Training Day 10

2nd July 2025

TOPICS COVERED

- **JavaScript is Single-Threaded**

JavaScript is single-threaded, meaning it can execute one operation at a time in a single call stack. However, it can handle asynchronous tasks like timers, network requests, or events using the event loop, callback queue, and Web APIs provided by the browser.

- **setTimeout() –** Delayed Execution

Used to run a function once after a delay (in milliseconds).

Example:

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

setTimeout(() => {

  console.log("Executed after 2 seconds");

}, 2000);

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

Output:

```
Start

End

Executed after 2 seconds
```

Even though setTimeout has a 2-second delay, JavaScript doesn't wait—it keeps executing the rest of the code.

- **fetch() –** Handling API Calls

fetch() is used to make HTTP requests and works asynchronously, returning a Promise.

# Without async/await:

```
fetch("https://dog.ceo/api/breeds/image/random")

  .then(response => response.json())
  .then(data => console.log(data));
```

# With async/await: (Cleaner syntax)

```
async function getData() {
    let response = await fetch("https://dog.ceo/api/breeds/image/random");
    let data = await response.json();
    console.log(data);
  }
getData();
```

- **Why Use async/await?**

1.Makes asynchronous code look synchronous

2.Improves readability

3.Must be used inside a function declared with async

4.Use try...catch to handle errors

TOOLS USED

Visual Studio Code (VS Code)

Chrome Browser (JavaScript Console)

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