

4.1 Document Object Model (DOM)

- Document Object Model (DOM) is a standard object model and programming interface for manipulating HTML and XML documents.
- It defines a standard for accessing documents.
- It is a cross-platform and language-neutral interface
- It dynamically accesses and updates the content, structure, and style of a document.
- When a web page is loaded, the browser constructs a DOM of that page. In this DOM, a document is represented as a tree. This tree is a hierarchy of nodes or elements. Using this hierarchy, any HTML element in the DOM is accessible.
- In DOM, Tags are called nodes or elements.

Look at the following HTML document:

```
<html>
  <head>
    <title> DOM</title>
  </head>
  <body>
    <h5> this is demo </h5>
    <p> demo of DOM </p>
  </body>
</html>
```

The DOM representation of the above HTML document is:

In this DOM tree, the document is the always root node. The root node has one child, <html> element. After that < head> and < body> are its children. <title> is child of <head>. In this way, entire DOM is represented.

4.1.1 DOM Properties

Following is the list of DOM properties. DOM properties are accessed using the “document” object. Syntax for using property is *document.property_name*

Properties	Description
links	Refers to all <a> and <area> elements that have a “href” attribute set.
anchors	Refers to all <a> elements that have a “name” attribute set.
images	Refers to all elements
doctype	Specify the document's doctype
scripts	Refers to all <script> elements
head	Refers to the <head> element
forms	Refers to all <form> elements
cookie	Returns all name/value pairs of cookies in the document.
domain	Refers to the domain name of the server.
lastModified	Get the date and time of the last updated document.
readyState	Get the status of the document.
title	It sets or gets the title of the document. i.e. Refers to <title> element.
URL	Specify the URL of the document.

4.1.2 DOM Methods

Following is the list of DOM methods. DOM methods are accessed using the “document” object. Syntax for using method is *document.method_name_name(parameter)*

Methods	Description
write()	It writes the output of JavaScript code or HTML expressions to a

	document.
writeln()	It is the same as write(), but adds a newline character after each statement.
getElementById()	It refers to the element by using its ID
getElementsByName()	It returns a collection containing all the elements having a specified name.
getElementsByTagName()	It returns a collection containing all elements with a specified tag name.
open()	It is used to open an HTML output stream to collect the output from document.write().
close()	It is used to close the output stream previously opened with document.open().

Example:

Following example demonstrate various properties and methods of DOM.

```
<html>
  <title> Demo of DOM properties and methods </title>
  <body>
    
    <p id="test" />
    <input id="input1" name="in" value="shaurya">
    <p id="test1"/>
    <script>
      //using title
      document.write("Document title: " + document.title);
      document.write("<br>");

      //using image
      document.write("<br>");
      document.write("Total no. of images : " + document.images.length);
      document.write("<br>");

      //Using getElementById
      doc = document.getElementById("input1");
```

```
document.write("value of input element: " +doc.value);

//using getElementByName
doc1 = document.getElementsByName("in");
document.write("Element having name=in: " + doc1[0].tagName);

//Using getElementByTagName
document.write("<br>");
document.write("Total no. p elements: " +
document.getElementsByTagName("p").length);
</script>

</body>
</html>
```

Output:

Document title: Demo of DOM properties and methods
Total no. of images : 1
value of input element: shaurya
Element having name=in: INPUT
Total no. p elements: 2

4.2 JavaScript Object

JavaScript supports Object-Oriented concept. In JavaScript, an Object is a non-primitive data type. The object has properties and methods. Properties have values of primitive data types and methods are functions. It is a set of named value pairs.

Ways to create JavaScript Objects

There are 3 ways to create objects in JavaScript.

- (1) By object literal
- (2) By creating an instance of Object
- (3) By using an object constructor

(1) By object literal

The object literal is an easy way of creating an object. It is created using { ...} brackets. The syntax of creating an object with object literal is:

```
var object_name = {property1:value1, property2:value2, ..., propertyN:valueN};
```

Parameter list values are written as **<name : value>** pairs, where name and value are separated by a colon(:) and each pair is separated by comma(,).

Above syntax can be written as:

```
var object_name = {  
    property1:value1,  
    property2:value2  
    , ...,  
    propertyN:valueN };
```

Example: The following is the example of creating an object using object literal.

```
var student = {name:"Shubh",  
    age:11,  
    marks:95};  
//accessing property  
document.write( student.name+" is "+student.age+" years old");
```

Explanation: Here object **student** is created with three properties.

- The first property is "name" and the value is "Shubh".
- The second property is "age" and the value is 11.
- Third property is "marks" and the value is 95.

(2) By creating an instance of Object

The second way to create an object is using **new** keyword with Object(). The new operator is used to create an instance of an object. Dot (.) and [] brackets can be used to specify properties and methods.

Syntax:

```
var object_name = new Object(); // create object  
object_name.property_name = property_value ; // specify properties using dot.  
object_name .["property_name"] = property_value ; // specify properties using [ ]
```

Example:

```
// create object student  
student=new Object();  
//assign properties to the object student  
student.name = "Shubh";  
student["age"]=11;  
student.marks = 95;
```

(3) By using an Object constructor

This is another way to create an object. Here, first, create a function with parameters. Then, assign value to each parameter using **this** keyword. **this** keyword refers to the current object.

Syntax:

```
function function_name(parameter1, parameter2,..., parameterN) { // create function
    //assign value to parameters
    this. parameter1= value1;
    this. parameter2= value2;
    .....
    this. parameterN= valuen;
}
// create object using constructor
object_name = new function_name(value1, value2,...,valuen);
```

Example:

Following example creates object student, and assign "Shubh" to name, "11" to age, and "95" to marks.

```
function stud (name,age, marks) {
    this. name = name;
    this.age=age;
    this.marks=marks;
}
student = new stud("Shubh", 11, 95);
```

The following example demonstrates how to set and get values of Object:

```
<script>

    function stud (name,age, marks) {
        this. name = name;
        this.age=age;
        this.marks=marks;
    }
    student = new stud("Shubh", 11, 95);
    document.writeln("Name:" + student.name);
    document.writeln("Age:" + student.age);
    document.writeln("Marks:" + student.marks);
</script>
```

Output:

Name:Shubh

Age:11

Marks:95

4. 3 Date object

The purpose of the Date object is to work with date (days, months, years) and time (milliseconds, seconds, minutes, hours)

Date object is created using **new Date()**. Following code create a date object.

```
var dateObj = new Date(); // create date object dateObj
```

4.3.1 Date constructor:

There are four basic forms for the Date() constructor:

- (1) **new Date()** : This is a default constructor with no argument. It initializes the Date object with the current date and time.
- (2) **new Date(milliseconds)**: This parameter is an integer value representing the date in milliseconds since January 1, 1970.

Example:

```
dateObj = new Date(999999);
```

- (3) **new Date(dateString)**: A dateString parameter is string value representing a date, in the format accepted by the Date.parse() method.

Example:

```
str = "10/11/1975";
```

```
dateObj = new Date(str);
```

- (4) **new Date(year, month, day [, hours, minutes, seconds, milliseconds])**: This constructor take 3 to 7 parameters. All these parameters are integer value. Description of these parameters is:

Parameter	Description
year	specify the year
month	Specify the month, ranging from 0 (for January) to 11 (for December)
day	Specify the day of the month. Its default value is 1.
hours	Specify the hour of the day in 24-hour format. Its default value is 0.
minutes	Specify the minute part of a time
seconds	Specify the second part of a time

milliseconds	Specify the millisecond part of a time
--------------	--

Example:

Following code create date object birth_date using 7 parameters.

```
birth_date = new Date(1981, 0, 18, 16, 45, 30, 0);
```

4.3.2 Date methods:

Following is the list of Date methods.

Method	Description
getDate()	Returns the day (from 1 to 31) of the month.
getDay()	Returns the day of the week, value is from 0 (Sunday) to 6 (Saturday)
getMonth()	Returns the month, value is from 0 (January) to 11 (December)
getHours()	Returns the hour, value is from 0 to 23
getMinutes()	Returns the minutes
setDate()	sets the day of the month of a given date object
setHours()	Sets the hour of a given date object
setMonth()	Sets the month of a given date object
setMinutes()	Sets the minutes
toString()	Converts a Date object to a string value
toDateString()	Returns the date portion of a Date object

Examples:

Following example demonstrate various methods of Date object. Set birth_date= new Date(); You will get current date and time information.

```
<script>
    birth_date= new Date(1981, 0, 18, 16, 45, 30, 0); //set
    date
    document.write("day of month: "+birth_date.getDate());
    document.write("<br>");
    document.write("hour: "+birth_date.getHours());
    document.write("<br>");
    document.write("minutes:"+birth_date.getMinutes());
    document.write("<br>");
    document.write("second:"+birth_date.getSeconds());
    document.write("<br>");
    document.write("day      of      the      week:"+
    birth_date.getDay());
    document.write("<br>");
    document.write("month:"+birth_date.getMonth());
    document.write("<br>");
    birth_date.setDate(20); // set day to 20
</script>
```

Output:

```
Day of month: 18
hour: 16
minutes:45
second:30
day of the week:0
month:0
```

Following example demonstrate use of setDate method. It is also used to add days to a date:

```
<script>
    dateObj = new Date();
    dateObj.setDate(10);
    document.write(dateObj.getDate() + "<br>");
    dateObj.setDate(dateObj.getDate() + 15);
    document.write(dateObj.getDate());
</script>
```

Output:

10

25