What is JavaScript?

JavaScript (often called **JS**) is a **high-level, dynamic programming language** used mainly for making web pages interactive. It runs directly in web browsers and allows developers to add features like animations, form validations, dynamic content updates, and interactive buttons. It is one of the core technologies of web development, along with **HTML** and **CSS**.

Why We Use JavaScript

We use JavaScript because it:

- Makes web pages **interactive and engaging**.
- Allows **real-time updates** without reloading the page (using AJAX or Fetch API).
- Enables **client-side validation**, reducing server load.
- Works on both the **frontend** (browser) and **backend** (with Node.js).
- Supports building **web apps, mobile apps, and even games**.
- Has a huge community and many **libraries & frameworks** like React, Angular, and Vue.

When We Use JavaScript

We use JavaScript when:

- We need to add **interactivity** (like button clicks, dropdowns, sliders).
- We want to **manipulate HTML or CSS dynamically** (e.g., hide/show elements).
- We build **web applications** that require user interaction.
- We need **form validation** before submitting data.
- We want **asynchronous data fetching** (without page reload).
- We develop **server-side logic** using **Node.js**.

Where We Use JavaScript

JavaScript is used in:

- **Web browsers** – to make websites interactive and responsive.
- **Web servers** – using Node.js for backend development.
- **Mobile apps** – with frameworks like React Native or Ionic.
- **Desktop applications** – using tools like Electron.js.
- **Game development** – using libraries like Phaser.js.
- **Internet of Things (IoT)** devices and **cloud computing** platforms.

Types of Using JavaScript

JavaScript can be used in different ways depending on where and how it's written in a web project.
There are mainly **three types of ways to use JavaScript**:

---

1. Inline JavaScript

- The JavaScript code is written **directly inside an HTML element** using the `onclick`, `onchange`, or other event attributes.
- It's used for **small scripts or quick actions**.

Example:

```
<button onclick="alert('Hello, World!')">Click Me</button>
```

-

---

2. Internal JavaScript

- The JavaScript code is written **inside the HTML file** within the `<script>` tag.

- Usually placed inside the `<head>` or at the bottom of the `<body>` tag.
- Used when the script is **specific to a single web page**.

Example:

```
<html>
<head>
  <script>
    function showMessage() {
      alert("Welcome to my website!");
    }
  </script>
</head>
<body>
  <button onclick="showMessage()">Click</button>
</body>
</html>
```

-

---

3. External JavaScript

- The JavaScript code is written in a **separate file** with the extension **`.js`**.
- That file is linked to the HTML using the `<script src="filename.js"></script>` tag.
- Best practice for **large projects** because it keeps code **organized, reusable, and easy to maintain**.

If We Use JavaScript, What Happens to Our Website?

When we use **JavaScript**, our website becomes **dynamic, interactive, and user-friendly** — instead of being just static text and images.

Here's what happens:

1. **Website Becomes Interactive**
    - Buttons, forms, sliders, menus, and animations start responding to user actions.
    - Example: Clicking a button shows a popup or hides an element.
2. **Faster and Smoother Experience**
    - JavaScript can update part of a webpage **without reloading the whole page** (using AJAX or Fetch API).
    - Example: Chat apps and social media feeds update instantly.
3. **Better User Interface (UI)**
    - You can add transitions, animations, and visual effects easily.
4. **Real-Time Data Handling**
    - JS helps to show live data such as weather updates, notifications, or stock prices.
5. **Client-Side Validation**
    - Forms can check for errors (like empty fields or invalid emails) before sending data to the server.
6. **Improved Functionality**
    - Enables features like dark mode toggle, drag & drop, and interactive charts.

Other Front-End Programming Languages (Besides JavaScript)

Technically, **JavaScript is the only native programming language** that browsers understand. However, there are **alternatives or languages that compile (convert) into JavaScript** for front-end use.

Here are some of them:

1. TypeScript

- A **superset of JavaScript** created by Microsoft.
- Adds **type safety** and better **error checking**.
- Used in large projects (Angular, React, etc.).
- Compiles into JavaScript.

2. Dart

- Created by Google.
- Used mainly with the **Flutter** framework for building **web, mobile, and desktop apps**.
- Compiles to JavaScript for browsers.

3. CoffeeScript

- A simplified syntax that compiles into JavaScript.
- Makes code shorter and cleaner.

4. Elm

- A **functional language** for building front-end web apps.
- It compiles into JavaScript and helps avoid runtime errors.

5. PureScript

- A **strongly typed functional language** similar to Haskell.
- Compiles to JavaScript for front-end use.