Training Day 10 Daily Dairy

Day 10 – 2nd July 2025

TOPICS COVERED – setTimeout, setInterval, API & Asynchronous JavaScript

1. setTimeout()

Used to delay the execution of a function by a given amount of milliseconds.

```
setTimeout(() => {
  console.log("Executed after 2 seconds");
}, 2000);
```

- Executes **once** after the delay.
- Can be canceled using clearTimeout().

2. setInterval()

Repeats the execution of a function at regular intervals.

```
setInterval(() => {
  console.log("Runs every 3 seconds");
}, 3000);
```

- Executes **repeatedly** after every interval.
- Can be stopped using clearInterval().

3. Asynchronous JavaScript

JavaScript is single-threaded. Asynchronous behavior helps in non-blocking execution using:

- **setTimeout**
- **setInterval**
- **Promises**
- **fetch API**
- **async/await**

This allows long-running operations like API calls to complete in the background.

4. Promises

Used to handle asynchronous operations.

```
let promise = new Promise((resolve, reject) => {
  let success = true;
  success ? resolve("Done") : reject("Error");
});

promise.then(data => console.log(data)).catch(err => console.log(err));
```

5. async / await

Modern syntax to write asynchronous code in a cleaner way.

```
async function fetchData() {
  let response = await fetch("https://api.example.com/data");
  let data = await response.json();
  console.log(data);
}
```

6. Introduction to APIs

API (Application Programming Interface) allows apps to communicate with external services.

- REST APIs work with HTTP methods like GET, POST, PUT, DELETE.
- We use fetch() to call APIs.

Example:

```
fetch("https://jsonplaceholder.typicode.com/posts")
  .then(res => res.json())
  .then(data => console.log(data));
```

7. Minor Concepts Covered

- **clearTimeout & clearInterval**: Cancel timers
- **Callback in setTimeout**: Using functions in delay
- **Basic API error handling** with .catch()
- **Browser Dev Tools → Network Tab**: To see API requests

CODE TASKS

1. Create a timer using setTimeout that shows a message after 5 seconds.
2. Use setInterval to update the time on the screen every second.
3. Fetch data from a public API (like JSONPlaceholder) and display it on the page.
4. Practice a function using both then/catch and async/await for API calls.

TASK FOR TOMORROW

- Deep dive into **event delegation**
- Learn about **localStorage & sessionStorage**
- Start with **form handling** in JavaScript