Debugging Approach

1. Syntax Errors

Issue:

The code contained a syntax error in the app.js file.

Debugging Steps:

Reviewed the app.js file line by line.
Identified the syntax error: setTimout instead of setTimeout.
Corrected the syntax by changing setTimout to setTimeout.

2. Logical Errors

Issue:

The code contained a logical error in the fetchUsers function.

Debugging Steps:

Examined the fetchUsers function.
Noticed that the callback function passed to setTimeout had a typo: setTimout instead of setTimeout.
Corrected the typo to setTimeout to ensure the callback function is executed after the specified delay.

3. Data Fetching Errors

Issue:

The fetchUsers function simulated data fetching but had a delay of 1 second.

Debugging Steps:

Analyzed the fetchUsers function.
Observed that it used setTimeout to simulate an asynchronous API call.
Reduced the delay from 1000 milliseconds to 0 milliseconds to make the data appear immediately for debugging purposes.

4. Additional Debugging

Issue:

The displayUsers function contained a typo: innerhtml instead of innerHTML.

Debugging Steps:

Examined the displayUsers function.

Identified the typo in userDiv.innerhtml.
Corrected the typo to userDiv.innerHTML to correctly set the HTML content of the user's name.

5. Debugging Output

Debugging Output:

- Throughout the debugging process, used console.log statements to print variables, function calls, and debugging messages to the browser's console.
- Inspected the browser's console for error messages and identified issues in the code.
- Made incremental changes and tested the code after each change to verify that it was working correctly.

By following these steps, the code was successfully debugged, and the identified issues were resolved. The corrected code now simulates data fetching, displays users, and is ready for further feature enhancements.