Exercise 4 - Loading Indicator (Async/Await Version)

Objective

In this revised version of the exercise, you will use the `async/await` syntax to handle Promises and make the code more readable. Your task remains the same: to fetch user data from JSONPlaceholder and to show and hide a loading indicator appropriately.

You can use some of the code from the fifth exercise in the 'Promises' section.

Instructions

1. **Setup the URLs**: The URL to fetch user data is already defined as `usersURL`.
2. **Select HTML Elements**: Use `getElementById` to select the users container and the loading indicator.
3. **Create an Asynchronous Function**: Write an `async` function named `fetchUsers` that will handle the data fetching and processing.
4. **Display Loading Indicator**: Inside the `fetchUsers` function, set the loading indicator's display property to 'block' initially.
5. **Make an Asynchronous Fetch Request**: Use the `await` keyword to make a fetch request to the `usersURL`.
6. **Handle the Response with Async/Await**: Process the response with `await` to convert it to JSON. This step must be inside a `try` block to catch any errors.
7. **Process the Data**: Iterate through the users and create div elements to display each user's information. This part remains similar to the earlier version.
8. **Hide the Loading Indicator**: Inside a `finally` block, set the loading indicator's display property to 'none'. This ensures that the indicator is hidden whether the fetch is successful or not.
9. **Handle Errors with Try/Catch**: Use a `catch` block to handle any errors. Display an appropriate error message if an exception occurs.

10. **Invoke the Async Function**: Outside of the function, call `fetchUsers()` to execute the code.

Challenge

- Enhance the user experience by adding styling to the loading indicator or using a loader gif.