Assignment 2: Rewriting Fetch with Promises

Lesson Duration: ~ 2 hours

Assignment Overview

This assignment is designed to deepen your understanding of how asynchronous JavaScript operates, specifically through Promises and async/await, by rewriting a custom fetch function. The objective is to clearly see how Promises underlie modern async/await syntax.

Learning Objectives

- Gain practical experience in handling asynchronous operations with Promises.
- Understand the internal workings of JavaScript's fetch API.
- Demonstrate clear knowledge of the relationship between async/await and Promises.

Assignment Tasks (1.5 hours)

Task 1: Analyze Provided Code

- Carefully examine the provided `code.js`, which currently uses `async/await`:

```
async function getData() {
  const response = await fetch("./data/test.json");
  return await response.json();
}

async function foo1() { ... }
async function foo2() { ... }
async function foo3() { ... }

async function main() {
  app.textContent = "Fetching user data...";
  const user = await foo3();
  app.textContent = `User data: ${JSON.stringify(user)}`;
}

btn.addEventListener("click", () => {
  main();
});
```

- Understand clearly the flow of asynchronous data retrieval.

Task 2: Rewrite Using Promises

- Rewrite the provided code without using `async/await`. You must use Promises (`then` and `catch` methods) exclusively.
- Modify the existing functions (`getData`, `foo1`, `foo2`, `foo3`, `main`) to correctly handle Promises.

```
function getData() {
  return fetch("./data/test.json");
}

function foo1() { ... }
function foo2() { ... }
function foo3() { ... }

function main() { ... }
```

Task 3: Implement the `run` Function

- Define and implement the `run` function using Promises (`then/catch`) without async/await:

```
function run(func) {
  // Your Promise-based implementation here
}
```

- This function should execute your Promise-based `main` function, correctly handle fetch results, and manage any errors gracefully.

Task 4: Testing and Validation

- After completing the implementation, uncomment the provided testing line in `replace_await.html`.
- Test thoroughly to confirm correct functionality and robust error handling.

Task up the solution (30 min.)

- Submit the fully functional JavaScript file (`task.js`) with your Promise-based implementation.
- Clearly comment your code, explaining each Promise and handling decision.
- Provide brief written reflections on what you learned about async/await and Promises.