Exercise: Refactoring the Movie Rating List to use Async/Await

Objective:

In the previous exercise, you created a web page that fetches movie details from an external API using Promises and the `.then()` syntax. Now, your task is to refactor the `fetchMovies` function to use the modern `async/await` syntax, which will make the code more concise and easier to read.

Steps:

1. **Modify the `fetchMovies` Function**:

We'll keep everything else in the project the same and focus solely on refactoring the `fetchMovies` function.

a. Declare the Function as Async:

First, declare the function as an `async` function by adding the `async` keyword before the `function` keyword.

```
async function fetchMovies() {
  // ...
}
```

b. Use the `await` Keyword:

Inside the `fetchMovies` function, you'll replace the `.then()` calls with the `await` keyword.

1. **Await the fetch Call**: You'll use the `await` keyword when calling the `fetch` function. This will pause the execution of the code until the Promise is resolved, and the result will be stored in a variable.

2. **Await the Response Transformation**: Next, you'll need to transform the response into JSON. You can use `await` again for this purpose:

```
const data = await response.json();
```

c. Process the Results:

Now, you can process the results just as you did in the previous exercise, using the data from the API.

```
data.results.forEach(item => {
  // Your previous code here...
});
```

d. Handle Errors with Try/Catch:

Instead of the `.catch()` method, you'll use a `try/catch` block to handle errors in an `async` function.

```
try {
  // Your fetch and processing code here...
} catch (err) {
  console.error(err);
}
```

Conclusion:

Refactoring the `fetchMovies` function to use `async/await` makes the code more concise and easier to follow. By following these steps, you've learned how to work with modern asynchronous JavaScript features.

Challenge:

- Extend the `createErrorMessage` function (if you implemented it in the previous challenge) to handle errors in a more user-friendly way.

Remember that `async/await` is just syntactic sugar on top of Promises and provides a cleaner way to handle asynchronous operations, especially when dealing with multiple asynchronous operations that depend on each other.