Exercise 3 - Promise Error Handling with Fetch API

Objective:

To create a JavaScript function that fetches users from the JSONPlaceholder API and displays them on the webpage. If there's an error, it will display a nicely styled error message on the screen.

Requirements:

You are provided with the HTML and CSS for this exercise. Your task is to write the JavaScript code to achieve the objective above.

Instructions:

1. **Get the Containers**: Start by selecting the div elements that will contain the users and the error message. Use `document.getElementById` method.
2. **Fetch the Data**: Utilize the `fetch` method to get users from the JSONPlaceholder API. The endpoint is `'https://jsonplaceholder.typicode.com/users'`.
3. **Handle the Response**: After fetching, you must handle the response. Check if the response is OK. If not, throw an error with a message like `'Network response was not ok'`.

   Note: You can introduce an error by using an incorrect URL or by shutting down your server.
4. **Parse the JSON**: If the response is OK, parse it to JSON using the `response.json()` method.
5. **Iterate Through Users**: Once you have the JSON, iterate through the users, creating new div elements for each user. Include their name and email in the content.
6. **Append Users to Container**: Append the created div elements to the users' container on the page.
7. **Handle Errors**: Use a `catch` block to handle errors. This block will catch any errors thrown in the previous steps, including response errors and JSON parsing errors. Display the error message in the error container on the page.

Note: You can experiment with different types of errors by changing the endpoint URL to an incorrect one, or you can manually throw an error in the code.

Error Handling Insights:

Errors can occur in several ways:

- **Network Errors**: Such as no internet connection or failure to reach the server.
- **Response Errors**: The server might respond with an error status code.
- **Parsing Errors**: If the server does not return valid JSON, parsing it will result in an error.

Tips for Debugging:

- Use `console.error` to log the errors in the catch block.
- Use the browser's developer tools to see the network requests and responses.

Expected Outcome:

On successful execution, the page should display the users fetched from the API. If an error occurs, a nicely styled error message should be displayed on the screen.