Exercise 5 - Creating a Loading Indicator with JSONPlaceholder

Objective

In this exercise, you will implement JavaScript to fetch user data from JSONPlaceholder and display it on the page. You will also create a loading indicator that will be shown while the data is being fetched and hidden once the data is loaded. This will provide a more engaging user experience.

You can use some code from earlier exercises for this exercise.

Instructions

1. **Setup the URLs**: Define the URL for fetching user data from JSONPlaceholder. You will need this URL to make the fetch request.
2. **Select HTML Elements**: Using JavaScript, select the necessary HTML elements, such as the container for the users and the loading indicator.
3. **Display Loading Indicator**: Initially, set the display property of the loading indicator to 'block' so that it's visible to the users when the data fetching begins.
4. **Make a Fetch Request**: Make a fetch request to the JSONPlaceholder user URL. This request will return a Promise.
5. **Handle the Response**: Once you receive the response, you need to process it by converting it to JSON format. Remember to handle both successful and unsuccessful responses appropriately.
6. **Process the Data**: Loop through the array of users returned from JSONPlaceholder. For each user, create a new div element, add a class to it, and set its inner HTML to display the user's name and email, like in the previous exercises. Append each div to the users' container.
7. **Hide the Loading Indicator**: Once the users are successfully loaded and displayed, in the scond `then`, hide the loading indicator by setting its display property to 'none'.

8. **Handle Errors**: In case of an error, hide the loading indicator by setting its display property to 'none' and display an appropriate error message.

9. **Test with Network Throttling**: To understand how the loading indicator works, open the browser's developer tools and use the network throttling feature to simulate a slow network. Observe how the loading indicator behaves.

Tips & Considerations

- Be mindful of the Promise structure and the asynchronous nature of fetch requests.
- Use appropriate methods for handling Promises, such as `.then()` and `.catch()`.
- Focus on clean and maintainable code. Break down the process into smaller functions if needed.

Challenge

- Try fetching additional data from JSONPlaceholder, such as posts or comments, and display them similarly.
- Enhance the user experience by adding loading gifs, transitions or animations to the loading indicator.