Exercise: JavaScript Async Functions I

1. Hello World

Using **`setTimeout`** with callbacks, write a function **`helloWorld()`** which logs two messages to the console with a time delay between them. You have to log the first message immediately and the second message after a 2-second delay.

Requirements

- The first message, "**`Hello`**", should be logged to the console immediately when the script runs.
- The second message, "**`World`**", should be logged to the console exactly 2 seconds after the first message.

Hints

- Start by logging the message "**`Hello`**" to the console. This message should appear immediately when the script runs.
- Use the **`setTimeout`** function to schedule another message, "**`World`**", to be logged to the console after a 2-second delay.

2. Chained Promises

Using **`setTimeout`** with **callbacks**, write a function **`chainedPromises()`** which logs a series of messages to the console with increasing delays. The function must log "**`Start`**" immediately, then "**`1`**" after 1 second, "**`2`**" after 2 seconds, and "**`3`**" after 3 seconds.

Requirements

- "**`Start`**" should be logged immediately.
- "**`1`**" should be logged 1 second after "**`Start`**".
- "**`2`**" should be logged 2 seconds after "**`Start`**".
- "**`3`**" should be logged 3 seconds after "**`Start`**".

Hints

- Start by logging the message "**`Start`**" to the console. This message should appear immediately when the script runs.
- Use **`setTimeout`** to schedule logging the messages, holding the numbers.

3. Simple Promise

Using promises, write a function **`simplePromise()`** which creates a promise that resolves to "**`Success!`**" after 2 seconds and logs the result to the console.

Requirements

- The promise should resolve to "**`Success!`**" after 2 seconds.
- "**`Success!`**" should be logged to the console.

Hints

- Create a new promise that resolves to the string "**`Success!`**" after a delay of 2 seconds.

- Use **`.then`** to log the resolved value of the promise to the console.

4. Promise Rejection

Using promises, write a function **`promiseRejection()`** which creates a promise that rejects with an error message after 1 second and logs the error.

Requirements

- The promise should reject with an error message "**`Something went wrong!`**" after 1 second.
- The error message should be logged to the console.

Hints

- Use .**`catch`** to log the error message to the console.

5. Promise with Multiple Handlers

Using promises, write a function **`promiseWithMultipleHandlers()`** which creates a promise that resolves to "**`Hello World`**" after 2 seconds and logs the message twice using multiple **`.then`** handlers.

Requirements

- The promise should resolve to "**`Hello World`**" after 2 seconds.
- "**`Hello World`**" should be logged twice to the console.

Hints

- Use **`.then`** to log the resolved value of the promise to the console twice.

6. Promise All

Using **`Promise.all`**, write a function **`allPromise()`** code which creates three promises that resolve after **1**, **2**, and **3** seconds respectively and logs all results together when they are all resolved.

Requirements

- Create three promises with specified delays.
- Use **`Promise.all`** to wait for all promises to resolve.
- Log all results together when they are all resolved.

7. Promise Race

Using **`Promise.race`**, write a function **`racePromise()`** which creates three promises that resolve after **1**, **2**, and **3** seconds respectively and logs the first resolved result.

Requirements

- Create three promises with the specified delays.
- Use **`Promise.race`** to log the first resolved result.

8. Async Function with Await

Using **`async/await`**, write an async function **`simplePromiseAsync()`** that waits for 2 seconds and then logs "**`Async/Await is awesome!`**"

Requirements

- Create an **`async`** function.
- Use **`await`** to wait for 2 seconds.
- Log the message "**`Async/Await is awesome!`**" after the delay.

9. Async Function with Error Handling

Using **`async/await`**, write an async function **`promiseRejectionAsync()`** which throws an error and catches it, logging the error message.

Requirements

- Create an async function that throws an error.
- Catch the error and log the error message.

10. Chained Promises with Async/Await

Using **`async/await`**, write an async **`chainedPromisesAsync()`** function that waits for three promises that resolve after **1**, **2**, and **3** seconds respectively and logs their results in order.

Requirements

- Create an async function.
- Wait for three promises with specified delays.
- Log the results of the promises in order.

11. Quiz

In this exercise, we will write a simple quiz game that runs in the console. The game will ask a series of questions, accept user input for answers, and provide feedback on whether the answers are correct or incorrect. The game will tally the score and display it at the end.

Requirements

- Use promises and **`async/await`** to handle asynchronous operations.
- Display each question and possible answers to the user.
- Accept user input for answers and provide feedback on correctness.
- Tally the score and display the final result.

12. Simple Stopwatch

In this exercise, we will create a simple stopwatch that starts counting elapsed time in seconds when the "**`Start Stopwatch`**" button is clicked and stops when the "**`Stop Stopwatch`**" button is clicked. The elapsed time will be displayed in the console. Additionally, the stopwatch will simulate saving the elapsed time every 5 seconds using a promise.

Requirements

- The stopwatch should start counting when the "**`Start Stopwatch`**" button is clicked.
- The elapsed time should be displayed in the console every second.

- The stopwatch should stop counting when the "**`Stop Stopwatch`**" button is clicked.
- The elapsed time should reset when the stopwatch is stopped.

13. Countdown Timer

In this exercise, we will create a countdown timer that starts counting down from **n** seconds when the "**`Start Countdown`**" button is clicked. The remaining time will be displayed in the console. Additionally, the timer will simulate saving the remaining time asynchronously when the countdown reaches zero.

Requirements

- The countdown should start from the input number **n** of seconds when the "**`Start Countdown`**" button is clicked.
- The remaining time should be displayed in the console every second.
- The countdown should stop when the timer reaches zero and display a finished message.
- The remaining time should be saved asynchronously when the countdown finishes.

14. Simple Text Adventure Game

In this exercise, we will create a simple text-based adventure game. The game will guide the player through a series of choices that lead to different outcomes. The player will input their choices through prompts, and the game will respond based on their decisions.

Requirements

- The game should start when the "**`Start Adventure`**" button is clicked.
- The game should prompt the player to make choices using the **`prompt`** function.
- The player's choices should determine the next scene in the game.
- The game should include simulated delays using promises and **`setTimeout`**.
- The game should handle different scenarios and provide feedback based on the player's decisions.
- The game should allow the player to restart or end the game based on their input.