Fetch API Data Display

This assignment is an excellent way to practice working with APIs, asynchronous JavaScript, and DOM manipulation. Let's break it down step by step before providing the code.

Steps to Complete the Assignment

1. Understanding the Fetch API

The `fetch()` function is used to make HTTP requests. It returns a `Promise` that resolves to a `Response` object, which we can then convert into JSON.

2. Choosing a Public API

We'll use **JSONPlaceholder**, a free fake online REST API. Specifically, we'll fetch a list of posts from:
👉 `https://jsonplaceholder.typicode.com/posts`

3. Displaying the Data

Once the data is fetched, we'll dynamically create HTML elements (like a table or list) and display the API results on the webpage.

Features to Implement

✅ Fetch data from the API asynchronously
✅ Display data dynamically in a structured format (table)
✅ Handle errors (like network issues)
✅ Provide a button to trigger data fetching

Code Implementation

We'll write the code in three separate files:

1. `index.html` – The webpage structure
2. `styles.css` – Styling
3. `script.js` – JavaScript to fetch and display data

---

1. index.html

This file contains the structure of our webpage, including a **button** to fetch data and a **table** to display the results.

```
html
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fetch API Example</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <h1>API Data Fetching</h1>
    <button id="fetchData">Fetch Posts</button>

    <table id="dataTable">
        <thead>
            <tr>
                <th>ID</th>
                <th>Title</th>
                <th>Body</th>
            </tr>
        </thead>
        <tbody>
            <!-- Data will be inserted here dynamically -->
        </tbody>
    </table>

    <script src="script.js"></script>
</body>
</html>
```

---

2. styles.css

This file improves the look of the webpage.

```
css
```

```css
body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin: 20px;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 5px;
    margin-bottom: 20px;
}

table {
    width: 80%;
    margin: 0 auto;
    border-collapse: collapse;
}

th, td {
    border: 1px solid black;
    padding: 10px;
}

th {
    background-color: #007bff;
```

```
    color: white;
}
```

---

3. script.js

This file fetches data and updates the table dynamically.

```
js

document.getElementById("fetchData").addEventListener("click", fetchData);

async function fetchData() {
    try {
        // Fetch data from API
        const response = await fetch("https://jsonplaceholder.typicode.com/posts");

        // Check if the response is okay
        if (!response.ok) {
            throw new Error(`HTTP error! Status: ${response.status}`);
        }

        // Convert response to JSON
        const data = await response.json();

        // Display data in table
        displayData(data);
    } catch (error) {
        console.error("Error fetching data:", error);
    }
}

function displayData(posts) {
    const tableBody = document.querySelector("#dataTable tbody");
    tableBody.innerHTML = ""; // Clear existing data

    posts.slice(0, 10).forEach(post => { // Show only first 10 posts
        const row = document.createElement("tr");

        row.innerHTML = `
            <td>${post.id}</td>
            <td>${post.title}</td>
            <td>${post.body}</td>
        `;

        tableBody.appendChild(row);
    });
}
```

How It Works

1. **Button Click Event**
    - When the button is clicked, `**fetchData()**` is called.
2. **Fetching Data**
    - `**fetch()**` is used to get data from the API asynchronously.
    - `**await response.json()**` converts the response into JSON format.
3. **Handling Errors**
    - If the API request fails, an error message is logged.

4. **Displaying Data in the Table**
    - We take the first 10 posts and insert them into the table dynamically.

Key Concepts You'll Learn

- **Asynchronous JavaScript** (`async/await`)
- **Fetching API Data** using `fetch()`
- **JSON Parsing** (`response.json()`)
- **DOM Manipulation** (`document.createElement`, `.innerHTML`)
- **Error Handling** (`try...catch`)

How to Run It

1. Save the files (`index.html`, `styles.css`, `script.js`) in the same folder.
2. Open `index.html` in your browser.
3. Click the "Fetch Posts" button.
4. The data should appear dynamically in the table! 🎉

This assignment covers **real-world skills** like fetching data, handling responses, and updating the UI dynamically. Let me know if you need any modifications! 🚀