Industrial Training Daily Diary

MERN Stack Training

Company: Sensation Software Solutions
Student Name: Tajinder Kaur
Training Duration: 6 Months
Day: 9

Day 9 – JavaScript Promises, Async/Await & API Basics

Objective of the Day

The objective of Day 9 was to understand **asynchronous programming in JavaScript** using callbacks, promises, and **async/await**. The session also introduced the concept of **APIs** and how JavaScript communicates with external data sources. These concepts are critical for both frontend (React) and backend (Node.js) development in the MERN Stack.

Understanding Asynchronous JavaScript

The trainer explained the difference between **synchronous** and **asynchronous** execution:

- **Synchronous:** Code runs line by line and waits for each task to finish
- **Asynchronous:** Long-running tasks (like API calls) run in the background without blocking execution

We learned why asynchronous programming is important for better performance and user experience.

Callbacks – Introduction

Callbacks were explained as functions passed as arguments to other functions and executed after a task is completed.

Issues with Callbacks

- Callback nesting (callback hell)
- Reduced readability
- Difficult error handling

This led to the introduction of promises as a better solution.

JavaScript Promises

The trainer explained that a **Promise** represents the eventual completion or failure of an asynchronous operation.

Promise States

- Pending
- Fulfilled
- Rejected

We learned how to create promises and handle results using `.then()` and `.catch()` methods.

Async and Await

The trainer introduced **async/await**, which provides a cleaner and more readable way to handle asynchronous code.

Key Benefits

- Code looks synchronous
- Better error handling using `try...catch`
- Improved readability

We practiced converting promise-based code into async/await syntax.

Introduction to APIs

The trainer explained what an **API (Application Programming Interface)** is and how it allows applications to communicate with each other.

Real-Life Examples of APIs

- Weather applications
- Payment gateways
- Login using Google or Facebook
- Fetching data from servers

Fetch API Basics

We learned how to use the **Fetch API** to request data from a server.

Topics covered:

- Making GET requests
- Handling JSON responses
- Using fetch with promises
- Using fetch with async/await

Practical Work Done

During the practical session, we:

- Created promise-based functions
- Used async/await to handle asynchronous operations
- Fetched sample data from public APIs
- Displayed API data on a web page dynamically

This practice helped in understanding real-world data handling.

Importance in MERN Stack

The trainer explained that APIs are the backbone of full-stack applications. React uses APIs to fetch data from Node.js servers, and Node.js uses APIs to communicate with MongoDB databases.

Conclusion of Day 9

Day 9 strengthened my understanding of asynchronous JavaScript and API communication. I learned how to handle real-time data efficiently using promises and async/await, which are essential skills for modern web development.

Outcome of the Day:

- Understanding of asynchronous JavaScript
- Knowledge of promises and async/await
- Introduction to API communication
- Practical experience with Fetch API