JAVASCIRPT

**JavaScript** is a lightweight, cross-platform, and interpreted compiled programming language which is also known as the scripting language for webpages. It is well-known for the development of web pages,

JavaScript can be used for [**Client-side**](https://www.geeksforgeeks.org/server-side-client-side-programming/) developments as well as [**Server-side**](https://www.geeksforgeeks.org/server-side-client-side-programming/) developments.

 <script>

        console.log("Welcome to GeeksforGeeks");

    </script>

 **Dynamically typed languages:** This language can receive different data types over time.

 **Case Sensitive Format:** JavaScript is case sensitive so you have to be aware of that.

**Handling:** Handling events is the main feature of JS, it can easily respond on the website when the user tries to perform any operation.

**JavaScript in body or head:** Scripts can be placed inside the body or the head section of an HTML page or inside both head and body.

**JavaScript in head:** A JavaScript function is placed inside the head section of an HTML page and the function is invoked when a button is clicked.

**Examples:**

<!DOCTYPE html>

<html>

   <head>

      <script>

         function gfg() {

           document.getElementById("demo").innerHTML = "Geeks For Geeks";

         }

      </script>

   </head>

   <body>

      <h2>JavaScript in Head</h2>

      <p id="demo" style="color:green;">geeksforgeeks.</p>

      <button type="button" onclick="gfg()">click it</button>

**External JavaScript**

**//** <p id="demo">geeksforgeeks.</p>

// <button type="button" onclick="myFunction()">Try it</button>

     // <script src="myScript.js"></script>

   </body>

</html

<p>Welcome</p>

<button type="button" onclick="myFunction()">

      Click Me!

  </button>

 <p id="geek"></p>

    <p id="geek1"></p>

    <p id="geek2"></p>

    <script>

  // Statement 1

        var a, b, c;

a=2;b=3;z=a+b;

document.getElementById(

          "geek").innerHTML =

            "The value of c is " + c + ".";

        function myFunction() {

            document.getElementById(

              "geek1").innerHTML = "Hello";

            document.getElementById(

              "geek2").innerHTML =

              "How are you?";

        }

    </script>

</body>

SERVER SIDE

CLINT SIDE

## Key Differences Between Server-side Scripting and Client-side Scripting

1. Server-side scripting is used at the backend, where the source code is not viewable or hidden at the client side (browser). On the other hand, client-side scripting is used at the front end which users can see from the browser.
2. When a server-side script is processed it communicates to the server. As against, client-side scripting does not need any server interaction.
3. The client-side scripting language involves languages such as HTML, CSS and JavaScript. In contrast, programming languages such as PHP, ASP.net, Ruby, ColdFusion, Python, C#, Java, C++, etc.
4. Server-side scripting is useful in customizing the web pages and implement the dynamic changes in the websites. Conversely, the client-side script can effectively minimize the load to the server.
5. Server-side scripting is more secure than client-side scripting as the server side scripts are usually hidden from the client end, while a client-side script is visible to the users.

| **Client-side scripting** | **Server-side scripting** |
| --- | --- |
| Source code is visible to the user. | Source code is not visible to the user because its output  of server-sideside is an HTML page. |
| Its main function is to provide the requested output to the end user. | Its primary function is to manipulate and provide access to the respective database as per the request. |
| It usually depends on the browser and its version. | In this any server-side technology can be used and it does not  depend on the client. |
| It runs on the user’s computer. | It runs on the webserver. |
| There are many advantages linked with this like faster.  response times, a more interactive application. | The primary advantage is its ability to highly customize, response  requirements, access rights based on user. |
| It does not provide security for data. | It provides more security for data. |
| It is a technique used in web development in which scripts run on the client’s browser. | It is a technique that uses scripts on the webserver to produce a response that is customized for each client’s request. |
| HTML, CSS, and javascript are used. | PHP, Python, Java, Ruby are used. |
| No need of interaction with the server. | It is all about interacting with the servers. |

**JavaScript Variables:** A JavaScript variable is the simple name of storage location where data to be stored. There are two types of variables in JavaScript which are listed below:

**Variables in JavaScript:**

Variables in JavaScript are containers that hold reusable data. It is the basic unit of storage in a program.

* The value stored in a variable can be changed during program execution.
* A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
* In JavaScript, all the variables must be declared before they can be used.
* **Local variables:** Declare a variable inside of block or function.
* **Global variables:** Declare a variable outside function or with a window object.
* <script>
* // Declare a variable and initialize it
* // Gloabal variable declaration
* var Name="Apple";
* // Function definition
* function MyFunction() {
* // Local variable declaration
* var num = 45;
* // Display the value of Gloabal variable
* document.writeln(Name);
* // Display the value of local variable
* document.writeln("<br>" + num );
* }
* // Function call
* MyFunction();
* </script>
* **Output:**
* Apple
* 45
* let globalVar = "This is a global variable";
* function fun() {
* let localVar = "This is a local variable";
* console.log(globalVar);
* console.log(localVar);
* }
* fun();

| **var** | **let** |
| --- | --- |
| **1.** | The var is a keyword that is used to declare a variable | The let is also a keyword that is used to declare a variable. |
| **2.** | Syntax -:  **var name = value;** | Syntax -:  **let name = value;** |
| **3.** | The variables that are defined with **var** statement have function scope. | The variables that are defined with **let** statement have block scope. |
| **4.** | We can declare a variable again even if it has been defined previously in the same scope. | We cannot declare a variable more than once if we defined that previously in the same scope. |
| **5.** | Hoisting is allowed with **var**. | Hoisting is not allowed with **let**. |
| **6.** | Example -:  **var websitename = “geeksforgeeks”;** | Example -:  **let x = 69;** |
| **7.** | var is an ECMAScript1 feature. | let is a feature of ES6. |
| **8.** | Its supported browsers are: Chrome, Internet Explorer, Microsoft Edge, Firefox, safari, opera | Its supported browsers are -: Chrome49, Microsoft Edge12, firefox44 , safari11, opera36 |

Basically we can declare variables in three different ways by using **var**, **let** and **const** keyword. Each keyword is used in some specific conditions.  
**var:** This keyword is used to declare variable globally. If you used this keyword to declare variable then the variable can accessible globally and changeable also. It is good for a short length of codes, if the codes get huge then you will get confused. 

* **Syntax:**

var variableName = "Variable-Value;"

* **Code:**

## javascript

|  |
| --- |
| <script>      var geeks = "GeeksforGeeks";      console.log(geeks);  </script> |

* **Output:**

GeeksforGeeks

**let:** This keyword is used to declare variable locally. If you used this keyword to declare variable then the variable can accessible locally and it is changeable as well. It is good if the code gets huge. 

* **Syntax:**

let variableName = "Variable-Value;"

* **Code:**

## javascript

|  |
| --- |
| <script>  if (true) {      let geeks = "GeeksforGeeks";      console.log(geeks);      }        /\* This will be error and         show geeks is not defined \*/      console.log(geeks);  </script> |

* **Output:**

GeeksforGeeks

**const:** This keyword is used to declare variable locally. If you use this keyword to declare a variable then the variable will only be accessible within that block similar to the variable defined by using let and difference between let and const is that the variables declared using const values can’t be reassigned. So we should assign the value while declaring the variable.

* **Syntax:**

const variableName = "Variable-Value;"

* **Code:**

## javascript

|  |
| --- |
| <script>      const geeks = "GeeksforGeeks";      console.log(geeks);  </script> |

* **Output:**

GeeksforGeeks

When we execute the function fun(), the output shows that both global, and local variables, are accessible inside the function as we are able to console.log them. This shows that inside the function we have access to both global variables (declared outside the function) and local variables (declared inside the function). Let’s move the console.log statements outside the function and put them just after calling the function.

**Local Variable:**

 <p id="Geeks"></p>

        <p id="geeks"></p>

        <script>

            myfunction();

            function myfunction() {

                var petName = "Sizzer"; // local variabl

                document.getElementById("Geeks").innerHTML =

                    typeof petName + " " + petName;

            }

            document.getElementById("geeks").innerHTML =

                typeof petName;

        </script>

However, that a statement outside of the function can’t refer to the variable named petName without causing an error. That’s because it has local scope

**Global Variable:** Si

.

 <p id="geeks"></p>

        <p id="Geeks"></p>

        <script>

            var petName = "Rocky";//global variable

            myFunction();

            function myFunction() {

                document.getElementById("geeks").innerHTML =

                  typeof petName + "- " + "My pet name is " + petName;

            }

            document.getElementById("Geeks").innerHTML =

               typeof petName + "- " + "My pet name is " + petName;

        </script>

Local variables have **Function Scope**:

They can only be accessed from within the function.

<p><b>carName</b> is undefined outside myFunction():</p>

<p id="demo1"></p>

<p id="demo2"></p>

<script>

myFunction();

function myFunction() {

let carName = "Volvo";

function myFunction() {  
  var carName = "Volvo";   // Function Scope  
}

function myFunction() {  
  let carName = "Volvo";   // Function Scope  
}

function myFunction() {  
  const carName = "Volvo";   // Function Scope  
}

document.getElementById("demo1").innerHTML = typeof carName + " " + carName;

}

document.getElementById("demo2").innerHTML = typeof carName;

</script>

**let** is a keyword used to declare variables in JavaScript that are block scoped. There are three ways you can declare variables in JavaScript **let, const, var(old)**.

It describes that the const variable cannot be reassigned.

|  |  |  |
| --- | --- | --- |
| <script type="text/javascript">  const x = 22;      {          const x = 90;          console.log(x);            {              const x = 77;              console.log(x);          }          {              const x = 45;              console.log(x);          }      }      console.log(x);  </script>  **Example 5:** It describes that the array values can be modified only reference to array cannot be change.   |  | | --- | | <script type="text/javascript">        // Changing the content of array is      // possible in cost array      const arr1 = ["pankaj", "sumit", "chandan", "ajay"];        console.log(arr1.toString());        arr1[2] = "Narayan"; // possible        console.log(arr1.toString());  </script> |   **Output:**  pankaj, sumit, chandan, ajay  pankaj, sumit, Narayan, ajay  **Example 6:** It describes that the object properties can be modified only reference to object cannot be changed.   |  | | --- | | <script type="text/javascript">        const person = {          first\_name: "Pankaj",          last\_name: "Singh",          Age: 20,          About: "Web Developer and Competitive Programmer"      };        console.log(person);        // It is possible      person.first\_name = "Aryan";      person.last\_name = "Yadav";      person.Age = 22;      person.About = "Commerce undergraduate";        console.log(person);        // it is not possible      // const person={      //     "first\_name":"Aryan",      //     "last\_name":"Yadav",      //     "Age":22,      //     "About":"Commerce undergraduate"      // }  </script> |   **Output:** **Prototypes** |

**JavaScript Operator:** JavaScript operators are symbols that are used to compute the value or in other word we can perform operations on operands. Arithmetic operators ( +, -, \*, / ) are used to compute the value and Assignment operator ( =, +=, %= ) are used to assign the values to variables.

**Example:**

|  |
| --- |
| <script>    // Variable Declarations  var x, y, sum;    // Assign value to the variables  x = 3;  y = 23;    // Use arithmetic operator to  // add two numbers  sum = x + y;    document.write(sum);    </script> |

**Output:**

26

**JavaScript Expression:** Expression is the combination of values, operators, and variables. It is used to compute the values.

**Example:**

|  |
| --- |
| <script>    // Variable Declarations  var x, num, sum;    // Assign value to the variables  x = 20;  y = 30    // Expression to divide a number  num = x / 2;    // Expression to add two numbers  sum = x + y;    document.write(num + "<br>" + sum);    </script> |

**Output:**

10

50

**JavaScript Keyword:** The keywords are the reserved words that have special meaning in JavaScript.

// var is the keyword used to define the variable

var a, b;

// function is the keyword which tells the browser to create a function

function GFG(){};

**JavaScript Comments:** The comments are ignored by JavaScript compiler. It increases the readability of code. It adds suggestions, Information and warning of code. Anything written after double slashes // (single line comment) or between /\* and \*/ (multi-line comment) is treated as comment and ignored by JavaScript compiler.

**Example:**

|  |
| --- |
| <script>    // Variable Declarations  var x, num, sum;    // Assign value to the variables  x = 20;  y = 30    /\* Expression to add two numbers \*/  sum = x + y;    document.write(sum);    </script> |

50

**JavaScript Data Types:** JavaScript provides different datatype to hold different values on variable. JavaScript is a dynamic programming language, it means do not need to specify the type of variable. There are two types of data types in JavaScript.

* Primitive data type
* Non-primitive (reference) data type

// It store string data type

var txt = "GeeksforGeeks";

// It store integer data type

var a = 5;

var b = 5;

// It store Boolean data type

(a == b )

// To check Strictly (i.e. Whether the datatypes of both variables are same) === is used

(a===b)---> returns true to the console

// It store array data type

var places= ["GFG", "Computer", "Hello"];

// It store object data (objects are represented in the below way mainly)

var Student = {

firstName:"Johnny",

lastName:"Diaz",

age:35,

mark:"blueEYE"}

// Javascript(Dynamically typed)

var x = 5; // can store an integer

var name = 'string'; // can also store a string.

* **Numbers**: Represent both integer and floating-point numbers. Example: 5, 6.5, 7 etc.
* **String**: A string is a sequence of characters. In JavaScript, strings can be enclosed within the single or double quotes. Example: “Hello GeeksforGeeks” etc.
* **Boolean**: Represent a logical entity and can have two values: true or false.
* **Null**: This type has only one value : *null.*
* **Undefined**: A variable that has not been assigned a value is *undefined.*
* **Symbol:** Unlike other primitive data types, it does not have any literal form. It is a built-in object whose constructor returns a symbol-that is unique.
* **bigint:** The bigint type represents the whole numbers that are larger than 253-1. To form a bigint literal number, you append the letter n at the end of the number.
* **Object**: It is the most important data-type and forms the building blocks for modern JavaScript. We will learn about these data types in detail in further articles.

**JavaScript Functions:** JavaScript functions are the blocks of code used to perform some particular operations. JavaScript function is executed when something calls it. It calls many times so the function is reusable.

**Syntax:**

function functionName( par1, par2, ....., parn ) {

// Function code

}

JavaScript function can contain zero or more arguments.

**Example:**

|  |
| --- |
| <script>    // Function definition  function func() {        // Declare a variable      var num = 45;        // Display the result      document.writeln(num);  }    // Function call  func();    </script> |

**Output:**

45

# JavaScript | Output

The output can be display by using four different ways which are listed below:

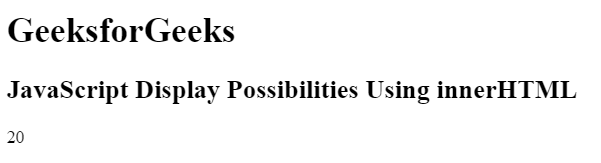
* **innerHTML:** It is used to access an element. It defines the HTML content.

**Syntax:**

document.getElementById(id)

**Example:** This example uses innerHTML to display the data.

|  |
| --- |
| <!DOCTYPE html>  <html>    <head>      <title>          JavaScript Output using innerHTML      </title>  </head>    <body>      <h1>GeeksforGeeks</h1>        <h2>          JavaScript Display Possibilities          Using innerHTML      </h2>        <p id="GFG"></p>        <!-- Script to uses innerHTML -->      <script>          document.getElementById("GFG").innerHTML                  = 10 \* 2;      </script>  </body>    </html> |

**Output:**  


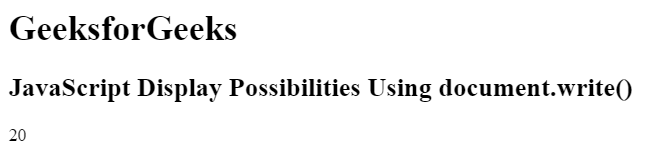
* **document.write():** It is used for testing purpose.

**Syntax:**

document.write()

**Example:** This example uses document.write() property to display data.

|  |
| --- |
| <!DOCTYPE html>  <html>    <head>      <title>          JavaScript Output using document.write()      </title>  </head>    <body>      <h1>GeeksforGeeks</h1>        <h2>          JavaScript Display Possibilities          Using document.write()      </h2>        <p id="GFG"></p>        <!-- Script to uses document.write() -->      <script>          document.write(10 \* 2);      </script>  </body>    </html> |

**Output:**  


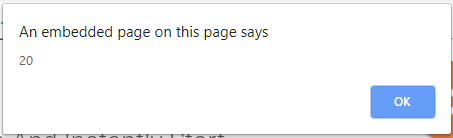
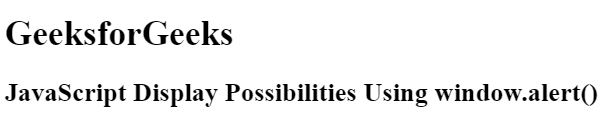
* **window.alert():**It displays the content using an alert box.

**Syntax:**

window.alert()

**Example:** This example uses window.alert() property to display data.

|  |
| --- |
| <!DOCTYPE html>  <html>    <head>      <title>          JavaScript Output using window.alert()      </title>  </head>    <body>      <h1>GeeksforGeeks</h1>        <h2>          JavaScript Display Possibilities          Using window.alert()      </h2>        <p id="GFG"></p>        <!-- Script to use window.alert() -->      <script>          window.alert(10 \* 2);      </script>  </body>    </html> |

**Output:**  
  


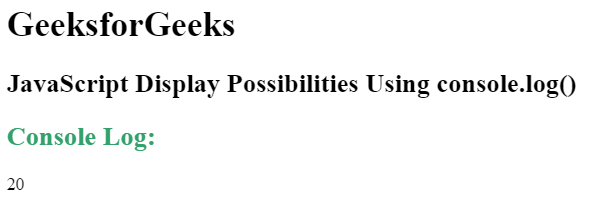
* **console.log():** It is used for debugging purposes.

**Syntax:**

console.log()

**Example:** This example uses console.log() property to display data.

|  |
| --- |
| <!DOCTYPE html>  <html>    <head>      <title>          JavaScript Output using innerHTML      </title>  </head>    <body>      <h1>GeeksforGeeks</h1>        <h2>          JavaScript Display Possibilities          Using console.log()      </h2>        <p id="GFG"></p>        <!-- Script to use console.log() -->      <script>          console.log(10\*2);      </script>  </body>    </html> |

**Output:**  


<!DOCTYPE html>

<html>

<head>

    <title>

        JavaScript Comments

    </title>

    <script>

        // Function to add two numbers

        function add() {

            // Declare three variables

            var x, y, z;

            // Input a number and store it into variable x

            x = Number( document.getElementById("num1").value );

            // Input a number and store it into variable x

            y = Number( document.getElementById("num2").value );

            // Sum of two numbers

            z= x +y;

            // Return the sum

            document.getElementById("sum").value = z;

        }

    </script>

</head>

<body>

    Enter the First number: <input id="num1"><br><br>

    Enter the Second number: <input id="num2"><br><br>

    <button onclick="add()">

        Sum

    </button>

    <input id="sum">

</body>

</html