Day 10 – [2nd July 2025]

TOPICS COVERED

JavaScript is Single-Threaded:

JavaScript is a single-threaded programming language, meaning it can execute only one task at a time in the main thread. It runs tasks line by line, but thanks to the browser's Web APIs and the event loop, JavaScript can handle asynchronous behavior without blocking the main thread.

setTimeout() and setInterval():

These functions allow us to schedule code execution for the future.

setTimeout(callback, delay):

Executes the callback once after the delay (in ms).

setInterval(callback, interval):

Executes the callback repeatedly after every interval (in ms) until stopped.

Example:

```
console.log("Start");

setTimeout(() => {
  console.log("Timeout done");
}, 2000);

console.log("End");

// Output:

// Start

// End

// Timeout done
```

Here, even though the timeout is 2 seconds, the code doesn't wait — it continues execution. This is how asynchronous behavior works in single-threaded JS.

Fetch API with async and await:

The fetch() function is used to make HTTP requests (like getting data from an API).

By default, fetch() returns a promise, which we can handle using .then() or more cleanly using async and await.

Example:

```
async function loadData() {
  const response = await fetch("https://jsonplaceholder.typicode.com/posts/1");
  const data = await response.json();
  console.log(data);
}
loadData();
```

async makes a function return a promise.

await pauses the function execution until the promise is resolved.

This helps in writing clean, readable, step-by-step async code, without chaining multiple .then() blocks.

TOOLS USED:

Visual Studio Code

Chrome Developer Tools

Hoppscotch API Tester

TASK:

Read and understand:

try and catch blocks

How to handle errors in JavaScript

Use Hoppscotch to try an API request

➤ Visit https://hoppscotch.io

➤ Make a GET request

Observe the JSON response and match it with the output from fetch() in your JavaScript code