Subject Code: US03IDBCA04
Subject Title: Web Application Development-III
UNIT-2 Advanced JavaScript–II

TOPICS:-
OBJECT, String Object, Math Object, Date Object, Introduction to Browser Object Model (BOM), Introduction to Document Object Model (DOM), Accessing Form elements, Event handling

OBJECT:-

- JavaScript is an object-based language.
- An object is a collection of properties and methods that are grouped together in a single name.
- Properties contain data and methods perform operations on that data contained in those properties.
- Properties and methods are the core elements from which objects formed in JavaScript.

Creating Objects in JavaScript

In JavaScript, there are three ways to create user-defined objects

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

1. By creating Object using object literal

- In JavaScript, '{}' is represented by the object literal.
- You can add pair of key-value pairs between curly braces to define an object.
- You can follow the syntax below to use the object literal to define objects.
- You can add key-value pairs between curly braces '{}'.
- Each key-value pair is comma separated, and you need to add a colon (:) between the key and value.

Syntax: var <object-name> = { key1: value1, key2: value2,...};

EXAMPLE:-

```
var person = { firstName: "John" }; // create object with single property
var myBook = { title: "Perl", author: "Mohtashim",  pages: 355,}
var myObj = { };    //   create empty object
```

- Object literal is the simplest and the most popular way to create a user-defined object in JavaScript. We can create a user-defined object with several properties by using object literal, as follows:

EXAMPLE:- create an HTML program where we will create an object person and access the various properties and display them on browser's window.

```
<script>
```

```
  var person = {   // create object  person
                   // Declaration of properties of an object person.
      firstName:"John",
      lastName:"Herry",
      age: 25,
      skinColor:"White"
    };
                   // Accessing properties of person object.
     document.write("First name: " +person.firstName+ "<br>");
     document.write("Last name: " +person.lastName+ "<br>");
     document.write("Age: " +person.age+ "<br>");
     document.write("Skin color: " +person.skinColor);
</script>
```

Output:

```
First name: John
Last name: Herry
Age: 25
Skin color: White
```

- In the above code, we have created an object called person, which holds four properties: firstName, lastName, age, and skinColor. The values contained in each property are “John”, “Herry”, 25, and “White” respectively.

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

- Creating objects is using the Object () constructor function using **the new keyword.**
- **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.
- These constructors are built-in JavaScript functions (native object).
- The syntax of creating object directly is given below:

Syntax: - var objectname=new Object();

- In the following example, the constructor methods are Object(), Array(), and Date(),string(),Math() ,function() etc.

EXAMPLE:-

```
//  empty object
var myObj = new Object();
var str = "Hello String!"; // create String  Object
var a = new Date(); //create date Object
var x = new Array(); //create Array Object
var y = new Object(); //create  Object
```

```
var employee = new Object();
var books = new Array("C++", "Perl", "Java");//create Array Object pass value
var day = new Date("August 15, 1947"); //create date Object pass value
```

EXAMPLE:-

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

OUTPUT:- 101 Ravi Malik 50000

3. By using an Object constructor

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

The example of creating object by object constructor is given below.

```
<script>
function emp(id,name,salary)
{
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

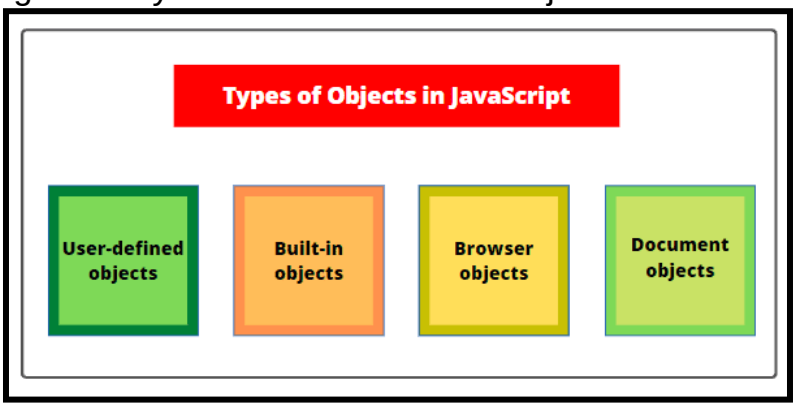
Output of the above example

103 Vimal Jaiswal 30000

Types of Objects in JavaScript

- In JavaScript, there are four types of objects. They are as follows:
  - User-defined objects
  - Built-in objects (such as Date Math , Array, String ,objects)
  - Browser objects (such as window, navigator, and history objects)
  - Document objects (for example, link, forms and images)
- An object is a non-primitive, primitive (structured) data type (String, Array, Date, Math etc) in JavaScript.

- Objects are same as variables in JavaScript; the only difference is that an object holds multiple values in terms of properties and methods.
- All user-defined objects and built-in objects are descendants of an object called Object.
- We can use object literals to create a new user-defined object.
- Alternatively, we can create a constructor function and then invoke this function using new keyword to instantiate an object.



1. User-defined Objects in JavaScript

- User-defined objects in JavaScript are custom objects created by the programmer for specific programming tasks.
- They are associated with properties and methods.
- **For example,** a person is an object. The properties of a person are name, height, weight, age, etc.
- All persons have these properties, but values of those person's properties may differ from person to person.
- Object literal is the simplest and the most popular way to create a user-defined object in JavaScript.
- We can create a user-defined object with several properties by using object literal, as follows:

```
var person = {
  firstName: "John",        // Declaration of properties of an object person.
  lastName: "Herry",
  age: 25,
  skinColor: "White" };
```

NOTE:-In the above code, we have created an object called person, which holds four properties: firstName, lastName, age, and skinColor. The values contained in each property are "John", "Herry", 25, and "White" respectively.

EXAMPLE:- create an HTML program where we will create an object person and access the various properties and display them on browser's window.

Program code 1:

```
<html>
<head>
  <title>User defined Object Example</title>
<script>
                    // create user define of person object
var person = { firstName:"John", lastName:"Herry", age: 25, skinColor:"White" };
                    // Accessing properties of person object.
    document.write("First name: " +person.firstName+ "<br>");
    document.write("Last name: " +person.lastName+ "<br>");
    document.write("Age: " +person.age+ "<br>");
    document.write("Skin color: " +person.skinColor);
</script>
```

Output:

First name: John
Last name: Herry
Age: 25
Skin color: White

2. Built-in Objects in JavaScript

- Built-in objects are native objects that are part of core JavaScript and they are defined in Script standard. JavaScript supports the number of built-in objects.
- These built-in objects are available for both client side JavaScript and server-side applications. Some important built-in objects include:
  - String object
  - Array object
  - Date object
  - Math object

(NOTE: These topics will be detail discussing of **built-in objects** (STRING, DATE,MATHS)in JavaScript in the further topics one by one.)

3. Browser Objects in JavaScript

- Browser objects are those objects that interact with the browser window. These objects are not part of JavaScript language but most browser commonly support them. Example of browser objects are:
  - Window object
  - History object
  - Location object
  - Navigator object
  - Screen object

(NOTE: These topics will be detail discussing of **Browser object** in JavaScript in the next topics one by one.)

4. Document Objects in JavaScript

- Document objects are part of Document Object Model (DOM).
- Every window is connected with document object. It is a container for all HTML objects associated with the HTML tags of an HTML document.
- The document object provides various properties such as anchor, alink Color, cookie, forms, history, etc. and methods such as clear, close, open, etc.
- We can access document objects using document property of the window object provided by the browser. For example, we can change document's background color by setting the document's bgcolor property.

The syntax is as follows: window.document.bgcolor = "red";

(NOTE: These topics will be detail discussing of **Document Object Model (DOM)** in JavaScript in the next topics one by one.)

JavaScript Object Properties

- Object properties (or fields) define the characteristics of an object.
- For instance, a car has a size, model, weight, color, price, and many other attributes. These characteristics are called fields or properties.
- An object's properties simply store data related to an object.
- There are 3 ways to access object properties in JavaScript.
  - Using the dot notation
  - Using the square bracket notation
  - Using the expression
- The general syntax to access an object's properties is as follows:

```
objectName.propertyName;          OR
objectName["propertyName"];       OR
objectName[expression]
```

- Here, object Name is the name of an object, and property Name is the name of property (data filed) of an object. Both are case-sensitive.

1. The Dot Notation

- You can access the object property using the dot notation/ syntax.

Example-1

The 'fruit' object in the example below contains the name and price property. We access the object properties using the dot notation

```
<body>
        <p id = "output"> </p>
```

```
        <script>
                const fruit = {name: "Banana",  price: 20,}
                document.getElementById("output").innerHTML =
                "The price of the " + fruit.name + " is " + fruit.price;
        </script>
</body>
```
Output: The price of the Banana is 20

2. The Square Bracket Notation

- You can use the square bracket pair containing the property as a string followed by the object name to access a particular property.

Example:-

In the example below, we access the fruit object's name and price property values.

```
<body>
        <p id = "output"> </p>
        <script>
                const fruit = {name: "Banana",  price: 20,}
                document.getElementById("output").innerHTML =
    "The price of the " + fruit["name"] + " is " + fruit["price"];
        </script>
</body>
```
Output: The price of the Banana is 20

3. Using the expression inside the bracket

- Sometimes, you require accessing the object properties dynamically using the variable or expression. So, you can write the expression inside the square bracket notation.
- The expression can be a variable, a mathematical expression, etc.

EXAMPLE:-

```
<body>
        <p id = "output"> </p>
        <script>
           const num = {10: "ten",20: "Twenty",}
                const x = 10;
                document.getElementById("output").innerHTML =  num[x + 10];
  </script>
</body>
```
Output: Twenty

JavaScript Object Methods

- Object methods are actions that can be performed on objects.
- A method is a function definition stored as a property value.
- Object methods are a powerful way to add functionality to objects. They allow you to encapsulate code and make it reusable.

Accessing Object Methods

- You access an object method with the following syntax:

```
objectName.methodName()
```

Example:- If you invoke(call) the fullName property with (), it will execute as a function:

```
<body>
<p id="demo"></p>
<script>
// Create an Object
const person ={firstName:"John", lastName:"Doe", id:5566, fullName:function()
{
   return this.firstName + " " + this.lastName;}
};
     document.getElementById("demo").innerHTML = person.fullName();
</script>
</body>
```

OUTPUT:-John Doe

EXAMPLE: If you access the fullName property without (), it will return the function definition:

```
<body>
<p id="demo"></p>
<script>
// Create an Object
const person = { firstName: "John", lastName: "Doe",  id: 5566,  fullName:
function()
{
   return this.firstName + " " + this.lastName;
  }
};
document.getElementById("demo").innerHTML = person.fullName;
</script>
```

OUTPUT:-function() { return this.firstName + " " + this.lastName; }

BUILT-IN OBJECTS IN JAVASCRIPT

- These built-in objects are available for both client side JavaScript and server-side applications. Some important built-in objects include:
  - String object
  - Array object
  - Date object
  - Math object

String Object:-

- The JavaScript string is an object that represents a sequence of characters.
- There are 2 ways to **create string** in JavaScript
  - By string literal
  - By string object (using new keyword)

1) By string literal

- The string literal is created using double quotes. The syntax of creating string using string literal is given below:

Syntax:-var stringname="string value";

EXAMPLE:-

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

Output: This is string literal

2) By string object (using new keyword)

The syntax of creating string object using new keyword is given below:
var stringname=new String("string literal");
Here, new keyword is used to create instance of string.

Let's see the example of creating string in JavaScript by new keyword.

```
<script>
var stringname=new String("hello javascript string");
document.write(stringname);
</script>
```

Output:hello javascript string

String Length Property

- The length property returns the number of characters in a string.
- It counts spaces and punctuation symbol as characters.

Syntax: Objectname(variable name).length
EXAMPLE:-Return the number of characters in a string:

```
<script type="text/javascript">
var a = "apms", b=”fybca”;
alert(a.length);
alert(a.length);
</script>
```

String Methods

- String methods produce a new string without altering the original string.
- String methods are charAt, indexOf, substr, toLowerCase, toUpperCase etc.

1. charAt() Method

- The JavaScript string charAt() method is used to find out a char value present at the specified index in a string.
- The index number starts from 0 and goes to n-1, where n is the length of the string. The index value can't be a negative, greater than or equal to the length of the string.

Syntax: String.charAt(index)

Parameter: index - It represent the position of a character.

Example 1 Let's see a simple example to print 4th & last character..

```
<script>
var str="Javatpoint";
document.write(str.charAt(4));
document.write(str.charAt(str.length-1));
</script>
```

Output:t
t

2. indexOf() method

- The indexOf() method returns the character position of a specified value in a string.
- The character positions start at 0.
- if the search characters no match in string, this method return -1.

Syntax: variable name. index Of (search string, start)

Example:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.indexOf("d") + "<br />");
document.write(str.indexOf("WORLD") + "<br />");
document.write(str.indexOf("world"));
</script>
```

The output of the code above will be: 10,-1, 6

3. substr() Method

- The substr() method extracts(cut) the characters from a string and returns the new sub string.
- These methods specify index position for start point of string value and index position for ending point (length of string) of string value.
- It counts spaces and punctuation symbol as characters.

Syntax: Variable name.substr (start, length)

Example:

```
<script type="text/javascript">
var str="Hello world!";
document.write(str.substr(3)+"<br />");
document.write(str.substr(3,4));
</script>
```

The output of the code above will be:

lo world!
lo w

4. toLowerCase () Method

- The toLowerCase() method converts a string to lowercase letters.

Syntax: variablename.toLowerCase ()

Example:

```
<script type="text/javascript">
var str="Hello World!";
document.write(str.toLowerCase());
</script>
```

5. to Uppercase() Method

- The toUpperCase() method converts a string to uppercase letters.

Syntax: variable name.toUpperCase()

Example:

```
<script type="text/javascript">
 var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

Math Object

- The JavaScript math object provides properties and methods to perform mathematical operation.
- Unlike other global objects, Math is not a constructor.
- All the properties and methods of Math are static and can be called by using Math as an object without creating it.

- **Math PI Property**:
- The PI property returns the ratio of a circle's area to the square of its radius, approximately 3.14159.

Syntax: Math.PI

Example:

```
<script type="text/javascript">
document.write("PI: " + Math.PI);
</script>
```

The output of the code above will be:PI: 3.141592653589793

Math Methods

- Following is the list of methods of Math class in JavaScript −

1. abs() Method:-

- The JavaScript Math.abs () method accepts a number as a parameter and returns the absolute value of the provided number.
- If the provided value is not a valid number or cannot be converted to a number, the result is NaN.
- If the provided value is null, the method returns 0.

Syntax: Math.abs(value)

Example:

```
<script type="text/javascript">
document.write(Math.abs(7.25));
document.write(Math.abs(-7.25));
document.write(Math.abs(null) );
document.write(Math.abs("Hello"));
document.write(Math.abs(7.25-10));
</script>
```
The output of the code above will be:7.25 7.25 0 NaN 2.75

2. ceil() Method:-

- The ceil() method rounds a number UPWARDS(round up) to the nearest integer, and returns the result.
- If we pass an empty number or non-numeric value as an argument to this method, it returns "NaN"

Syntax:Math.ceil(value)

Example:
```
<script type="text/javascript">
document.write(Math.ceil(0.60));
document.write(Math.ceil(0.40) );
document.write(Math.ceil(5) );
document.write(Math.ceil(5.1));
document.write(Math.ceil(-5.1));
document.write(Math.ceil(-5.9));
</script>
```
The output of the code above will be:1 1 5 6 -5 -5

3. floor() Method:-

- The floor () method rounds a number DOWNWARDS (round down) to the nearest integer, and returns the result.
- If we pass an empty number or non-numeric value as an argument to this method, it returns "NaN"

Syntax: Math.floor(value)

Example:
```
<script type="text/javascript">
document.write(Math.floor(0.60));
document.write(Math.floor(0.40) );
```

```
document.write(Math.floor(5));
document.write(Math.floor(5.1));
document.write(Math.floor(-5.1));
document.write(Math.floor(-5.9));
</script>
```
The output of the code above will be:0 0 5 5 -6 -6

4. max() Method:-

- The max() method returns to find the number with the highest(maximum) value in the list.

Syntax: Math.max(x,y,z,...,n)

Example:
```
<script type="text/javascript">
document.write(Math.max(5,10));
document.write(Math.max(0,150,30,20,38));
document.write(Math.max(-5,10));
document.write(Math.max(-5,-10));
document.write(Math.max(1.5,2.5));
</script>
```
The output of the code above will be:10 150 10 -5 2.5

5. min() Method:-

- The min () method returns to find the number with the lowest (minimum) value in the list.

Syntax: Math.min(x,y,z,...,n)

Example:
```
<script type="text/javascript">
document.write(Math.min(5,10));
document.write(Math.min(0,150,30,20,38));
document.write(Math.min(-5,10));
document.write(Math.min(-5,-10));
document.write(Math.min(1.5,2.5));
</script>
```
The output of the code above will be: 5 0 -5 -10 1.5

6. round() Method:-

- The round () method return the round a number to the nearest integer.
- It round up only if the decimal part is 5 or greater than other wise it return round down

Syntax:Math.round(value)

Example:

```
<script type="text/javascript">
document.write(Math.round(0.60));
document.write(Math.round(0.50) );
document.write(Math.round(0.49) );
document.write(Math.round(-4.40));
document.write(Math.round(-4.60));
</script>
```

The output of the code above will be:1 1 0 -4 -5

Date Object

- The JavaScript date object can be used to get year, month and day.
- You can display a timer on the webpage by the help of JavaScript date object.
- You can use different Date constructors to create date object.

Get methods:-

1. getDate() Method:-

The getDate() method returns the **day of the month** (from 1 to 31) for the specified date, according to local time.

Syntax:Variable name.getDate()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getDate());
</script>
```

The output of the code above will be:

11 (assuming today is 11 September 2024)

2. getDay() Method:-

The getDay() method returns the **day of the week** (from 0 to 6) for the specified date, according to local time.

Syntax:Variable name.getDay()

Example:
```
<script type="text/javascript">
var d = new Date();
document.write(d.getDay());
</script>
```
The output of the code above will be:
3 (assuming today is Wednesday)

3. getFullYear() Method:-

The getFullYear() method **returns the year** (four digits) of the specified date, according to local time

Syntax:Variable name.getFullYear()

Example:
```
<script type="text/javascript">
var d = new Date();
document.write(d.getFullYear());
</script>
```
The output of the code above will be:
2024

4. getMonth() Method:-

The getMonth() method **returns the month** (from 0 to 11) for the specified date, according to local time.

Syntax:Variable name.getMonth()

Example:
```
<script type="text/javascript">
var d = new Date();
document.write(d.getMonth());
</script>
```
The output of the code above will be:0

5. getTime() Method:-

The getTime() method returns the number of **milliseconds** since midnight of January 1, 1970 and the specified date.

Syntax:Variable name.getTime()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getTime() + " milliseconds since 1970/01/01");
</script>
```

Output:

1326898266603 milliseconds since 1970/01/01

6. getHours() Method:-

The getHours() method returns the **hour** (from 0 to 23) of the specified date and time, according to local time.

Syntax:Variable name.getHours()

Example :

```
<script type="text/javascript">
var d = new Date();
document.write(d.getHours());
</script>
```

The output of the code above will be:

20

7. getMinutes() Method:-

The getMinutes() method returns the **minutes** (from 0 to 59) of the specified date and time, according to local time.

Syntax:Variable name.getMinutes()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getMinutes());
</script>
```

The output of the code above will be:

24

8. getSeconds() Method:-

The getSeconds() method returns the **seconds** (from 0 to 59) of the specified date and time, according to local time.

Syntax:Variable name.getSeconds()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getSeconds());
</script>
```

Output:

21

Set methods:-

1. setDate () Method:-

The setDate() method sets the day of the month (1 to 31), according to local time

Syntax:Variable name.setDate(day)

Example:

```
<script type="text/javascript">
var d = new Date();
d.setDate(15);
document.write(d);
</script>
```

The output of the code above will be:

Sun Jan 15 20:30:03 UTC+0530 2012

2. setFullYear() Method:-

The setFullYear() method sets the year (four digits), according to local time.

Syntax:Variable name.setFullYear(year,month,day)

Example:

```
<script type="text/javascript">
var d = new Date();
d.setFullYear(2020);
document.write(d);
</script>
```

The output of the code above will be:

Sat Jan 18 20:31:32 UTC+0530 2020

3. setMonth() Method:-

The setMonth() method sets the month (from 0 to 11), according to local time.

Syntax:Variable name.setMonth(month,day)

Example:

```
<script type="text/javascript">
var d = new Date();
d.setMonth(0);
document.write(d);
</script>
```

The output of the code above will be: Wed Jan 18 20:33:45 UTC+0530 2012

4. setTime() Method:-

The setTime() method sets a date and time by adding or subtracting a specified number of milliseconds to / from midnight January 1, 1970.

Syntax : Variable name.setTime(millisec)

Example :

```
<script type="text/javascript">
var d = new Date();
d.setTime(77771564221);
document.write(d);
</script>
```

The output of the code above will be: Mon Jun 19 05:12:44 UTC+0200 1972

5.setHours() Method:-

The setHours() method sets the hour (from 0 to 23), according to local time.

Syntax:Variable name.setHours(hour,min,sec,millisec)

Example:

```
<script type="text/javascript">
var d = new Date();
d.setHours(15);
document.write(d);
</script>
```

The output of the code above will be:Wed Jan 18 15:36:04 UTC+0530 2012

6. setMinutes() Method:-

The setMinutes() method sets the minutes (from 0 to 59), according to local time.

Syntax :Variable name.setMinutes(min,sec,millisec)

Example:

```
<script type="text/javascript">
var d = new Date();
d.setMinutes(1);
document.write(d);
</script>
```

The output of the code above will be: Wed Jan 18 20:01:59 UTC+0530 2012

7. setSeconds() Method:-

The setSeconds() method sets the seconds (0 to 59), according to local time.

Syntax :Variable name.setSeconds(sec,millisec)

Example:

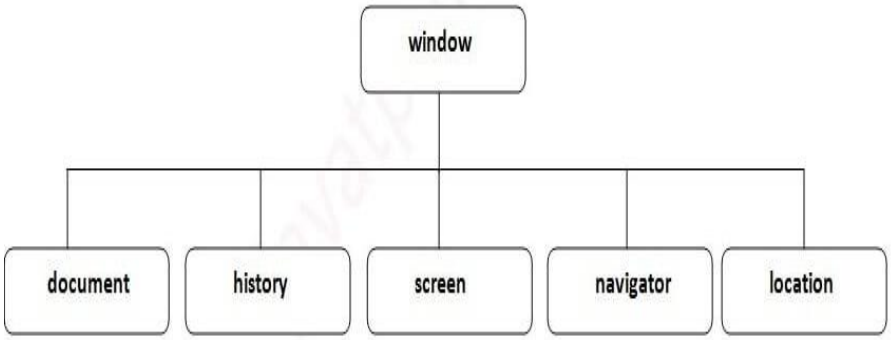
```
<script type="text/javascript">
var d = new Date();
d.setSeconds(1);
document.write(d);
</script>
```

The output of the code above will be: Wed Jan 18 20:39:01 UTC+0530 2012

BROWSER OBJECT MODEL

- The **Browser Object Model (BOM)** in JavaScript refers to the objects provided by the browsers to interact with them. By using these objects, you can manipulate the browser's functionality. For example, you can get the browser history and window size, navigate to different URLs, etc. The Browser object model is not standardized. It depends on which browser you are using. Here, we have listed all objects of the Browser Object Model with descriptions
- **Window −** The 'window' object represents the current browser window. You can use it to manipulate the browser window.
- **Document −** The 'document' object represents the currently opened web page in the browser window. You can use it to customize the property of the document.
- **Screen −** It provides information about the user's device's screen.
- **History −** It provides the browser's session history.
- **Navigator −** It is used to get browser's information like default language, etc.

- **Location –** It is used to get the URL information, such as the hostname of the current web page.



1. Window Object

- The JavaScript window object represents the browser's window. In JavaScript, a 'window' object is a global object. We can use different methods and properties of the window object to manipulate current browser window. For example, showing an alert, opening a new window, closing the current window, etc.
- All the JavaScript global variables are properties of window object. All global functions are methods of the window object.
- The other objects listed above such as document, screen, history etc., are the properties of the window object. We can access these objects as properties of the window object. We can also access them with referencing the window object.

Window Object Properties

- The 'window' object contains the various properties, returning the status and information about the current window.

Property Name	Property Description
document	It is used to access the HTML document opened in the current window.
history	It is used to get the history object of the window.
location	It is used to access the location object of the current window.
navigator	It is used to get the Navigator object of the browser.
screen	It returns the 'screen' object of the current window.

Window Object Methods

- The 'window' object also contains methods like properties to manipulate the current browser window.

Method Name	Method Description
alert()	It is used to show the alert message to the visitors.
confirm()	It shows the confirm box to get the confirmation from users.
prompt()	It allows you to show a prompt box to get user input.

Syntax: - Follow the syntax below to access the 'window'object's properties and methods

```
window.property name;
window.method name();
```

Example-1 of alert() in javascript

```
<script type="text/javascript">
function msg(){
window. alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

Example-2 of confirm() in javascript

```
<script type="text/javascript">
function msg(){
var v= window.confirm("Are u sure?");
if(v==true)
{
alert("ok");
}
else
{
alert("cancel");
}
}
</script>
<input type="button" value="delete record" onclick="msg()"/>
```

Example-3 of prompt() in javascript

```
<script type="text/javascript">
function msg(){
var v= window.prompt("Who are you?");
alert("I am "+v);
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

2. Document Object

- The document object is a property of the JavaScript window object.
- The whole HTML document is represented as a document object. The document object forms HTML DOM.
- The document object is a JavaScript object that provides the access to all elements of an HTML document. When the HTML document is loaded in the web browser, it creates a document object. It is the root of the HTML document.
- The document object contains the various properties and methods you can use to get details about HTML elements and customize them.
- Follow the syntax below to access the document object's properties and methods

Syntax: - document object access with the window object.

window.document.property name;
window.document.method name;

OR

Syntax: - document object access it without using the window object.

document. Property name;
document. Method name;

Document Object Methods

- The JavaScript Document Object provides us with various methods that allow us to interact with and manipulate the HTML document.

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.

getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.

Example-1
One simple way to print "Hello World" is to use document.write() method.

```
<script>
  document.write("Hello World)
</script>
```

Example-2
In the example below, we define a div element with id="output". We access this element using the document.getElementById("output"). Then we change the innerHTML property and display our message, "Hello World".

```
<body>
  <div id = "output"> </div>
  <script>
    document.getElementById("output").innerHTML = "Hello World";
</script>
</body>
```

Document Object Properties

Property	Description
documentElement	To get the <html> element.
forms	It returns an array of <form> elements of the document.
images	It returns the collection of the <img> elements.
links	To get the collection of all <a> elements.

Example-1 of document object that prints name with welcome message.

```
<script type="text/javascript">
function printvalue()
{
var name=document.form1.name.value;
alert("Welcome: "+name);
```

```
}
</script>

<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

3. Screen Object

- The screen object in JavaScript is a property of the 'window' object.
- The screen object has a number of properties that provide information about the device of window screen's such as orientation, resolution, and more. These properties can be used to develop applications that are responsive to different screen sizes and orientations.

Screen Object Properties

Property	Description
availHeight	It gives the height of the screen without including the taskbar.
availWidth	It gives the width of the screen without including the taskbar.
colorDepth	It gives the depth of the color palette to display images.
height	It returns the total height of the screen.
pixelDepth	It is used to get the color resolution of the screen.
width	To get the total width of the screen.

Follow the syntax below to access the screen object's properties and methods

Syntax: screen object access using the window object.

window. screen. property name;
window. screen. method name();

OR

Syntax: screen object access without using the window object.

screen . property name;
screen . method name();

Example: In the example below, we access the various properties of the screen object with referencing the screen as the property of window object

```
<body>
  <p> Screen Information </p>
  <div id = "output"> </div>
  <script>
    document.getElementById("output").innerHTML =
        "screen height: " + window.screen.height + "<br>" +
    "screen width: " + window.screen.width + "<br>" +
    "screen colorDepth: " + window.screen.colorDepth + "<br>" +
    "screen pixelDepth: " + window.screen.pixelDepth + "<br>" +
    "screen availHeight: " + window.screen.availHeight + "<br>" +
    "screen availWidth: " + window.screen.availWidth;
  </script>
</body>
```

4. History Object

- In JavaScript, the history object is also a property of the window object. It contains a list of the visited URLs in the current session. The history object provides an interface to manipulate the browser's session history.
- We can use the methods of history object to navigate the URLs in the history list. For example to go the previous page/URL in the history list, we can use history.back() method.

History Object Property

Property	Description
length	It returns the object's length, representing the number of URLS present in the object.

History Object Methods

Method	Description
back()	It takes you to the previous web page.
forward()	It takes you to the next web page.
go()	It can take you to a specific web page.

Follow the syntax below to access the history object's properties and methods

SYNTAX: - history object can be accessed using window object.
window. history. property name;
window. history. method Name ();

OR

SYNTAX: - history object can be accessed without using window object
history. property;
history.methodName();

Example of history object
history.back();//for previous page
history.forward();//for next page
history.go(2);//for next 2nd page
history.go(-2);//for previous 2nd page

5. Navigator Object

- The JavaScript navigator object is also a property of the window object. Using the 'navigator' object, you can get the browser version and name and check whether the cookie is enabled in the browser.
- There are many properties of navigator object that can be used to access the information about the user's browser.

Navigator Object Properties

- There are many properties of navigator object that can be used to access the information about the user's browser.

Property	Description
appName	It gives you a browser name.
appVersion	It gives you the browser version.
appCodeName	It gives you the browser code name.
cookieEnabled	It returns a boolean value based on whether the cookie is enabled.
language	It returns the browser language.
platform	It gives you a platform or operating system in which the browser is used.

JavaScript Navigator Object Methods

Method	Description
javaEnabled()	It checks whether the Java is enabled in the web browser.

Follow the syntax below to access the navigator object's properties and methods
Syntax: navigator object can be accessed using window object.
window. navigator. property name;
window. navigator. method name ();

OR

Syntax: navigator object can be accessed without using window object.
navigator.property name;
navigator. method name ();

Example
In the example below, we accessed the navigator object as a property of the window object. Then we accessed different properties of this navigator object

```
<body>
  <p> Browser Information</p>
<p id = "demo"> </p>
  <script>
        document.getElementById("demo").innerHTML =
        "App Name: " + window.navigator.appName + "<br>" +
        "App Code Name: " + window.navigator.appCodeName + "<br>" +
        "App Version: " + window.navigator.appVersion + "<br>" +
        "Cookie Enabled: " + window.navigator.cookieEnabled + "<br>" +
        "Language: " + window.navigator.language + "<br>" +
        "platform: " + window.navigator.platform + "<br>" +
          </script>
 </body>
```

Output
Browser Information
App Name: Netscape
App Code Name: Mozilla
App Version: 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36
Cookie Enabled: true
Language: en-US
platform: Win32

6. Location Object

- location object is a property of the window objects.

- The location object in JavaScript provides information about the browser's location, For example, to get the host from the current URL, The 'location' object contains various properties and methods to get and manipulate the information of the browser's location (URL).

Follow the syntax below to access the location object's properties and methods
Syntax:- location object can be accessed using window object.
window.location.property name;
window.location.method name();

OR

Syntax: location object can be accessed without using window object.
location.property;
location.method name();

Location Object Properties

Property	Description
hash	It is used to set or get the anchor part of the URL.
host	It is used to set or get the hostname or port number of the URL.
hostname	It is used to set the hostname.
href	It is used to set or get the URL of the current window.
origin	It returns the protocol, domain, and port of the URL.
pathname	To update or get the path name.
port	To update or get the port of the URL.
protocol	To update or get the protocol.
search	To set or get the query string of the URL.

Location Object Methods List

Method	Description
assign()	To load resources at a particular URL.

reload()	To reload the web page.
replace()	To replace the resources of the current webpage with another webpage's resources.
toString()	Returns the URL in the string format.

Example-1: Accessing location host property

```
<body>
  <div id="output"></div>
  <script>
    const host = location.host;
    document.getElementById("output").innerHTML =
        "The host of the current location is: " + host;
  </script>
</body>
```

Output: The host of the current location is: www.tutorialspoint.com

INTRODUCTION TO DOCUMENT OBJECT MODEL (DOM)

- The document object represents the whole html document.
- When html document is loaded in the browser, it becomes a document object.
- It is the root element that represents the html document.
- It has properties and methods.
- By the help of document object, we can add dynamic content to our web page.
- The HTML DOM allows JavaScript to access and modify the content of HTML elements.
- JavaScript can change all HTML elements, attributes, CSS styles in the page.
- JavaScript can also add, remove the HTML elements and attributes.
- Using JavaScript, we can even create new events in the page.
- Every web page resides inside a browser window which can be considered as an object.
- A Document object represents the HTML document that is displayed in that window.
- The Document object has various properties that refer to other objects which allow access to and modification of document content.

What is DOM?

- The DOM is an acronym for the Document Object Model. It is a programming interface for Core, XML, and HTML DOM.
- It is a W3C (World Wide Web Consortium) standard.
- The DOM defines the logical or tree-like structure of the web page or document.
- In the tree, each branch ends in a node, and each node contains objects.
- DOM methods allow us programmatic access to the tree. Using the DOM methods, you can change the document's structure, content or style.

What is HTML DOM?

- HTML creates the web page's structure, and JavaScript adds interaction to the web page by manipulating the HTML elements.
- JavaScript can't interact directly with HTML elements. So, whenever a web page loads in the browser, it creates a DOM.
- So, the document object represents the HTML document displayed in that window.
- Furthermore, each iframe in the webpage creates a new DOM.
- The Document object has various properties that refer to other objects that allow access to and modify document content.
- The way document content is accessed and modified is called the Document Object Model, or DOM. The Objects are organized in a hierarchy.
- This hierarchical structure applies to the organization of objects in a Web document.

- **Document object** − Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page. It is used to access and modify the elements of the web page.
- **Form object** − everything enclosed in the <form>...</form> tags sets the form object.
- **Form control elements** − The form object contains all the elements defined for that object, such as text fields, buttons, radio buttons, and checkboxes.

Follow the syntax below to access the document object's properties and methods

Syntax: - document object access with the window object.

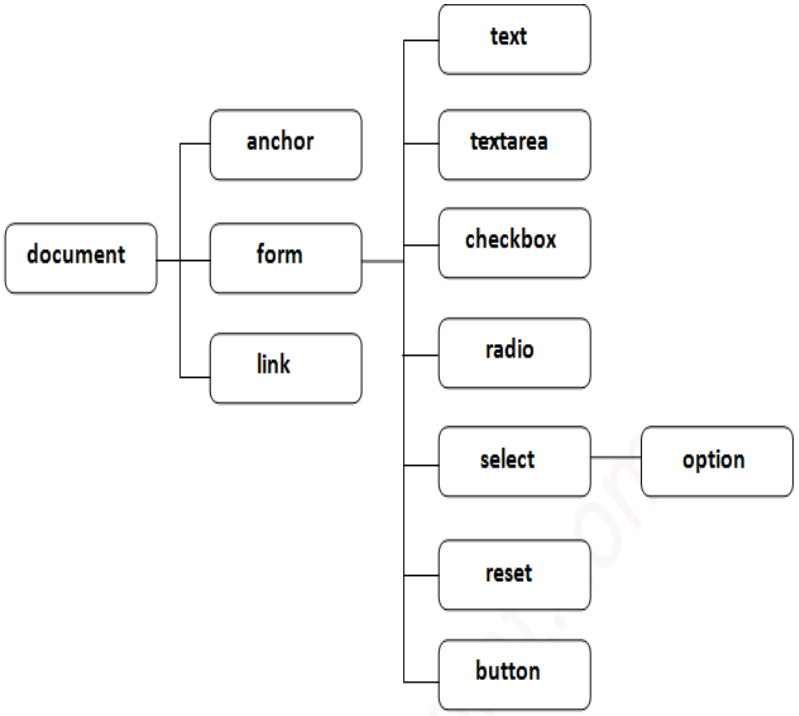
```
window.document.property name;
window.document.method name;
```

OR

Syntax: - document object access it without using the window object.

```
document. Property name;
document. Method name;
```

- Here is a simple hierarchy of a few important objects −



Document Object Methods

The JavaScript Document Object provides us with various methods that allow us to interact with and manipulate the HTML document.

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.

Example-1

In JavaScript, the simplest way to print "Hello World" is to use document.write() method.

```
<script>
  document.write("Hello World)
```

```
</script>
```

Example-2

In the example below, we define a div element with id, "output".
We access this element using the document.getElementById("output").
Then we change the innerHTML property and display our message, "Hello World".

```
<body>
  <div id = "output"> </div>
  <script>
    document.getElementById("output").innerHTML = "Hello World";
  </script>
</body>
```

Document Object Properties

Property	Description
documentElement	To get the <html> element.
forms	It returns an array of <form> elements of the document.
images	It returns the collection of the <img> elements.
links	To get the collection of all <a> elements.

DOM Collections

- The DOM (Document Object Model) collections are a way to group together related HTML elements. They are read-only and can be accessed using the properties of DOM objects such as the document object or a DOM node.
- There are many different types of DOM collections, including:
- **The HTMLCollection** object is an array-like list (collection) of HTML elements.
- The Form element collection in HTML DOM is used to set or return the collection of all <input> elements inside a form element.
- The HTML **DOM forms** collection is used for returning all the form elements that are present inside the HTML document as a collection.
- DOM collections can be used to perform a variety of tasks, such as:
  - Traversing the DOM
  - Adding, removing, or modifying elements
  - Changing the style or content of elements

- Responding to user events

Properties and Methods of HTMLCollection Object

- Here, we have listed the properties and methods that can be used with HTML collections.

Method / Property	Description
length	To get a count of HTML elements in the collection.

Collections of document Object and DOM Elements

- The document object contains some built-in collection to get the elements from the document.
- In the below table, we have listed all collections which can be accessed using the document object.

Collection Name	Description
document.links	To get all <a> elements from the document.
document.forms	To get all <form> elements from the document.
document.images	To get all <img> elements from the document.

EXAMPLE:-1

```
<body>
<a href="/html/default.asp">HTML</a>
<a href="/css/default.asp">CSS</a>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML ="Number of links: " +
document.links.length;
</script>
</body>
```

OUTPUT: Number of links: 2

EXAMPLE-2:-

```
<body>
<form action="">
```

```
First name: <input type="text" name="fname" value="Donald">
<input type="submit" value="Submit">
</form>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML ="Number of forms: " +
document.forms.length;
</script>
</body>
```

OUTPUT: Number of forms: 1

EXAMPLE-3

```
<body>
<img src="pic_htmltree.gif" width="486" height="266">
<img src="pic_navigate.gif" width="362" height="255">
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML ="Number of images: " +
document.images.length;
</script>
</body>
```

OUTPUT:-Number of images: 2

ACCESSING FORM ELEMENTS

1. Input Text Object

The Input Text object represents an HTML <input> element with type="text".

Input Text Object Methods

Method	Description
focus()	Gives focus to a text field

Input Text Object Properties

Property	Description
value	Sets or returns the value of the value attribute of the text field

- The value property sets or returns the value of the value attribute of a text field.
- The value property contains the default value OR the value a user types in (or a value set by a script).

Syntax:-

Return the value property: textObject.value

Set the value property: textObject.value = text

EXAMPLE:-

```
<body>
<input type="text" id="myText" value="Mickey">
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
  var x = document.getElementById("myText").value;
  document.getElementById("demo").innerHTML = x;
}
</script>
</body>
```

2. Input Button Object

The Input Button object represents an HTML <input> element with type="button".

Input Button Object Properties

- The value property sets or returns the value of the value attribute of an input button.
- The value attribute defines the text that is displayed on the button.

Property	Description
value	Sets or returns the value of the value attribute of an input button

Syntax:-

Return the value property:buttonObject.value

Set the value property: buttonObject.value = text

EXAMPLE:-

```
<body>
<input type="button" id="myBtn" value="Volvo">
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
  document.getElementById("myBtn").value = "BMW";
}
</script>
</body>
```

3. Input Radio Object

The Input Radio object represents an HTML <input> element with type="radio".

Input Radio Object Properties

Property	Description
checked	Sets or returns the checked state of a radio button
value	Sets or returns the value of the value attribute of the radio button

checked Property

The checked property sets or returns the checked state of a radio button.

Syntax:

Return the checked property:radioObject.checked
Set the checked property:radioObject.checked = true|false

value Property

- The value property sets or returns the value of the value attribute of the radio button.
- If a radio button is in checked when the form is submitted, the name of the radio button is sent along with the value of the value property if the radio button is not checked, no information is sent .

Value	Description
true|false	Specifies whether a radio should be checked or not. true – The radio is checked false - Default. The checkbox is not checked

Syntax:-

Return the value property: radioObject.value
Set the value property: radioObject.value = text

EXAMPLE:-

```
<body>
<input type="radio" id="myRadio" >
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
  var x = document.getElementById("myRadio").checked;
  document.getElementById("demo").innerHTML = x;
}
</script>
</body>
```

4. Input Checkbox Object

The Input Checkbox object represents an HTML <input> element with type="checkbox".

Input Checkbox Object Properties

Property	Description
checked	Sets or returns the checked state of a checkbox
value	Sets or returns the value of the value attribute of a checkbox

checked Property

Syntax:

Return the checked property:checkboxObject.checked
Set the checked property:checkboxObject.checked = true|false

Value	Description
true|false	Specifies whether a checkbox should be checked or not. true - The checkbox is checked false - Default. The checkbox is not checked

value Property

- If a checkbox is in checked when the form is submitted, the name of the checkbox is sent with the value of the value property if the checkbox is not checked, no information is sent.

Syntax:-

Return the value property:checkboxObject.value
Set the value property:checkboxObject.value = text

EXAMPLE:-

```
<body>
<input type="checkbox" id="myCheck">
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
  var x = document.getElementById("myCheck").checked;
  document.getElementById("demo").innerHTML = x;
}
</script>
</body>
```

5. Select Object

The Select object represents an HTML <select> element.

Select Object Collections

Collection	Description
options	Returns a collection of all the options in a drop-down list

The options collection returns a collection of all <option> elements in a drop-down list.

Syntax:-selectObject.options

```
<body>
<form>
  <select id="mySelect" size="4">
    <option>Apple</option>
    <option>Orange</option>
    <option>Pineapple</option>
    <option>Banana</option>
  </select>
</form>

<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction()
{
  var x = document.getElementById("mySelect").options.length;
  document.getElementById("demo").innerHTML = "Found " + x + " options in the list.";
}
</script>
</body>
```

Property	Description
selectedIndex	Sets or returns the index of the selected <option> element in the collection (starts at 0)

Syntax:

Return the selected Index property: selectObject.selectedIndex

Set the selected Index property: selectObject.selectedIndex = number

EXAMPLE:-

```
<body>
<select id="mySelect">
```

```
  <option>Apple</option>
  <option>Orange</option>
  <option>Pineapple</option>
  <option>Banana</option>
</select>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
  document.getElementById("mySelect").selectedIndex = "2";
}
</script>
```

INTRODUCTION TO EVENTS

- JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.
- Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.
- Events are a part of the Document Object Model (DOM).
- The change in the state of an object is known as an **Event**. When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.
- In html, there are various events which represents that some activity is performed by the user or by the browser. When javascript code is included in HTML, js react over these events and allow the execution. This process of reacting over the events is called Event Handling. Thus, js handles the HTML events via Event Handlers.
- For example, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

- **onclick Event:** The onclick event occurs when the user clicks on an element.
  Syntax
  <input type=element onclick="SomeJavaScriptCode">
  Example:
  <input type=button onclick="copyText()">Copy Text</button>
- **onblur Event:** The onblur event occurs when an object loses focus.
  Syntax
  <input type=element onblur="SomeJavaScriptCode">

Example

```
<input type="text" id="fname" onblur="upperCase()">
```

- **onfocus Event:** The on focus event occurs when an element gets focus.

  Syntax:

  ```
  <input type=element onfocus="SomeJavaScriptCode">
  ```

  Example

  ```
  <input type="text" id="fname" onfocus="setStyle(this.id)">
  ```

- **onload Event:** The onload event occurs when the page is loaded.

  Syntax:

  ```
  <input type=element onload="SomeJavaScriptCode">
  ```

  Example

  ```
  <input type=element onload="SomeJavaScriptCode">
  <body onload="load()">
  ```

- **onmousedown Event:** The onmousedown event occurs when a user presses a mouse button over an element.

  Syntax

  ```
  <input type=element onmousedown="SomeJavaScriptCode">
  ```

  Example

  ```
  <p onmousedown="mouseDown()">Click the text!</p>
  ```

- **onmouseup Event:** The onmouseup event occurs when a user releases a mouse button over an element.

  Syntax

  ```
  <input type=element onmouseup="SomeJavaScriptCode">
  ```

  Example

  ```
  <p onmouseup="mouseUp()">Click the text!</p>
  ```

- **onmouseout Event:** The onmouseout event occurs when a user moves the mouse pointer out of an element.

  Syntax

  ```
  <input type=element onmouseout="SomeJavaScriptCode">
  ```

  Example

  ```
  <img onmouseout="normalImg(this)" src="smiley.gif" alt="Smiley" />
  ```

- **onmouseover Event:** The onmouseover event occurs when a user mouse over an element.

  Syntax

<input type=elementonmouseover="SomeJavaScriptCode">

Example

- **onReset event:** The on Reset event handler is used to execute specified JavaScript code whenever the user resets a form by clicking a Reset button.

  Syntax

  <input type=element onreset="SomeJavaScriptCode">

  Example

  <FORM NAME="form1" onReset="alert('Reset to Book')">

- **onSubmit event:** The on Submit event handler is used to execute specified JavaScript code when the user submits a form by clicking a submit button.

  Syntax:

  <input type=element onSubmit ="SomeJavaScriptCode">

  Example

  <FORM onSubmit="submitEvent()">
  <FORM onSubmit="return validate(this)">

- **onunload Event:** The on unload event occurs before the browser closes the document.

  Syntax

  <input type= element onunload="SomeJavaScriptCode">

  Example

```
<body onunload="OnUnload()">
   ...
</body>
```

NOTE: This material was prepared using the following websites:
https://www.tutorialspoint.com/javascript/index.htm
https://way2tutorial.com/javascript/tutorial.php
https://www.w3schools.com/js/
https://www.javatpoint.com/
https://www.geeksforgeeks.org/javascript/
https://www.tutorialsteacher.com/javascript
https://www.scientecheasy.com/2022/03/objects-in-javascript.html/