

Task 6: Fetching User Data with Promises

Objective:

Create a function that fetches user data from a public API using Promises. The function should handle successful responses and errors. It should return the user's name and email if the fetch is successful, or an error message if the fetch fails.

Pre-requisites:

- Basic JavaScript (variables, functions, objects)
- Understanding of Promises
- Fetch API

Concepts Covered:

- Promises
- Fetch API
- Error handling in asynchronous operations

Setup:

Install Node.js:

- Ensure Node.js is installed on your machine. You can download it from nodejs.org.

Tasks:

1. Define the `fetchUserData` Function:

- **Task:**
 - Define a function named `fetchUserData` that returns a Promise.
 - The function should make a GET request to the following URL: `https://jsonplaceholder.typicode.com/users/1`.
 - If the request is successful, the function should resolve with an object containing the user's name and email.
 - If the request fails, the function should reject with an appropriate error message.
 - Ensure proper error handling within the function to manage network or other errors.
- **Outcome:**
 - Ensure the function correctly handles successful responses and errors.

Example:

JavaScript File (`fetchUserData.js`):

```
function fetchUserData() {
  return new Promise((resolve, reject) => {
    fetch('https://jsonplaceholder.typicode.com/users/1')
      .then(response => {
        if (!response.ok) {
          throw new Error('Network response was not ok');
        }
        return response.json();
      })
      .then(data => {
        resolve({ name: data.name, email: data.email });
      })
      .catch(error => {
        reject('Error fetching user data');
      });
  });
}

// Example Usage:
fetchUserData()
  .then(data => console.log(data)) // Output: { name: "Leanne Graham",
  email: "Sincere@april.biz" }
  .catch(error => console.log(error)); // Output: Error fetching user data
```

Instructions:

- **Perform the following tasks:**
 - Write the required code in `fetchUserData.js`.
 - Run the file using Node.js to ensure the code executes without errors and demonstrates the use of Promises and the Fetch API.





Example Input:

1. Calling `fetchUserData()`


Expected Output:

1. On Success: `{ name: "Leanne Graham", email: "Sincere@april.biz" }`
2. On Error: `"Error fetching user data"`

Resources:


-  **Using promises - JavaScript | MDN**
A Promise is an object representing the eventual completion or failure of an asynchronous...
 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises
-  **Fetch API - Web APIs | MDN**
The Fetch API provides an interface for fetching resources (including across the network). I...
 https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

•



Control flow and error handling - JavaScript | MDN

JavaScript supports a compact set of statements, specifically control flow statements, th...

 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling

Videos:

•


JavaScript Promises In 10 Minutes





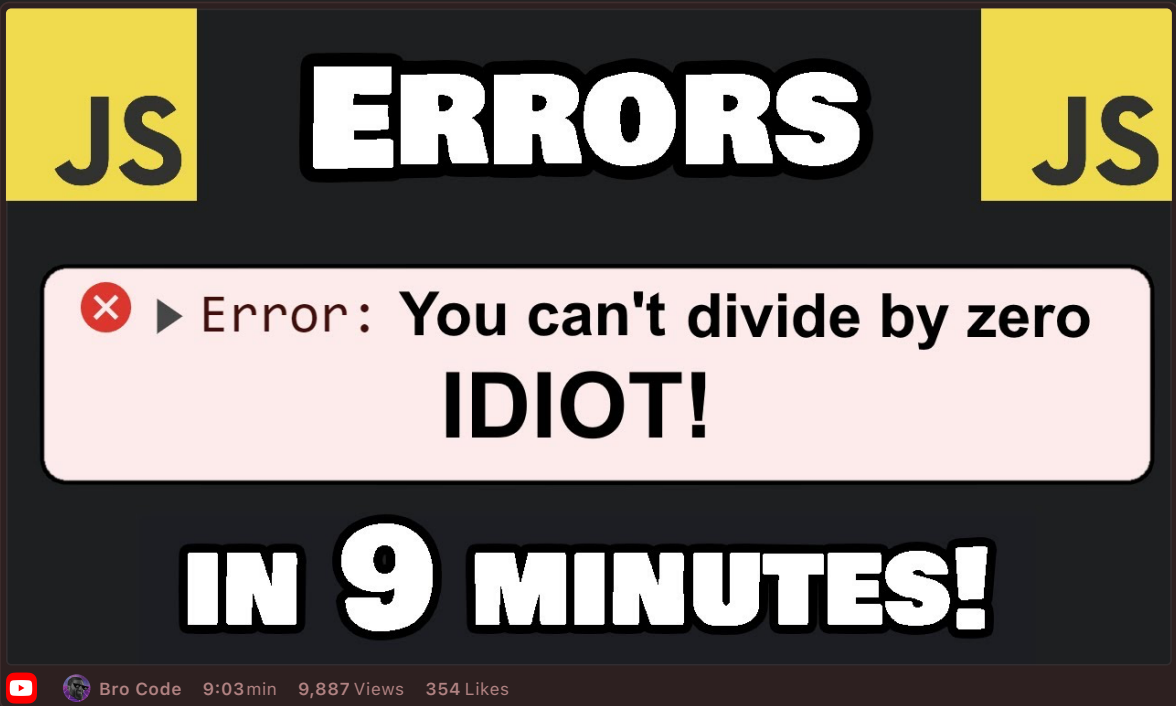
  Web Dev Simplified 11:31min 1,757,162 Views 53,678 Likes

•

EVENT LOOP IN JAVASCRIPT



  Coding Nomad 3:14min 34,817 Views 944 Likes



GitHub Instructions:

1. Open in Visual Studio Code:

- After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, Visual Studio Code (VSCode) will open the repository directly.
- If prompted, select "Open" or "Allow" to open the repository in VSCode.

2. Open the Terminal in VSCode:

- In VSCode, open a terminal by selecting Terminal > New Terminal from the top menu.

3. Complete the Task:

- In VSCode, write your solution in the `fetchUserData.js` file.

4. Run and Test Your Code:

- In the VSCode terminal, navigate to the directory containing `fetchUserData.js`.
- Run your code to ensure it works correctly. Use the following command:

```
node fetchUserData.js
```

5. Commit Your Changes:

- In the VSCode terminal, add your changes to git:

```
git add fetchUserData.js
```

- Commit your changes with a meaningful message:

```
git commit -m "Completed task 6"
```

6. Push Your Changes to Your Repository:

- Push your changes to your forked repository:

```
git push origin main
```

7. Create a Pull Request:

- Go to your repository on GitHub.
- Click on the "Pull Requests" tab.
- Click the "New Pull Request" button.
- Ensure the base repository is the original template repository and the base branch is `main`.
- Ensure the head repository is your forked repository and the compare branch is `main`.
- Click "Create Pull Request".
- Add a title and description for your pull request and submit it.

Summary of Commands:

```
# Open in Visual Studio Code

# Open terminal in VSCode

# Complete the task by editing fetchUserData.js

# Navigate to the directory containing fetchUserData.js
cd path/to/your/fetchUserData.js

# Run your code
node fetchUserData.js

# Add, commit, and push your changes
git add fetchUserData.js
git commit -m "Completed task 6"
git push origin main

# Create a pull request on GitHub
```