Day 12: Fetching Data with Axios & Displaying Data from a Backend API

---

1. What is Axios?

- **Axios** is a popular JavaScript library used to make HTTP requests (like GET, POST, PUT, DELETE).
- It works with both the browser and Node.js.
- Similar to `fetch()`, but easier and more powerful (handles JSON automatically, supports interceptors, error handling, etc.).

👉Installation:

```
npm install axios
```

👉Importing in React:

```
import axios from 'axios';
```

---

2. Why Use Axios in React?

- To fetch data from a backend API (like user details, posts, products, etc.).
- To send data to a backend API (form submission, login, etc.).
- To handle **side effects** in React (must use inside `useEffect`).

---

3. Fetching Data with Axios (GET Request)

Example: Fetching a list of users from an API

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

function UserList() {
  const [users, setUsers] = useState([]); // store data from API
  const [loading, setLoading] = useState(true); // for loading state
  const [error, setError] = useState(null); // for errors

  useEffect(() => {
    // Side effect: fetching data from API
    axios.get('https://jsonplaceholder.typicode.com/users')
      .then(response => {
        setUsers(response.data); // set fetched data into state
        setLoading(false);
      })
      .catch(error => {
        setError(error.message); // capture error
        setLoading(false);
      });
  }, []); // empty dependency => runs once when component mounts

  if (loading) return <p>Loading...</p>;
  if (error) return <p>Error: {error}</p>;

  return (
    <div>
      <h2>User List</h2>
      <ul>
        {users.map(user => (
          <li key={user.id}>{user.name} - {user.email}</li>
        ))}
      </ul>
    </div>
  );
}

export default UserList;
```

Breakdown:

- `axios.get(url)` → Makes GET request.
- `response.data` → Contains the actual data.
- `setUsers(response.data)` → Stores the data into state.
- `useEffect` → Ensures request happens after component mounts.
- `loading` & `error` → Provide better user experience.

---

4. Displaying Data

- After fetching, store it in **state** (`useState`).
- Use `.map()` to loop through and display each item.
- Always include `key` when rendering lists in React.

Example of rendering posts:

```
<ul>
  {posts.map(post => (
    <li key={post.id}>{post.title}</li>
  ))}
</ul>
```

5. Common Mistakes & Tips

👈🏼Always use `useEffect` for API calls. 👈🏼Always handle **loading** and **error states**. 👈🏼Use unique `key` for list items. 👈🏼Never put API calls directly in the component body → causes infinite re-renders. 👈🏼Secure your API URLs (don't hardcode sensitive tokens).

---

6. Exercise (15-20 mins)

👉 Build a React component called **PostList**: 1. Use Axios to fetch posts from this API: `https://jsonplaceholder.typicode.com/posts` 2. Show a **loading** state while fetching. 3. Show an **error** if request fails. 4. Display the **title** and **body** of each post in a styled list.

👎🏿Bonus Challenge: - Add a search box to filter posts by title. - Display only the first 10 posts initially, and add a "Load More" button.

👈🏼By completing this, you'll fully understand **fetching and displaying backend data in React using Axios**.