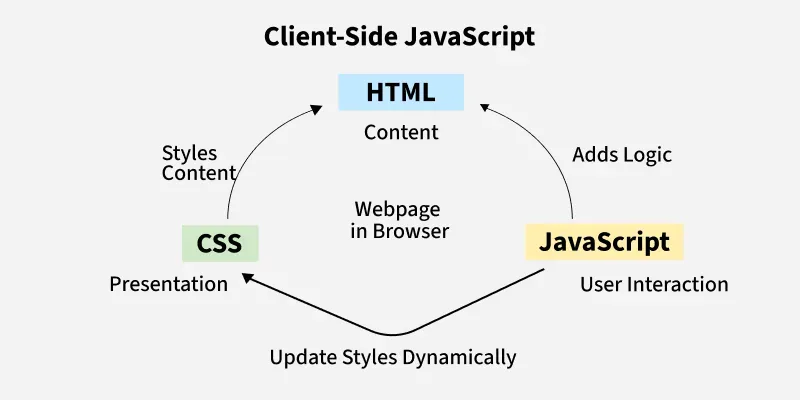
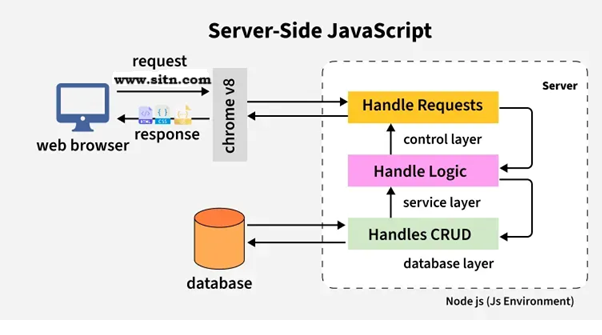
# JavaScript

JavaScript is a programming language used to create dynamic content for websites. It is a lightweight, cross-platform, and single-threaded programming language. It's an interpreted language that executes code line by line, providing more flexibility.

* **JavaScript on Client Side:** On the client side, JavaScript works along with HTML and CSS. HTML adds structure to a web page, CSS styles it, and JavaScript brings it to life by allowing users to interact with elements on the page, such as actions on clicking buttons, filling out forms, and showing animations. JavaScript on the client side is directly executed in the user's browser.
* **JavaScript on Server Side:** On the Server side (on Web Servers), JavaScript is used to access databases, file handling, and security features to send responses, to browsers.





## 

## **Conceptual Overview**

JavaScript is a lightweight, interpreted scripting language used to make web pages interactive. It runs in the browser, and can:

* Respond to user events like clicks and keyboard inputs.
* Modify HTML and CSS dynamically.
* Validate forms before submitting.
* Interact with the browser console for debugging.

## **Why Learn JavaScript?**

* JavaScript is core language for web development, enabling dynamic and interactive features in websites with fewer lines of code.
* It is highly in demand, offering many job opportunities in Frontend, Backend (Node.js), and Full Stack Development.
* JavaScript supports powerful frameworks and libraries like React, Angular, Vue.js, Node.js, and Express.js, widely used in modern web applications.
* JavaScript is an object-oriented and event-driven language, ideal for building scalable and responsive applications.
* It is cross-platform and runs directly in all modern web browsers without the need for installation.
* Major companies like Google, Facebook, and Amazon use JavaScript in their tech stacks.

## **Enabling JavaScript**

## All modern browsers come with built-in support for JavaScript, and it has enabled JavaScript by default. Frequently, you may need to enable or disable this support manually. This chapter explains how to turn JavaScript support on and off in your browsers: Chrome, Microsoft Edge, Firefox, Safari, and Opera.

## **JavaScript in Chrome**

## Here are the steps to turn on or turn off JavaScript in Chrome −

## Click the Chrome menu at the top right-hand corner of your browser.

## Select the Settings option.

## Click on the Privacy and Security tab from the left sidebar.

## Click Show advanced settings at the end of the page.

## Next, click on the Site Settings tab.

## Now, scroll to the bottom of the page, and find the content section. Click on the JavaScript tab in the content section.

## Here, you can select a radio button to turn JavaScript on or off.

## Also, you can add the URLS of the custom website to block and unblock JavaScript on particular websites.

## **JavaScript in Microsoft Edge**

## Here are simple steps to turn on or turn off JavaScript in your Microsoft Edge −

## Click Edge menu (three dots) at top right-hand corner of the edge browser.

## Follow More Tools Internet Options from the menu.

## Select Security tab from the dialog box.

## Click the Custom Level button.

## Scroll down till you find Scripting option.

## Select Enable radio button under Active scripting.

## Finally click OK and come out.

## To disable JavaScript support in your Microsoft Edge, you need to select Disable radio button under Active scripting.

## **JavaScript in Firefox**

## Here are the steps to turn on or turn off JavaScript in Firefox −

## Open a new tab type about: config in the address bar.

## Then you will find the warning dialog. Select Ill be careful, I promise!

## Then you will find the list of configure options in the browser.

## In the search bar, type javascript.enabled.

## There you will find the option to enable or disable javascript by right-clicking on the value of that option select toggle.

## If javascript.enabled is true, it converts to false upon clicking toggle. If javascript is disabled, it gets enabled upon clicking toggle.

## **JavaScript in Safari**

## When you install the Safari web browser, JavaScript comes installed by default. If you have disabled it and want to enable it, follow the steps below.

## Click on the safari menu from the top-left corner.

## Select the preferences in the dropdown menu. It will open a new window.

## Open the security tab.

## Check the Enable JavaScript checkbox in the web content section to enable the javascript. You can disable the JavaScript by unchecking the checkbox.

## Now, close the preference window and reload the web page.

## **JavaScript in Opera**

## Here are the steps to turn on or turn off JavaScript in Opera −

## Follow Tools Preferences from the menu.

## Select the Advanced option from the dialog box.

## Select Content from the listed items.

## Select Enable JavaScript checkbox.

## Finally, click OK and come out.

## To disable JavaScript support in your Opera, you should not select the enable JavaScript checkbox.

## **JavaScript in Brave**

## The Brave is well-known for its security and privacy. So, It doesnt allow us to disable the JavaScript permanently, but we can disable the JavaScript for the particular website by following the below steps.

## Open the website URL to disable the browser for it.

## Now, Click on the Brave Shields icon in the address bar.

## Find the Scripts option in the Shields panel.

## The default value of the Scripts is Allow Scripts. If you want to disable JavaScript, choose the "Block Scripts" option.

## **Warning for Non-JavaScript Browsers**

## If you have to do something important using JavaScript, then you can display a warning message to the user using <noscript> tags.

## You can add a noscript block immediately after the script block as follows −

## <html>

## <head>

## <script>

## document.write("Hello World!")

## </script>

## <noscript>

## Sorry...JavaScript is needed to go ahead.

## </noscript>

## </head>

## <body>

## </body>

## </html>

## 

## Now, if the user's browser does not support JavaScript or JavaScript is not enabled, then the message from </noscript> will be displayed on the screen.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## **JavaScript Syntax**

## JavaScript syntax comprises a set of rules that define how to construct a JavaScript code. JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

## You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.

## The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

## <script ...>

## JavaScript code

## </script>

## 

## The script tag takes two important attributes

## **Language −**This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.

## **Type −**This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript". JavaScript has become default lannguage in HTML5, and modern browsers, so now adding type is not required.

## So your JavaScript segment will look like −

## <script language = "javascript" type = "text/javascript">

## JavaScript code

## </script>

## 

## **Your First JavaScript Code**

## Let us take a sample example to print out "Hello World". We call **document.write** method which writes a string into our HTML document. This method can be used to write text, HTML, or both. Take a look at the following code −

## <html>

## <head>

## <title> Your first JavaScript program </title>

## <head>

## <body>

## <script language = "javascript" type = "text/javascript">

## document.write("Hello World!")

## </script>

## </body>

## </html>

## 

## This code will produce the following result −

## Hello World!

## 

## **JavaScript Values**

## In JavaScript, you can have two types of values.

## Fixed values (Literals)

## Variables (Dynamic values)

### **JavaScript Literals**

## In the below code, 10 is a Number literal and Hello is a string literal.

## <html>

## <body>

## <script>

## document.write(10); // Number Literal

## document.write("<br />"); // To add line-break

## document.write("Hello"); // String Literal

## </script>

## </body>

## </html>

## 

## This code will produce the following result −

## 10

## Hello

## 

### **JavaScript Variables**

## In JavaScript, variables are used to store the dynamic data.

## You can use the below keyword to define variables in JavaScript.

## var

## let

## const

## You can use the assignment operator (equal sign) to assign values to the variable.

## In the below code, variable a contains the numeric value, and variable b contains the text (string).

## <html>

## <body>

## <script>

## let a = 5; // Variable Declaration

## document.write(a); // Using variable

## document.write("<br>");

## let b = "One";

## document.write(b);

## </script>

## </body>

## </html>

## 

## This code will produce the following result −

## 5

## One

## 

## **Whitespace and Line Breaks**

## JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

## **Semicolons are Optional**

## Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

## <script>

## var1 = 10

## var2 = 20

## </script>

## 

## But when formatted in a single line as follows, you must use semicolons −

## <script>

## var1 = 10; var2 = 20;

## </script>

## 

## *It is a good programming practice to use semicolons.*

## **Case Sensitivity**

## JavaScript is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.

## So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

## In the code below, we compare the time and Time strings, which returns false.

## Open Compiler

## <html>

## <body>

## <script>

## let a = "time";

## let b = "Time";

## document.write("a == b? " + (a == b));

## </script>

## </body>

## </html>

## 

## This code will produce the following result −

## a == b? false

## 

## *Care should be taken while writing variable and function names in JavaScript.*

## 

## 

## JavaScript and Camel Case

## **Pascal Case** − We can create variables like SmartWatch, MobileDevice, WebDrive, etc. It represents the upper camel case string.

## **Lower Camel Case** − JavaScript allows developers to use variable names and expression names like smartwatch, mobileDevice, webDriver, etc. Here the first character is in lowercase.

## **Underscore** − We can use underscore while declaring variables like smart\_watch, mobile\_device, web\_driver, etc.

## *JavaScript doesnt allow adding the hyphen in variable name or expression name.*

## JavaScript Keywords

## JavaScript contains multiple keywords which we can use for a particular task. For example, the function keyword can be used to define the function. The let, var, and const keywords can be used to define variables.

## Lets understand the use of the keyword via the example below.

### Example

## In this example, we used the function keyword to define the function. We used the var keyword inside the function to define the sum variable.

## Also, we used the let and const keywords outside the function to define two variables and initialize them with values. After that, we called the function using the function name and passed variables as an argument.

## 

## 

## 

## 

## 

## <html>

## <body>

## <script>

## function getSum(first, second) {

## var sum = first \* second;

## document.write("The product of " + first + " and " + second + " is " + sum);

## }

## let first = 3;

## const second = 4;

## getSum(first, second);

## </script>

## </body>

## </html>

## 

## This code will produce the following result −

## The product of 3 and 4 is 12

## 

## *JavaScript doesnt allow to use the of keywords as a variable or expression name.*

## **JavaScript Identifiers**

## In JavaScript, identifiers are the name of variables, functions, objects, etc.

## For example, p is an identifier in the below code.

## <script>

## pet p = 90;

## </script>

## 

## The 'test' is an identifier in the below code.

## <script>

## function test() {

## }

## </script>

## 

## Here are the rules which you should follow to define valid identifiers.

## Identifiers should always start with the alphabetical characters (A-Z, a-z), $(dollar sign), or \_ (underscore).

## It shouldnt start with digits or hyphens.

## The identifier can also contain digits except for the start of it.

## **Comments in JavaScript**

## JavaScript supports both C-style and C++-style comments, Thus −

## Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.

## Any text between the characters /\* and \*/ is treated as a comment. This may span multiple lines.

## JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.

## The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

### 

### 

### 

### Example

## The following example shows how to use comments in JavaScript.

## <script>

## // This is a comment. It is similar to comments in C++

## /\*

## \* This is a multi-line comment in JavaScript

## \* It is very similar to comments in C Programming

## \*/

## </script>

## 

## 

## 

## 

## **Operators in JavaScript**

## JavaScript contains various arithmetic, logical, bitwise, etc. operators. We can use any operator in JavaScript, as shown in the example below.

### Example

## In this example, we have defined var1 and va2 and initialized them with number values. After that, we use the \* operator to get the multiplication result of var1 and var2.

## 

## <html>

## <body>

## <script>

## var1 = 10

## var2 = 20

## var3 = var1 \* var2;

## var4 = 10 + 20;

## document.write(var3, " " ,var4);

## </script>

## </body>

## </html>

## 

## This code will produce the following result −

## 200 30

## 

## In this way, programmers can use other operators with operands.

## *When any of two operands of the + operator is a string, it converts the other operand to a string and merges both strings.*

## **Expressions in JavaScript**

## You can create expressions in JavaScript by combining the variable, values, and operators.

## For example, the below expression adds two numbers.

## 10 + 20;

## 

## The below expression multiplies the values of two variables.

## a \* b;

## 

## The below expression divides the value of variable c with 2.

## c / 2;

## 

### Example

## In the below code, we have used the assignment and arithmetic expressions.

## <html>

## <body>

## <script>

## let a = 10;

## let b = 2;

## 

## let c = a; // Assigning a value of a to c. Assignment expression.

## let d = a + b; // Adding a and b. Arithmetic expression.

## let e = a - b; // Subtracting b from a.

## 

## document.write("c = " + c + "<br>");

## document.write("d = " + d + "<br>");

## document.write("e = " + e + "<br>");

## </script>

## </body>

## </html>

## 

## This code will produce the following result −

## c = 10

## d = 12

## e = 8

## 

## 

## 

## 

## 

## 

## 

## **Experiment Steps**

### 1. Inline JavaScript

Objective: Execute a function directly from an HTML element.

<!DOCTYPE html>

<html>

<head><title>Inline JS</title></head>

<body>

<button onclick="alert('Hello from Inline JavaScript!')">Click Me</button>

</body>

</html>

### 2. Internal JavaScript

Objective: Write JavaScript inside a <script> tag within the HTML document.

<!DOCTYPE html>

<html>

<head><title>Internal JS</title></head>

<body>

<h2 id="msg">Welcome</h2>

<button onclick="changeText()">Click</button>

<script>

function changeText() {

document.getElementById("msg").innerHTML = "Hello from Internal JavaScript!";

}

</script>

</body>

</html>

### 3. External JavaScript

Objective: Separate logic and HTML by linking an external .js file.

HTML (index.html):

<!DOCTYPE html>

<html>

<head><title>External JS</title></head>

<body>

<button onclick="showMessage()">External JS Test</button>

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

</body>

</html>

JavaScript (script.js):

function showMessage() {

alert("This alert is from an external JS file!");

}

## Task-Based Activities

### Task 1: Display Student Info

* Prompt the user for their name using prompt().
* Show the name using alert() and log it using console.log().

<script>

let name = prompt("Enter your name:");

alert("Welcome, " + name);

console.log("Student Name: " + name);

</script>

### Task 2: Console and DOM Practice

* Add a button.
* On click, log a message to the console and change text in a <p> tag.

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

<button onclick="updateOutput()">Click Me</button>

<script>

function updateOutput() {

console.log("Button clicked!");

document.getElementById("output").innerHTML = "Updated Message!";

}

</script>

### 

### Task 3: External Calculator Script (Addition)

* Create an HTML file that takes two numbers as input.
* Link an external JS file that adds them and displays the result in the browser.

HTML:

<input type="number" id="a"> +

<input type="number" id="b">

<button onclick="add()">Add</button>

<p id="result"></p>

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

calc.js:

function add() {

let a = parseFloat(document.getElementById("a").value);

let b = parseFloat(document.getElementById("b").value);

document.getElementById("result").innerHTML = "Sum: " + (a + b);

}

Final Problem Statement of Practical 1:

Design and develop an interactive college department webpage for the SIT Nagpur Computer Science Department that displays a welcome message using JavaScript and links to a student information page. On the student page, prompt the user for their name, display it using an alert, log it in the browser console, and dynamically update the content on the page using DOM manipulation. The solution must demonstrate the use of both internal and external JavaScript in a modular structure.