1.Description:

**1. Accesses Document Properties:**

* document.title: Gets the **title of the web page**.
* document.URL: Retrieves the **current page URL**.
* document.lastModified: Returns the **last modification date/time** of the document.

**2. Modifies Page Content Using the DOM:**

* Changes the color of the <h1> text (id="heading") to **blue**.
* Displays the gathered document info (title, URL, last modified date) inside the #output <div> using innerHTML.

Program:

<html>

<head>

<title>Document Object Example</title>

</head>

<body>

<h1 id="heading">hello World!</h1>

<p>This is a demo of document object.</p>

<button onclick="showInfo()">Show document Info</button>

<div id="output"></div>

<script>

function showInfo() {

// Using document properties

let title = document.title;

let url = document.URL;

let lastModified = document.lastModified;

// Using document methods

document.getElementById("heading").style.color = "blue";

// change heading color

document.getElementById("output").innerHTML =

"Title: " + title + "<br>" +

"URL: " + url + "<br>" +

"Last Modified: " + lastModified;

}

</script>

</body>

</html>

2.Description:

**1. showWindowInfo()**

Triggered by the **"Show Window Info"** button. It:

* Retrieves:
  + window.innerWidth: The **inner width** of the browser window (viewport).
  + window.innerHeight: The **inner height** of the window.
  + window.location.href: The **current page URL**.
* Displays this info inside the #output <div>.

**2. openNewWindow()**:Triggered by the **"Open New Window"** button. It:

* Uses window.open() to open **Google (**[**https://www.google.com**](https://www.google.com)**)** in a **new popup window**.
* Sets dimensions: width=400 and height=300.
* Stores the new window reference in newWin (used later to close it).

**3. closeNewWindow():**Triggered by the **"Close New Window"** button. It:

* Checks if newWin exists (i.e., a new window was opened).
* Calls newWin.close() to close the opened window.

Program:

<!DOCTYPE html>

<html>

<head>

<title> Window Object Example</title>

</head>

<body>

<h1> Window Object Demo</h1>

<button onclick="showWindowInfo()">Show Windown Info</button>

<button onclick="OpenNewWindow()">Open New Window</button>

<button onclick="CloseNewWindow()">Close New Window</button>

<div id="output"></div>

<script>

let newWin;//to store reference of opened window

function showWindowInfo()

{

//using window properties

let width=window .innerwidth;

let height=window.innerHeight;

let locationHref=window.location.href;

//Display info

document.getElementById("output").innerHTML=

"window width:"+width+"px<br>"+

"window Height:"+height+"px<br>"+

"current URL:"+locationhref;

}

function openNewWindow() {

//using window.open()

newWin=window.open("https://www.google.com","Example","width=400,height=300");

}

function closeNewWindow()

{

//using window.close()

if(newWin) {

newWin.close();

}

}

</script>

</body>

</html>

3.Description: This HTML and JavaScript code demonstrates how to use the **JavaScript Array object** to manage and manipulate lists of items. It highlights both **array properties** and **commonly used methods** for modifying and accessing array data.

The script begins by initializing an array and retrieving its **length**, a property that indicates how many items are in the array. It then uses various **array methods** to add elements to the beginning and end of the array, remove elements from both ends, sort the array alphabetically, and convert the array into a string using a specified separator.

The results of these operations are then displayed dynamically on the web page using the innerHTML property of a DOM element. This provides a clear demonstration of how array data can be manipulated and presented in real-time using JavaScript.

Program:

<html>

<head>

<title>Array Object properties and Methods</title>

</head>

<body>

<h2>Array object Example</h2>

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

<script>

//creating an array

let fruits=["Apple","Banana","Mango"];

//Array property:length

let length=fruits.length;

//Array methods

fruits.push("Orange");

fruits.unshift("Grapes");

let last=fruits.pop();

let first=fruits.shift();

let sorted=fruits.sort();

let joined=fruits.join(",");

//Display results

document.getElementById("output").innerHTML=

"<b>Original Length:</b>"+length+"<br>"+

"<b>After push&unshift:</b>Apple,Banana,Mango,Orange,Grapes<br>"+

"<b>Removed Last:</b>"+last+"<br>"+

"<b>Removed First:</b>"+first+"<br>"+

"<b>Sorted Array:</b> " + sorted + "<br> +

"<b>joined as string:</b> " + joined;

</script>

</body>

</html>

4.Description: This web page demonstrates the usage of the **JavaScript Math object**, which provides a collection of properties and methods for performing mathematical operations.

The script showcases key **properties** of the Math object that return mathematical constants such as π (pi), Euler’s number, and the square root of 2.

It also utilizes several **methods** to perform common mathematical tasks, including calculating square roots, exponents, absolute values, rounding numbers (both up and down), generating random numbers, and determining the maximum and minimum values from a set.

All the results from these operations are dynamically displayed in the browser using the DOM. This example is useful for understanding how the Math object can be applied in various numerical computations within JavaScript.

Program:

<!DOCTYPE html>

<html>

<head>

<title>Math Object Example</title>

</head>

<body>

<h2>Math Object Example</h2>

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

<script>

//Math Object Properties

let pivalue = Math.PI; // Value of pi

let evalue = Math.E; // Value of Euler's Formula

let sqrt2value = Math.SQRT2; // Square root of 2

// Math Object Methods

let sqrtresult = Math.sqrt(16); // Square root

let powerresult = Math.pow(2,5); // 2^5

let absresult = Math.abs(-25); // Absolute Value

let roundresult = Math.round(4.7); // Round to nearest integer

let ceilresult = Math.ceil(4.2); // Round up

let floorresult = Math.floor(4.8); // Round down

let randomresult = Math.random(); // Random Number between 0 and 1

let maxresult = Math.max(3,9,2,15,7); // Maximum

let minresult = Math.min(3,9,2,15,7); // Minimum

// Display Results

document.getElementById("output").innerHTML=

"<b>Properties:</b><br>"+

"PI: " + pivalue + "<br>" +

"E: " + evalue + "<br>" +

"SQRT2: " + sqrt2value + "<br><br>" +

"<b>Methods:</b><br>"+

"Square Root of 16: " + sqrtresult + "<br>" +

"2^5 (Power): " + powerresult + "<br>" +

"Absolute of -25: " + absresult + "<br>" +

"Round(4.7): " + roundresult + "<br>" +

"Ceil(4.2): " + ceilresult + "<br>" +

"Floor(4.8): " + floorresult + "<br>" +

"Random Number (0-1): " + randomresult + "<br>" +

"Max(3,9,2,15,7): " + maxresult + "<br>" +

"Min(3,9,2,15,7): " + minresult;

</script>

</body>

</html>

5.Description: This web page demonstrates the use of the **JavaScript String object**, showcasing its key **properties** and **methods** for working with text data.

The script begins by creating a string and then uses the length property to determine the number of characters in it.

Various string **methods** are applied to perform operations such as:

* Converting the string to **uppercase** and **lowercase**,
* **Trimming** whitespace,
* **Slicing** a portion of the string,
* **Replacing** a specific word,
* Finding a **character at a specific position**,
* Determining the **index** of a word,
* **Splitting** the string into an array of words.

All the results of these operations are displayed dynamically on the page using the DOM. This example helps illustrate how the String object can be used for **text manipulation and analysis** in JavaScript.

Program:

<!DOCTYPE html>

<html>

<head>

<title>String Object Example</title>

</head>

<body>

<h2>JavaScript String Object Example</h2>

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

<script>

// Create a string

let text = "Hello JavaScript World! ";

// Using string properties

let length = text.length;

// Using string methods

let upper = text.toUpperCase();

let lower = text.toLowerCase();

let trimmed = text.trim();

let sliced = text.slice(6, 16);

let replaced = text.replace("World", "Programmers");

let charAtPos = text.charAt(7);

let index = text.indexOf("JavaScript"); // corrected spelling

let splitted = text.trim().split(" ");

// Display the Results

document.getElementById("output").innerHTML =

"<b>Original String:</b> " + text + "<br>" +

"<b>Length:</b> " + length + "<br>" +

"<b>Uppercase:</b> " + upper + "<br>" + // corrected variable name

"<b>Lowercase:</b> " + lower + "<br>" +

"<b>Trimmed:</b> " + trimmed + "<br>" +

"<b>Sliced(6,16):</b> " + sliced + "<br>" +

"<b>Replaced:</b> " + replaced + "<br>" +

"<b>Character at 7:</b> " + charAtPos + "<br>" +

"<b>Index of 'JavaScript':</b> " + index + "<br>" +

"<b>Splitted:</b> " + splitted.join(", ");

</script>

</body>

</html>

6.Description:

This JavaScript program uses a regular expression (regex) to detect an email address in a string.  
The RegExp object is defined using a pattern that matches email formats.  
The test() method checks if the text contains a valid email.  
The result (true/false) is displayed using document.write().  
This demonstrates regex properties and methods for pattern matching in strings.

Program:

<!DOCTYPE html>0

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Regex Object Properties and Methods Demo</title>

</head>

<body>

<h1>Regex Demo</h1>

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

<script>

// Define regex to find all capitalized words

const regex = /\b[A-Z][a-z]\*\b/g;

const text = "Hello there! My name is ChatGPT. I Love JavaScript.";

const output = document.getElementById("output");

// Prepare output lines

let result = [];

// source property

result.push(`Pattern source: ${regex.source}`);

// flags property

result.push(`Flags: ${regex.flags}`);

// test() method

const hasCapitalWords = regex.test(text);

result.push(`Contains capitalized words? ${hasCapitalWords}`);

// exec() method in a loop to get all matches

let match;

regex.lastIndex = 0; // reset lastIndex before using exec loop

while ((match = regex.exec(text)) !== null) {

result.push(`Found '${match[0]}' at index ${match.index}`);

}

// lastIndex property after matching

result.push(`Last index after matching: ${regex.lastIndex}`);

// Display results in the page

output.innerHTML = result.join("<br>");

</script>

</body>

</html>

7.Description: This web page demonstrates the use of the **JavaScript Date object**, which allows developers to work with dates and times.

The script creates a new Date instance representing the **current date and time**, and then uses several built-in **methods** to extract individual components such as the year, month, date, day of the week, hours, minutes, seconds, and the time in milliseconds since January 1, 1970 (Unix Epoch).

It also includes an array of weekday names to convert the numeric day (0–6) returned by getDay() into a readable format.

Finally, all of this information is displayed dynamically on the web page using the DOM. This example provides a practical introduction to handling and formatting date and time values in JavaScript.

Program:

<!DOCTYPE html>

<html>

<head>

<title>Date Object Example</title>

</head>

<body>

<h2>JavaScript Date Object Example</h2>

<p id="dateInfo"></p>

<script>

// Create a new Date Object

let currentDate = new Date();

// Use Date Object methods

let year = currentDate.getFullYear();

let month = currentDate.getMonth() + 1; // Months are zero-based

let date = currentDate.getDate(); // Use lowercase 'date' to avoid conflict

let day = currentDate.getDay();

let hours = currentDate.getHours();

let minutes = currentDate.getMinutes();

let seconds = currentDate.getSeconds();

let time = currentDate.getTime();

// Array for week days

let days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

// Display the values

document.getElementById("dateInfo").innerHTML =

"Current Date: " + date + "/" + month + "/" + year + "<br>" +

"Day of the Week: " + days[day] + "<br>" +

"Current Time: " + hours + ":" + minutes + ":" + seconds + "<br>" +

"Milliseconds since Jan 1, 1970: " + time;

</script>

</body>

</html>

8.Description: This HTML and JavaScript example demonstrates how to create a user-defined object using a constructor function. The Student object includes properties (name, age, marks), a method to display details, a getter to return grade, and a setter to update marks. An object student1 is created and updated using the setter method. The student’s information and grade are then displayed on the webpage. This example shows basic object-oriented programming in JavaScript.

Program:

<!DOCTYPE html>

<html>

<head>

<title>User-Defined Object Example</title>

</head>

<body>

<h2>User-Defined Object in JavaScript</h2>

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

<script>

// Constructor function to define a user-defined object

function Student(name,age,marks) {

// Properties

this.name=name;

this.age=age;

this.marks=marks;

// Method

this.displayDetails = function() {

return "Name: " + this.name + ", Age: " + this.age + ", Marks: " + this.marks;

};

// Accessor (Getter)

this.getGrade = function() {

if (this.marks >= 90 ) return "A";

else if (this.marks >= 75 ) return "B";

else if (this.marks >= 50 ) return "C";

else return "Fail"; };

// Accessor (Setter)

this.setMarks = function(newMarks) {

this.marks = newMarks;

};

}

// Create an object using the constructor

let student1 = new Student("Bunny",18,95);

// Use setter to update marks

student1.setMarks(99);

// Display details and grade

document.getElementById("output").innerHTML =

student1.displayDetails() + "<br>" +

"Grade: " + student1.getGrade();

</script>

</body>

</html>