



Sheryians Coding
School

Live Cohort

Notes Day 14



Day 14 : Basics of JavaScript with ES6+

JavaScript

High-level, versatile language for web development.

Uses:

- DOM Manipulation → Modify HTML/CSS dynamically.
- Event Handling → Respond to clicks, key presses, etc.
- Asynchronous Communication → Fetch/send data (APIs, AJAX).
- Full-Stack → Client-side & server-side (Node.js).
- Cross-Platform → Web, mobile (React Native), desktop (Electron.js).

Example:

```
document.getElementById("btn").addEventListener("click", async () => {
  const response = await fetch("https://jsonplaceholder.typicode.com/todos/1");
  const data = await response.json();
  console.log(data);
});
```

Day 14 : Linking JavaScript Files Using <script>

Including an External JavaScript File

To link an external JavaScript file, use the <script> tag with the src attribute.

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Example</title>
</head>
<body>
    <h1>Welcome to JavaScript</h1>
    <script src="script.js"></script>
</body>
</html>
```

Best Practices

- Place <script> before </body> for better page load performance.
- Use defer for scripts that depend on HTML content.
- Use async for independent scripts that don't rely on DOM elements.

Day 14 : Logging with JavaScript

1. General Logging

Used to display information in the console.

```
console.log("Hello, World!"); // Outputs general info
```

2. Informational Message

Logs important information.

```
console.info("Info Message"); // Displays an informational log
```

3. Warning Message

Displays warnings in the console.

```
console.warn("Warning Message"); // Highlights a potential issue
```

4. Error Logging

Logs errors in the console.

```
console.error("Error Message"); // Shows an error message
```

5. User Input (Prompt)

Asks for user input via a pop-up.

```
let name = prompt("Enter your name:"); // Gets input from the user
```

6. Alert Message

Shows a message in an alert box.

```
alert("Hello!"); // Displays an alert pop-up
```

7. Confirmation Box

Asks the user to confirm an action.

```
let response = confirm("Are you sure?"); // Returns true/false
```

Day 14 : Variables and Keywords in JavaScript (var, let, const)

1. var

- Scope: Function-scoped.
- Characteristics: Allows redeclaration and updating.

```
var name = "John"; // Redeclaration and updates allowed
```

2. let

- Scope: Block-scoped.
- Characteristics: Allows updates, but not redeclaration in the same scope.

```
let age = 25; // Can be updated, but not redeclared in the same block
```

3. const

- Scope: Block-scoped.
- Characteristics: Cannot be updated or redeclared.

```
const country = "India"; // Cannot be reassigned or redeclared
```

Day 14 : Variable Declaration, Initialization, and Updating

1. Declaration

- Declaring a variable without initializing it.

```
let x; // Declaration
```

2. Initialization

- Assigning a value to the variable at the time of declaration.

```
let y = 10; // Initialization
```

3. Updating

- Updating the value of a variable.
- let and var allow updates, but const does not.

```
y = 20; // Updating
```

Example:

```
let score;          // Declaration
score = 100;       // Initialization
score = 150;       // Updating
console.log(score); // Output: 150
```

Day 14 : JavaScript Statements and Semicolons

Statements

- Statements are individual instructions in JavaScript, like variable declarations or function calls.

```
let a = 10; // Declaration
let b = 20; // Declaration
console.log(a + b); // Function call
```

Semicolons

- Semicolons are optional in JavaScript, but they help prevent errors, especially when statements are on the same line.
- They are recommended to avoid potential issues with automatic semicolon insertion.

```
let a = 10; // Declaration with semicolon
let b = 20; // Declaration with semicolon
console.log(a + b); // Function call with semicolon
```

Day 14 : Adding Comments in JavaScript

1. Single-line Comment

- Use // to add a comment on a single line.

```
// This is a single-line comment
```

2. Multi-line Comment

- Use /* */ for comments spanning multiple lines.

```
/*
  This is a
  multi-line comment
*/
```

Example:

```
// This is a single-line comment
let x = 5; // This comment is at the end of a line

/*
  This is a multi-line comment
  explaining the following code
*/
let y = 10;
```

Day 14 : Expressions in JavaScript and Their Difference from Statements

1. Expression

- An expression produces a value (e.g., arithmetic operations or variable assignments).

```
let sum = 5 + 10; // Expression (produces the value 15)
```

2. Statement

- A statement performs an action (e.g., variable declaration, function call).

```
console.log(sum); // Statement (performs an action)
```

Example:

```
let sum = 5 + 10; // Expression (produces a value)
console.log(sum); // Statement (executes an action)
```

Day 14 : JavaScript Operators

Arithmetic Operators

- + (Addition)
- - (Subtraction)
- * (Multiplication)
- / (Division)
- % (Modulus – Remainder of division)

Example:

```
let num1 = 10;  
let num2 = 5;  
  
console.log(num1 + num2); // Output: 15 (Addition)  
console.log(num1 % num2); // Output: 0 (Modulus)
```

Day 14 : + Increment & Decrement Operators

Increment (++)

- Increases the value of a variable by 1.

```
count++; // Increment
```

Decrement (--)

- Decreases the value of a variable by 1.

```
count--; // Decrement
```

Example:

```
let count = 5;  
count++;           // Increment  
console.log(count); // Output: 6
```