Asynchronization

- Write JavaScript code that demonstrates the use of callback functions with asynchronous operations. The assignment will involve creating and using callbacks to handle data from simulated asynchronous tasks.

Instructions:

1. Create a Simulated Asynchronous Task:
   - Write a function fetchData that simulates fetching data from a server. Use setTimeout to simulate a delay in data retrieval. The function should accept a callback function that will be called with the fetched data.
2. Define Callback Functions:
   - Define two callback functions:
     - displaySuccess: This function should handle and display successful data retrieval.
     - displayError: This function should handle and display errors if data retrieval fails.
3. Invoke the Asynchronous Function:
   - Call fetchData with both displaySuccess and displayError as arguments. Simulate both successful and unsuccessful data retrieval scenarios.
4. Simulate Different Scenarios:
   - Update fetchData to randomly simulate success or failure. Ensure that your callback functions handle both outcomes appropriately.
5. Update the DOM:
   - Create an HTML page with a element to display messages. Update this based on the outcome of the asynchronous operation using your callback functions.

setTimeout () function :

- Program to display a text using setTimeout method
- program to display time every 3 seconds
- program to stop the setTimeout() method

Exercise: Creating a Timer

1. **Create a function** called `startTimer` that:
   - Takes a single parameter: the number of seconds you want the timer to run.
   - Uses `setInterval` to log the remaining time every second.
   - Stops the timer when the countdown reaches zero.
2. **Create another function** called `stopTimer` that:
   - Clears the interval and stops the timer.
3. **Integrate both functions** so that the timer starts when `startTimer` is called and can be stopped by calling `stopTimer`