
floor(x)	Returns greatest integer less than or equal to x i.e. round downwards
min(x,y,z,--)	Returns lowest value number
max(x,y,z,--)	Returns highest value number
pow(x,y)	Return the value of x to the power of y
random( )	Returns a random number between 0 and 1.
round(x)	Rounds argument to nearest integer
sqrt(x)	Returns Square root of x

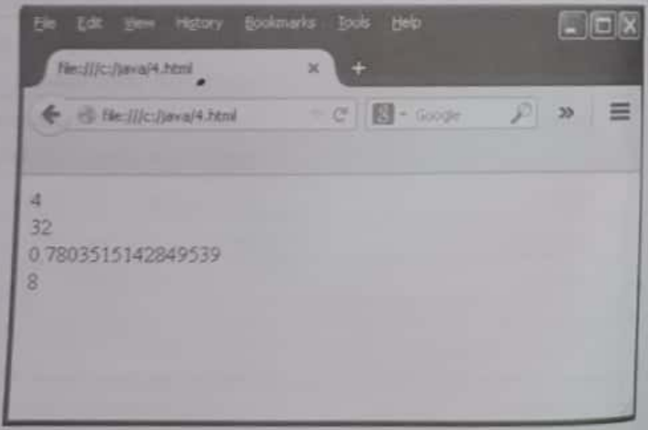
Example

Consider the following code :

```
<HTML>
  <BODY>
        <script  type="text/javascript">
            document.write(Math.sqrt(16)+"<br>");
            document.write(Math.pow(2,5)+"<br>");
            document.write(Math.random()+"<br>");
            document.write(Math.max(2,8,5.25)+"<br>");

        </script>
   </BODY>
</HTML>
```

Output :



```
4
32
0.7803515142849539
8
```

FIGURE 1.17

1.22.2.2 *Number Object*

In JavaScript, numbers can be written with or without decimal point. Extra large or extra small numbers can be written with scientific (exponent) notation.

Example

```
var x = 5.432
var y = 4
var z = 25e8
```

The **Number** object has properties for numerical constants, such as maximum value, not-a-number, and infinity. Properties of Number are shown in Table.

TABLE

Property	Description
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Return the smallest number possible in JavaScript
NaN	Special "not a number" value
NEGATIVE_INFINITY	Special infinite value; returned on overflow
POSITIVE_INFINITY	Special negative infinite value; returned on overflow

Example

```
biggest  = Number.MAX_VALUE
Smallest  = Number.MIN_VALUE
```

The Number prototype provides methods for retrieving information from Number objects in various formats. The following table summarizes the methods of **Number.prototype**.

TABLE

Method	Description
toExponential(x)	Returns a string representing the number in exponential notation.
toFixed(x)	Formats a number with x number of digi after the decimal point
toPrecision(x)	Returns a string representing the numbe to a specified precision in fixed-point nota
toString( )	Convets a number to string
valueOf( )	Returns the primitive value of a number

1.22.2.3 String Object

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

There are two ways to create string in JavaScript.

(i) **By String Literal**

The string literal is created using double quotes. Syntax is

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

Example

```
var name = "Amit Kumar";
```

(ii) **By String Object**

The syntax of creating string object using new keyword is :

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

Example

```
var name = new String("Amit Kumar");
```

A **string** object has one property, **length**, that indicates the number of characters in the string.

Example

```
name = "Amit Kumar"

x = name.length
```

The above code assigns value 10 to x.

The following table summarizes the methods of **string** objects.

TABLE

Property	Description
charAt(pos)	Returns the character at the specified position in string
charCodeAt(pos)	Return the ASCII value of character at the specified position in string
indexOf(str)	Return the index position of the given string
trim( )	Removes leading and trailing whitespaces from the string
lastIndexOf(str)	Returns the last index position of specified substring
concat(str)	Combines the text of two strings and returns a new string
slice (beginIndex, endIndex)	Extracts a section of an string from given beginIndex to endIndex and return a new sring
toLowerCase	Return the string in all lowercase letters
toUpperCase	Return the string in all uppercase letters

Example

Consider the following code :

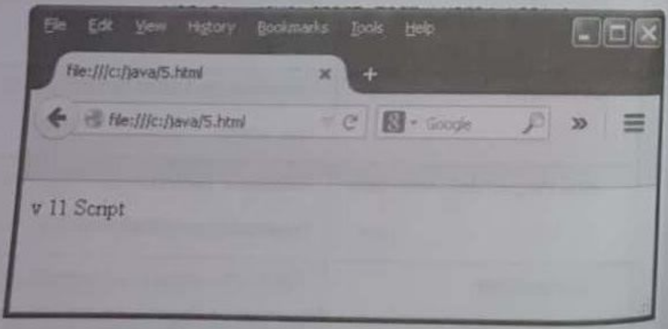
```
<HTML>
  <BODY>
        <script type="text/javascript">
          var s1="JavaScript from Sushil Goel";
          document.write(s1.charAt(2) + " ");

          var n=s1.indexOf("from");
          document.write(n + " ");

          var s2=s1.slice(4,10);
          document.write(s2 + " ");

        </script>
  </BODY>
</HTML>
```

Output :



```
v 11 Script
```

FIGURE 1.18

1.22.3 Boolean Object

JavaScript Boolean is an object that represents value in two states : true or false.

Syntax to create the JavaScript Boolean object is :

```
Boolean b=new Boolean(value);
```

The default value of Boolean object is false.

JavaScript Boolean Methods are :

Method	Description
toSource()	Returns the source of Boolean object as a string
toString()	Converts Boolean into String
valueOf()	Converts other type into Boolean

1.22.3.1 Date Object

The **Date** object has a large number of methods for setting, getting and manipulating dates and time.

The **Date** object rante is -100,000,000 days to 100,000,000 days relative to 01 January, 1970 UTC. Statement to create a **Date** object is :

```
dateObjectName = new Date ([parameters])
```

where **dateObjectName** is the name of the **Date** object being created; it can be a new object or a property of an existing object.

The **parameters** can be any of the following :

- **Nothing :** creates today's date and time.

  For example :

  today = new Date()

- A string representing a date in the following form : "Month day, year hours:minutes:seconds."

  For example :

  birthday = new Date("December 25, 1995 13:30:00")

  If you omit hours, minutes, or seconds, the value will be set to zero.