



# What is JavaScript?

- JavaScript (JS) is a high-level, interpreted programming language used to make web pages interactive.(JIT)
- Runs directly in web browsers, allowing dynamic and responsive behavior.
- Supports both client-side (browser) and server-side (Node.js) development.

# **Common Uses of JavaScript:**

- Form validation (checking user input before submission).
- Animations and transitions (smooth effects for UI elements).
- Interactive elements (buttons, modals, dropdown menus, carousels).
- DOM manipulation (dynamically updating content without reloading the page).
- Event handling (responding to user actions like clicks and keypresses).

# **History of JavaScript**

- Developed by Brendan Eich in 1995 in 10 days
- Originally called Mocha, then LiveScript, finally JavaScript
- Standardized as ECMAScript (ES)

# Why Learn JavaScript?

- Core for frontend (HTML + CSS + JS = Website)
- Used in modern frameworks (React, Angular, Vue)
- Also used in full stack (MERN, MEAN)



# Importance of JavaScript in Web Development

JavaScript is a core technology of modern web development along with HTML and CSS.

- Enhances User Experience Enables dynamic and interactive elements.
- Client-Side Processing Reduces server load by executing code in the browser.
- Asynchronous Operations Allows smooth loading of data without refreshing the page (AJAX, Fetch API).
- Cross-Platform Compatibility Works on all major browsers and devices.
- **Powerful Ecosystem** Numerous libraries and frameworks for fast development.

# **Famous JavaScript Frameworks and Libraries**

JavaScript has a wide range of libraries and frameworks for front-end and back-end development.

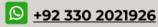
### Front-End Frameworks/Libraries:

- React.js Developed by Facebook, used for building UI components.
- Vue.js Lightweight and easy-to-learn framework for interactive UI.
- Angular.js Developed by Google, used for large-scale applications.
- **jQuery** Simplifies DOM manipulation and event handling.

### **Back-End Frameworks:**

- Node.js Enables JavaScript to run on the server.
- Express.js A lightweight framework for building web applications with Node.js.





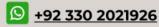
# JavaScript and the MERN Stack

MERN is a popular JavaScript-based full-stack development framework.

- M MongoDB (NoSQL database)
- E Express.js (Back-end framework)
- R React.js (Front-end framework)
- N Node.js (Server-side runtime)

# Why MERN?

- Uses JavaScript across the stack (front-end & back-end).
- Faster development with reusable components.
- Scalable and efficient for modern web applications.



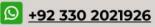
# **Setting Up JavaScript**

### **Using the Browser Console**

- Open Developer Tools (F12 or Ctrl + Shift + I in Chrome).
- Navigate to the Console tab.
- Type and execute JavaScript code directly.

# **Example:**

console.log("Hello, World!");



# **Adding JavaScript to HTML**

JavaScript can be added inside an HTML file in two ways:

# 1. Internal JavaScript

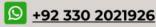
Writing JavaScript code directly inside the HTML file within a <script> tag.

# **Example:**

```
<script>
    console.log("Hello, JavaScript!");
</script>
```







### **External JavaScript File**

Writing JavaScript in a separate file and linking it using the <script> tag.

### HTML File (index.html):

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

# JavaScript File (script.js):

console.log("Hello, JavaScript!");



# **Writing Your First JavaScript Code**

### 1. Console Output

 The console.log() method is used for debugging and printing messages to the console.

### **Example:**

console.log("Hello, JavaScript!");

#### 2. Alert Box

The alert() function displays a popup message to the user.

### **Example:**

alert("Welcome to JavaScript!");







# **JavaScript Syntax and Basics**

#### **Variables**

A variable is a named **container** for **storing data values**. You can think of it like a labeled box where you can keep a value and use it later in your code.

#### 1. var

var is the **oldest** way to declare a variable in JavaScript (introduced in ES5 and earlier). It is function-scoped, which means the variable is available within the function in which it is defined.

#### 2. let

let was introduced in ES6 (2015). It is block-scoped, which means the variable is only available within the nearest set of curly braces {}.

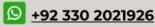
#### 3. const

const (also from ES6) stands for constant. It is also block-scoped, but the variable cannot be reassigned after it's declared.

# **Comparison Table**

Feature	var	let	const
Scope	Function	Block	Block
Hoisting	Yes (undefined)	Yes (TDZ*)	Yes (TDZ*)
Re-declaration	Yes	No	No
Reassignment	Yes	Yes	No





# أُسْتَغْفِرُ ٱللهَ

## **JavaScript Data Types**

JavaScript has two main categories of data types:

- Primitive Data Types
- Non-Primitive (Reference) Data Types

### 1. Primitive Data Types

These are the most basic types of data. They store single values, are immutable, and are stored by value.

Symbol

# **List of Primitive Types:**

Туре	Description	Example
String	Text data	"Hello" or 'Hi'
Number	Integer or floating point number	25, 3.14
Boolean	Logical value: true or false	true, false
Undefined	A variable declared but not assigned a value	let x;
Null	Represents no value (intentional empty)	let x = null;

Unique, immutable value often used as object keys

Symbol("id")

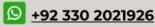
# **Characteristics of Primitive Types:**

- Stored directly in the variable
- Compared by value

### **Examples:**

```
let name = "Munim";  // String
let age = 23;  // Number
let isStudent = true;  // Boolean
let email;  // Undefined
let address = null;  // Null
let uniqueId = Symbol("id");  // Symbol
```







# 2. Non-Primitive (Reference) Data Types

These store collections of values and are mutable. They are stored and compared by reference, not value.

### **Common Non-Primitive Types:**

Туре	Description	Example
Object	Collection of key-value pairs	{name: "Munim", age: 25}
Array	Ordered collection of values	[1, 2, 3]
Function	Reusable block of code	function() {}

# **Characteristics of Non-Primitive Types:**

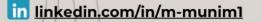
- Stored as reference in memory
- Mutable (can be changed after creation)
- Can contain multiple values
- Can have methods and properties

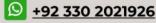
# **Examples:**

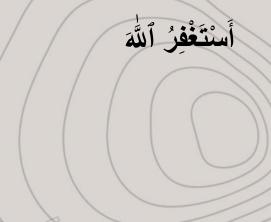
```
let user = {
  name: "Munim",
  age: 23
};

let numbers = [1, 2, 3, 4];

function greet() {
  console.log("Hello!");
}
```







# أُسْتَغْفِرُ ٱللهَ

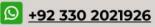
# **Key Differences: Primitive vs Non-Primitive**

Feature	Primitive	Non-Primitive
Stored as	Value	Reference (memory address
Mutability	Immutable	Mutable
Comparison	By value	By reference
Examples	String, Number	Object, Array, Function

# **Bonus: typeof Operator**

You can check data types in JavaScript using the typeof operator:

```
console.log(typeof "Hello"); // string
console.log(typeof 123); // number
console.log(typeof true); // boolean
console.log(typeof undefined); // undefined
console.log(typeof null); // object (quirk in JS)
console.log(typeof {}); // object
console.log(typeof []); // object
console.log(typeof function(){}); // function
```





# **Console and Output in JavaScript**

JavaScript provides several ways to display output to the user or developer. The most commonly used methods are:

# 1. console.log()

Displays a message in the browser's developer console (accessible via F12 or right-click  $\rightarrow$  Inspect  $\rightarrow$  Console tab).

Syntax:

console.log("Your message here");



### 2. alert()

Displays a popup alert box with a message and an OK button. It pauses script execution until the user closes the popup.

# **Syntax:**

alert("This is an alert!");

# 3. document.write() -> use, writeln

Writes a string of text directly into the HTML page, usually during page load.

### Syntax:

document.write("Your content here");