👷 Day 9 Lab Sheet – Advanced JavaScript & TypeScript

Objective

By the end of this lab, you will:

- Work with Promises and async/await
- Handle errors using try...catch
- Use debugging tools (console, debugger)
- Write basic TypeScript code with types, interfaces, and classes

---

Task 1: Promise Basics

1. Create day9_promise.html.
2. Inside <script>:

```
let myPromise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve("Hello JavaScript (after 2 seconds)");
  }, 2000);
});

myPromise
  .then(result => console.log(result))
  .catch(error => console.error(error));
```

✅ After 2 seconds → console shows: *Hello JavaScript*.

Task 2: Async/Await Fetch Example

1. Create day9_async.html.
2. Add:

```
async function fetchPosts() {
  try {
    let response = await fetch("https://jsonplaceholder.typicode.com/posts");
    let posts = await response.json();
```

```
    console.log("First Post:", posts[0]);
  } catch (error) {
    console.error("Error fetching posts:", error);
  }
}
fetchPosts();
```

✅ Console shows the first post object.

Task 3: Error Handling

```
try {
  let num = 10 / 0;
  console.log("Result:", num);
  throw new Error("Custom error message");
} catch (error) {
  console.error("Caught Error:", error.message);
} finally {
  console.log("This always runs");
}
```

✅ Console shows error handling message.

Task 4: Debugging with debugger

```
let x = 5;
let y = 0;
debugger;  // open browser dev tools, code stops here
let result = x / y;
console.log("Result:", result);
```

✅ Execution pauses → step through in browser dev tools.

Task 5: TypeScript Variables & Functions

1. Create day9.ts.
2. Add:

```
let username: string = "Alice";
let age: number = 22;
let isStudent: boolean = true;

function greet(name: string): string {
  return "Hello, " + name;
}

console.log(greet(username));
```

3. Compile:

```
tsc day9.ts
```

4. Run day9.js in browser/Node.js.

✅ Console prints: *Hello, Alice*.

Task 6: TypeScript Interfaces & Classes

```
interface Car {
  brand: string;
  model: string;
  year: number;
}

let car1: Car = {
  brand: "Tesla",
  model: "Model 3",
  year: 2024
};
console.log(car1);
```

```
class Person {
  name: string;
  constructor(name: string) {
    this.name = name;
  }
  greet(): void {
    console.log("Hi, I am " + this.name);
  }
}

let p = new Person("Bob");
p.greet();
```

✅ Console shows Car object + greeting message.

Task 7: TypeScript Module Example (Bonus)

1. Create math.ts:

```
export function square(x: number): number {
  return x * x;
}
```

2. Create app.ts:

```
import { square } from "./math";
console.log("Square of 5:", square(5));
```

3. Compile with tsc --module commonjs app.ts.

✅ Console prints: *Square of 5: 25*.

✅ Deliverables

- day9_promise.html (Promise)
- day9_async.html (Async/Await)

- day9.ts (TypeScript basics)
- math.ts + app.ts (TypeScript modules – optional)

Each file should run correctly in browser (JS) or Node.js/TS compiler (TS).