Practical 3 - ES6 Promises

Aim

Write an ES6 Promise that:
e. Resolves after 2 seconds with a success message.
f. Rejects if a condition fails.
g. Consume the Promise using then() and catch().
h. Display appropriate success or error messages.

Theory

A Promise in JavaScript is an object that represents the eventual completion or failure of an asynchronous operation. It has three states:

1. Pending - Initial state, neither fulfilled nor rejected.
2. Fulfilled - The operation completed successfully (resolve is called).
3. Rejected - The operation failed (reject is called).

Promises are created using the Promise constructor which takes an executor function with two parameters: resolve and reject. We consume promises using .then() for success and .catch() for errors.

Code

```
// (e) Promise that resolves after 2 seconds with a success message
// (f) Rejects if a condition fails
const isConditionMet = true; // Change to false to test rejection

const myPromise = new Promise((resolve, reject) => {
    setTimeout(() => {
        if (isConditionMet) {
            resolve("Operation completed successfully!");
        } else {
            reject("Condition failed: something went wrong.");
        }
    }, 2000);
});

// (g) Consume the Promise using then() and catch()
// (h) Display appropriate success or error messages
myPromise
    .then((successMsg) => {
        console.log("Success:", successMsg);
    })
    .catch((errorMsg) => {
        console.log("Error:", errorMsg);
    });
```

Output

When isConditionMet = true:

Practical 3 - ES6 Promises

```
Success: Operation completed successfully!
```

When isConditionMet = false:

```
Error: Condition failed: something went wrong.
```

Conclusion

In this practical, we learned how to create and consume ES6 Promises. The Promise resolves after 2 seconds if the condition is met, and rejects otherwise. Using .then() we handle the success case, and using .catch() we handle the error case. This pattern replaces deeply nested callbacks (callback hell) with cleaner, more readable code.