Exercise 4 - Promise.all Practice with JSONPlaceholder

Objective:

Your task is to fetch data concurrently from two different endpoints of the JSONPlaceholder API: the users and the posts. You will need to display this data in a structured format on the HTML page. Use the Promise.all method to handle these promises concurrently.

Requirements:

Part 1: Defining the URLs

1. **Define the URLs**: Create two constants, `usersURL` and `postsURL`, to store the endpoints for the users and posts from JSONPlaceholder.

Part 2: Creating Promises

2. **Fetch Users**: Create a promise for fetching users from the `usersURL`. Use the `fetch` method and chain a `.then` to parse the response to JSON.
3. **Fetch Posts**: Similarly, create a promise for fetching posts from the `postsURL`.
4. **Promise.all**: Pass an array with the above Promises to the `Promise.all` to handle both promises concurrently.

Part 3: Handling the Responses

5. After the `Promise.all()`, in the `then`, destructure the resolved promises to a `users` and a `posts` variables:

   ```
   Promise.all([usersPromise, postsPromise])
         .then(([users, posts]) => {
           // Rest of the code...
         })
   ```

6. **Handle the Users Data**: Once both promises are fulfilled, use the received users data to: a. Select the users container div from the HTML (with id 'users'). b. Iterate through the users,

creating div elements for each user with the class 'item'. c. Include the user's name and email inside the div, and append it to the users container.

```
const userDiv = document.createElement('div');
userDiv.classList.add('item');
userDiv.innerHTML = `<strong>${user.name}</strong><br>${user.email}`;
usersContainer.appendChild(userDiv);
```

7. **Handle the Posts Data**: Similarly, for the posts data: a. Select the posts container div from the HTML (with id 'posts'). b. Iterate through the posts, creating div elements for each post with the class 'item'. c. Include the post's title and body inside the div, and append it to the posts container.

Part 4: Error Handling

7. **Handle Errors**: In case any of the promises are rejected, handle the error by: a. Selecting the error container div from the HTML (with id 'error'). b. Setting its text content to display an appropriate error message.

Ways an Error Can Occur:

- Network failure or an incorrect URL.
- The response was not OK (e.g., a 404 Not Found status).
- An error in the processing of the promises.

Tips:

- Make use of the `fetch` method to get the data from the API.
- Remember to parse the response to JSON format using `response.json()`.
- Use `Promise.all` to wait for all promises to be fulfilled.
- Make extensive use of DOM manipulation methods like `getElementById`, `createElement`, `appendChild`, etc.

Expected Outcome:

Upon successful completion, your JavaScript code will fetch users and posts from the JSONPlaceholder API and display them neatly on the webpage. Any occurring error will be displayed in a designated error section.

Challenge:

- Explore more endpoints from JSONPlaceholder and expand the functionality.