Exercise 1 - Create and Handle a Simple Promise with Resolution and Rejection

Overview

In this exercise, you will create a JavaScript function that leverages the Promise object. The Promise will either resolve or reject based on the state of a checkbox provided in the HTML. If the checkbox is checked, the Promise will reject; otherwise, it will resolve. The outcome will be displayed in a message element on the page, with the text color changing based on the result.

Instructions

1. **Understand the HTML and CSS**: Before diving into the JavaScript, ensure that you understand the structure of the HTML and CSS provided. You will be working with a checkbox element with the ID `triggerRejection` and a button with the ID `triggerPromise`. The outcome of the Promise will be displayed in a div with the ID `message`.

2. **Set Up an Event Listener**: Begin by setting up an event listener for the button with the ID `triggerPromise`. This listener will trigger a function when the button is clicked.

3. **Create a Promise**: Inside the function triggered by the button click, create a new Promise. Utilize the `setTimeout` method to simulate a delay, such as 1000 milliseconds (1 second).

4. **Resolve or Reject the Promise**: Use the `if` statement to check the state of the checkbox with the ID `triggerRejection`. If the checkbox is `checked`, reject the Promise with an appropriate error message like 'The Promise has been rejected!'. If not `checked`, resolve the Promise with a success message like 'The Promise has been resolved!'.

5. **Handle the Promise Resolution and Rejection**: After finishing the creation of the Promise, chain the `.then` and `.catch` methods to the Promise to handle the resolved and rejected outcomes.

   - In the `.then` method, pass a callback function that recieves a `message` parameter. Inside this callback function, update the text content of the message element (ID `message`) with the success message that has been passed to this callback function.

Add a `resolved` class to change the text color to green, and ensure to remove the `rejected` class that might set it to red if it was previously rejected.
    - In the `.catch` method, pass a callback function that recieves an 'error' parameter. Update the content of the message element with the 'error' passed to the callback function. Add a class `rejected` to change the text color to red, and ensure to remove the `resolved` class that might set it to green if it was previously resolved.

6. **Testing Your Code**: Test your code by clicking the button with and without the checkbox selected. Ensure that the message and text color change as expected.

Tips

- Utilize the `document.getElementById` method to interact with specific elements in the HTML.
- Use the `classList.add` and `classList.remove` methods to dynamically add or remove classes from elements.
- Remember that Promises are a way to handle asynchronous operations, and think about how the `setTimeout` method is used to simulate this.
- Carefully follow the logic and understand the flow of execution, paying particular attention to the asynchronous nature of Promises.