📝 **React Report: useEffect for API Calls (using fetch)**

📅 **Date:** August 5, 2025
🧑‍💻 **Author:** Hamza Rafique Awan
📚 **Topic:** React useEffect + fetch API

🔍 Objective

To understand how to perform API calls inside React components using useEffect and the native fetch function. This approach helps load data when the component mounts or when certain dependencies change.

🔧 Tools & Concepts Used

- React (Functional Components)
- useEffect Hook
- fetch API
- useState to store response data

💡 Key Concepts

1. useEffect Hook

useEffect is used to perform side effects like data fetching, timers, subscriptions, etc., in functional components.

2. fetch API

The native browser API used to make HTTP requests. Returns a Promise.

✅ Steps to Perform API Call with useEffect

jsx

CopyEdit

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

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

  useEffect(() => {
    fetch("https://api.example.com/data")
      .then((res) => res.json())
      .then((json) => {
        setData(json);
        setLoading(false);
      })
      .catch((err) => {
        console.error("Error fetching data:", err);
        setLoading(false);
      });
  }, []); // Empty dependency array = run once on mount

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

  return (
    <div>
      <h2>Fetched Data:</h2>
      <pre>{JSON.stringify(data, null, 2)}</pre>
    </div>
  );
```

}

🧠 What I Learned

- useEffect runs **after** the component renders.
- Using an empty dependency array [] ensures the effect only runs once.
- Always handle errors using .catch() or try-catch if using async/await.
- Optional: use async/await inside useEffect via defining an inner async function.

🚀 Next Steps

- Learn about **axios** for easier API calls.
- Explore **loading skeletons** or **error boundaries**.
- Use **custom hooks** to reuse API logic across components.