Day 14: API Integration in React

Today, you'll learn how to **fetch data from APIs**, manage asynchronous requests, and display data dynamically in React applications. You'll explore `fetch()`, `axios`, and best practices for API integration.

---

1 Understanding APIs in React

An API (Application Programming Interface) allows you to fetch or send data to/from a server. Common API methods include:
- **GET**: Retrieve data
- **POST**: Send new data
- **PUT/PATCH**: Update existing data
- **DELETE**: Remove data

---

2 Fetching Data using `fetch()`

The `fetch()` function is built into JavaScript and is used for API calls.

Example: Fetching Data from an API

```
import { useEffect, useState } from "react";

function FetchData() {
  const [movies, setMovies] = useState([]);

  useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/posts")
      .then((response) => response.json())
      .then((data) => setMovies(data))
      .catch((error) => console.error("Error fetching data:", error));
  }, []);

  return (
    <div>
      <h2>Movie List</h2>
      <ul>
        {movies.map((movie) => (
          <li key={movie.id}>{movie.title}</li>
        ))}
      </ul>
    </div>
  );
```

```
}

export default FetchData;
```

✅ **Uses `useEffect()` to fetch data on component mount**
✅ **Stores API data in `useState()` and updates UI dynamically**

3️⃣ Fetching Data using `axios` (Alternative to `fetch`)

`axios` is a library that simplifies API requests and **automatically parses JSON responses**. Install it first:

```
npm install axios
```

Example: Fetching Data with `axios`

```
import axios from "axios";
import { useEffect, useState } from "react";

function FetchDataAxios() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    axios.get("https://jsonplaceholder.typicode.com/users")
      .then((response) => setUsers(response.data))
      .catch((error) => console.error("Error fetching data:", error));
  }, []);

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

export default FetchDataAxios;
```

✅ **Easier syntax with `axios.get()`**
✅ **Error handling with `.catch()`**

4️⃣ Handling Loading & Errors

To improve user experience, show a **loading state** while fetching data.

Example: Adding Loading & Error States

```
function FetchWithLoading() {
  const [data, setData] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/todos")
      .then((res) => {
        if (!res.ok) throw new Error("Failed to fetch data");
        return res.json();
      })
      .then((data) => {
        setData(data);
        setLoading(false);
      })
      .catch((error) => {
        setError(error.message);
        setLoading(false);
      });
  }, []);

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

  return (
    <ul>
      {data.map((item) => (
        <li key={item.id}>{item.title}</li>
      ))}
    </ul>
  );
}
```

✅ **Displays "Loading..." before data loads**
✅ **Shows an error message if the API request fails**

5️⃣ Sending Data (POST Request)

You can **send data** to an API using a **POST request**.

Example: Submitting Data to an API

```
function AddUser() {
  const [name, setName] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault();
    fetch("https://jsonplaceholder.typicode.com/users", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ name }),
    })
      .then((res) => res.json())
      .then((data) => console.log("User added:", data))
      .catch((error) => console.error("Error:", error));
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        placeholder="Enter name"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
      <button type="submit">Add User</button>
    </form>
  );
}

export default AddUser;
```

✅ **Sends user input to an API using `fetch()`**
✅ **Logs the response from the server**

---

6 Using `async/await` for Cleaner Code

Instead of `.then()`, you can use **async/await** for better readability.

Example: Fetching Data with `async/await`

```
async function fetchData() {
  try {
    const response = await fetch("https://jsonplaceholder.typicode.com/todos"
);
    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.error("Error fetching data:", error);
```

```
  }
}
fetchData();
```

✅ **No need for `.then()`, making it easier to read**
✅ **Uses `try...catch` for error handling**

7️⃣ Mini Project: Fetch & Display Movies

1. Create a Movie API Component

```jsx
import { useEffect, useState } from "react";
import axios from "axios";

function Movies() {
  const [movies, setMovies] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    axios.get("https://api.tvmaze.com/shows")
      .then((res) => {
        setMovies(res.data.slice(0, 10)); // Show first 10 movies
        setLoading(false);
      })
      .catch((err) => console.error(err));
  }, []);

  if (loading) return <p>Loading movies...</p>;

  return (
    <div>
      <h2>Movie List</h2>
      <ul>
        {movies.map((movie) => (
          <li key={movie.id}>{movie.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default Movies;
```

✅ **Fetches movies from an API**
✅ **Displays movie names in a list**
✅ **Handles loading state**

---

8 Summary of Day 14

- ✔ **Use `fetch()` or `axios` for API requests**
- ✔ **Handle loading and error states**
- ✔ **Send data using `POST` requests**
- ✔ **Use `async/await` for cleaner code**
- ✔ **Built a mini project fetching movies from an API**

🚀 Next Step: Day 15 - Deploying Your React App