JavaScript Introduction

Introduction to JavaScript

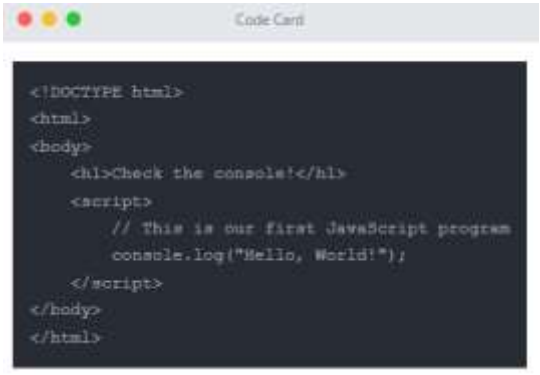
JavaScript is a **versatile, high-level programming language** that powers the interactive behavior of websites. It works alongside HTML (structure) and CSS (style) to create dynamic, responsive web pages.

- **Client-side**: Runs in the browser to handle user interactions.
- **Server-side**: Runs on servers using Node.js.
- **Dynamically typed**: Variable types are determined at runtime.
- **Interpreted**: Executes code line by line.
- **Single-threaded**: Handles one task at a time, but supports asynchronous operations.

---

2. Writing Your First JavaScript Program

A. In the Browser



- <script>: Embeds JavaScript in HTML.
- console.log(): Prints messages to the browser's developer console.

B. In Node.js (Server Console)

Create a file named hello.js:

// This is a comment

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

JavaScript Introduction

Run it in the terminal:

node hello.js

Output:

Hello, World!

---

3. Comments in JavaScript

Comments are notes in your code that are ignored by the interpreter.

- **Single-line**: // This is a comment
- **Multi-line**:

  /* This is a

  multi-line comment */

Use comments to explain logic, organize code, or temporarily disable lines.

---

4. Key Features of JavaScript

- **Client-side scripting**: Fast response without server communication.
- **Event-driven**: Reacts to user actions like clicks and typing.
- **Asynchronous**: Handles tasks like data fetching without freezing the page.
- **Versatile**: Used for simple scripts and complex applications.
- **Rich ecosystem**: Includes frameworks like React, Angular, and Vue.js.

---

5. Client-Side vs Server-Side JavaScript

Feature	Client-Side	Server-Side
Runs in	Browser	Server (Node.js)
Tasks	DOM manipulation, form validation	Database access, file handling, APIs

JavaScript Introduction

Feature	Client-Side	Server-Side
Examples	React, Vue, Angular	Node.js, Express.js

---

6. Programming Paradigms

JavaScript supports multiple styles:

- **Imperative**: Step-by-step instructions.

```
for (let i = 0; i < 3; i++) {
    console.log(i);
}
```

- **Declarative**: Describes what to do.

```
[1, 2, 3].forEach(num => console.log(num));
```

- **Object-Oriented**:

```
class Person {
    constructor(name) {
        this.name = name;
    }
    greet() {
        console.log("Hello, " + this.name);
    }
}
```

- **Functional**:

```
const doubled = [1, 2, 3].map(n => n * 2);
```

---

7. Limitations of JavaScript

- **Security risks**: Vulnerable to XSS if not handled properly.
- **Performance**: Slower than compiled languages for heavy tasks.
- **Complexity**: Advanced features require deeper understanding.
- **Weak typing**: No strict type enforcement, which can lead to bugs.

JavaScript Introduction

8. ECMAScript Versions

JavaScript evolves through ECMAScript standards.

Version	Year	Key Features
ES5	2009	Strict mode, JSON support
ES6	2015	let/const, classes, arrow functions
ES7–ES13	2016–2022	async/await, BigInt, optional chaining
ES14	2023	toSorted, findLast, static blocks

---

9. Real-World Applications

- **Form validation**
- **Interactive games**
- **Animations and effects**
- **APIs and data fetching**
- **Full-stack web apps**

---

10. Example: Interactive Button

```
<button id="myBtn">Click Me</button>
<p id="message"></p>

<script>
    document.getElementById("myBtn").addEventListener("click", function() {
        document.getElementById("message").innerText = "Button was clicked!";
    });
</script>
```

JavaScript Introduction

11. Best Practices

- Use let and const instead of var.
- Write clear, descriptive variable names.
- Keep functions focused and reusable.
- Comment only when necessary.
- Use ES6+ features for cleaner code.

---

12. Introduction to Node.js

What Is Node.js?

Node.js is a **JavaScript runtime environment** that allows you to run JavaScript outside the browser, on your computer or server.

- Built on Chrome's **V8 engine**
- Enables **server-side development** using JavaScript
- Comes with **npm** (Node Package Manager) for installing libraries

Why Node.js?

Benefit	Description
Full-stack development	Use JavaScript for both frontend and backend
Fast and efficient	Handles many tasks without blocking
Rich ecosystem	Thousands of packages via npm
Real-time apps	Ideal for chat, games, and live updates

JavaScript Introduction

Node.js vs Browser JavaScript

Feature	Browser	Node.js
Environment	Web browser	Server/computer
Access	DOM, window, alert	Files, network, database
Use case	UI and interactivity	Backend logic and APIs

Simple Node.js Server

```
const http = require('http');
const server = http.createServer((req, res) => {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello, CodeFrill!');
});
server.listen(3000, () => {
    console.log('Server running at http://localhost:3000');
});
```

Run it with:

node server.js

Visit http://localhost:3000 in your browser to see the result.

---

13. Summary

JavaScript is:

- A **powerful, flexible language** for web development
- Used on both the **client-side** and **server-side** (via Node.js)
- Capable of supporting **multiple programming styles**
- Constantly evolving through **ECMAScript updates**
- The foundation of **modern web applications**