Exercise 3 - Promise Error Handling with Fetch API

Objective:

Create a function that fetches users from the JSONPlaceholder API and displays them on the webpage, handling any errors gracefully.

Instructions:

1. **Get the Containers**:
    - Select the two div elements that will contain the users and the error message using `document.getElementById`.
2. **Fetch the Data**:
    - Make a fetch call to the JSONPlaceholder API at the specified endpoint to retrieve user data.
3. **Handle the Response**:
    - In the first `.then` block, check if the response is OK.
    - If the response status is not OK, throw an error with the specific message provided.
    - Introduce an error by using an incorrect URL or shutting down your server to see how the error is handled.
4. **Parse the JSON**:
    - If the response is OK, proceed to parse the response into JSON format.
5. **Iterate Through Users**:
    - In the next `.then` block, iterate through the array of users obtained from the JSON.
    - For each user, create a new div element, add the class `user`, and structure the inner HTML to include the user's name and email. Put the user's name inside an `H1` tag, and the user's email inside a `p` tag and surround it with a `strong` tag.

```
userDiv.innerHTML = `<h2>${user.name}</h2><p><strong>Email:
</strong>${user.email}</p>`;
```

6. **Append Users to Container**:
    - Append each user div element to the previously identified users' container, adding them to the page.
7. **Handle Errors**:
    - Implement a `catch` block at the end of the Promise chain to catch any errors.
    - The error caught could result from several stages in the process, including fetching, response checking, or JSON parsing.
    - Display a well-structured error message (`An error occurred: ${error.message}. Please try again later.`) in the error container on the webpage by setting the `textContent` of the error container.

Error Handling Insights:

- **Network Errors**: These could happen when the server is unreachable or if there's no internet connection.
- **Response Errors**: When the server responds with an error status code.
- **Parsing Errors**: Occur if the server's response is not valid JSON.

Tips for Debugging:

- Inside the catch block, use `console.error` to log the errors.
- Regularly check the browser's developer tools for insights into network requests and responses.

Expected Outcome:

- On successful execution, the page will display the users fetched from the API.
- If any errors occur at any stage of the process, the page will display a well-structured error message.

Guidelines:

- Ensure that the error handling covers all possible failure points, including network issues, server response errors, and JSON parsing.

- Test different error scenarios to verify that the error handling is robust and user-friendly.
- Focus on clear and concise code, using appropriate error messages that a user would understand.