Internship Report – Frontend Dev

Week 7: React.js Advanced

Name: Zainab
Father Name: Assad Qayyum
Date: 6th Aug, 2025
Internship Domain: Front-end Intern
Task: Context API Basics, useEffect for API Calls (using fetch)

Task Overview: (Day3)

Today's task focused on learning advanced React concepts – specifically the **Context API** for managing global state and using the **useEffect hook** to make API calls and display data. These concepts are essential for building larger, scalable applications where multiple components need access to shared data and dynamic content fetched from servers.

Content Covered:

The content covered is:

- Context API Basics
- useEffect Hook for API Calls(using fetch)

1) Context API Basics

The Context API allows us to share data globally across different components without having to pass props manually at every level. It's like having a shared backpack of data that any component can access directly.

Main Parts of Context API:

a) Create Context

You create a context object using React.createContext().

```
const UserContext = React.createContext();
```

b) Provider

A special component that wraps your app and provides data to child components.

```
<UserContext.Provider value={userData}>
  <App />
</UserContext.Provider>
```

c) Consumer (or useContext Hook)

You Any component inside the provider can access the data directly.

```
const data = useContext(UserContext);
```

Example:

```
import React, { createContext, useContext } from "react";

const ThemeContext = createContext();

function App() {
  return (
    <ThemeContext.Provider value="dark">
      <Toolbar />
    </ThemeContext.Provider>
  );
}

function Toolbar() {
  return <ThemedButton />;
}
function ThemedButton() {
  const theme = useContext(ThemeContext);
  return <button>Current Theme: {theme}</button>;
}
```

Benefits

- Avoids prop drilling.
- Centralized data management.
- Easy to maintain in larger applications.

2) useEffect for API Calls (using fetch)

useEffect is a React hook that lets you run code automatically when a component is mounted or updated.

When fetching data from a server, useEffect ensures the request is made only once when the component loads, preventing repeated calls.

- useEffect lets you run code when a component loads or updates.
- Commonly used to fetch data from a server and display it on your page.
- This prevents infinite loops and makes sure the data is fetched only when needed.

Parts of useEffect:

Callback Function: Code that runs after the component renders.

Dependency Array: Controls when useEffect runs.

Cleanup Function: (optional) to cancel or clean up side effects.

Steps for Fetching Data

1. Use useState to store the fetched data.
2. Use useEffect to fetch data when the component loads
3. Update the state with the fetched data.
4. Render the data in JSX.

Practice Code:

This React app uses the Context API to share country data across components and the useEffect hook to automatically fetch Pakistan's details from an API when the page loads. The page then displays key information (capital, region, population).

App.js

src > App.js > CountryDetails

```jsx
import React, { createContext, useContext, useState, useEffect } from "react";

// 1. Create Context
const CountryContext = createContext();

function App() {
  const [country, setCountry] = useState(null);

  // 2. Fetch Pakistan data using useEffect
  useEffect(() => {
    fetch("https://restcountries.com/v3.1/name/pakistan")
      .then(response => response.json())
      .then(data => setCountry(data[0]));
  }, []);

  const pageStyle = {
    fontFamily: "Arial",
    padding: "30px",
    backgroundColor: "#f2f6ff", // light blue background
    minHeight: "100vh",
  };

  const containerStyle = {
    maxWidth: "700px",
    margin: "0 auto",
    textAlign: "center",
  };

  const headingStyle = {
    color: "#003366",
```

Ln 103, Col 2 Spaces: 2 UTF-8 LF {} JavaScript Go Live

App.js

src > App.js > CountryDetails

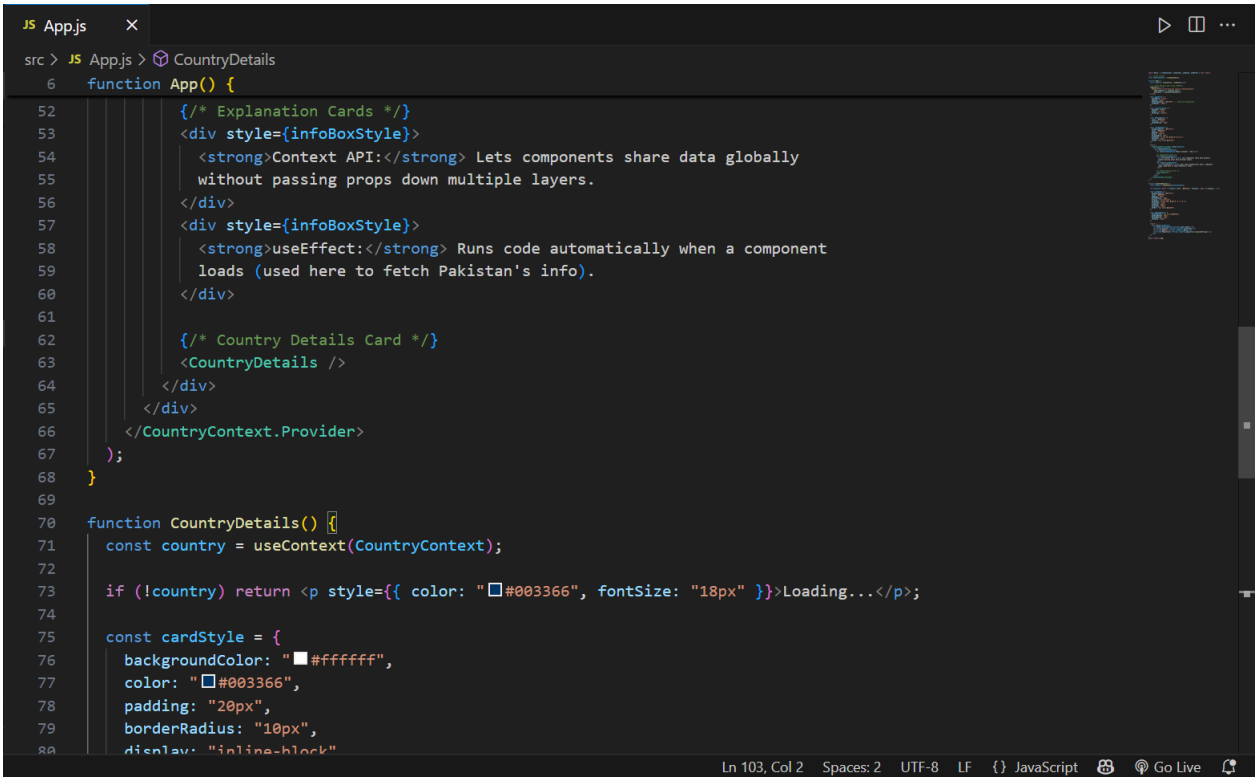
```jsx
function App() {
  const headingStyle = {
    color: "#003366",
    fontSize: "32px",
    marginBottom: "25px",
  };

  const infoBoxStyle = {
    backgroundColor: "#ffffff",
    color: "#003366",
    padding: "15px",
    margin: "12px 0",
    borderRadius: "8px",
    boxShadow: "0 4px 6px rgba(0,0,0,0.1)",
    fontSize: "16px",
    border: "1px solid #d1d9f0",
  };

  return (
    <CountryContext.Provider value={country}>
      <div style={pageStyle}>
        <div style={containerStyle}>
          <h1 style={headingStyle}>React Concepts - Day 3</h1>

          {/* Explanation Cards */}
          <div style={infoBoxStyle}>
            <strong>Context API:</strong> Lets components share data globally
            without passing props down multiple layers.
          </div>
```

Ln 103, Col 2 Spaces: 2 UTF-8 LF {} JavaScript Go Live

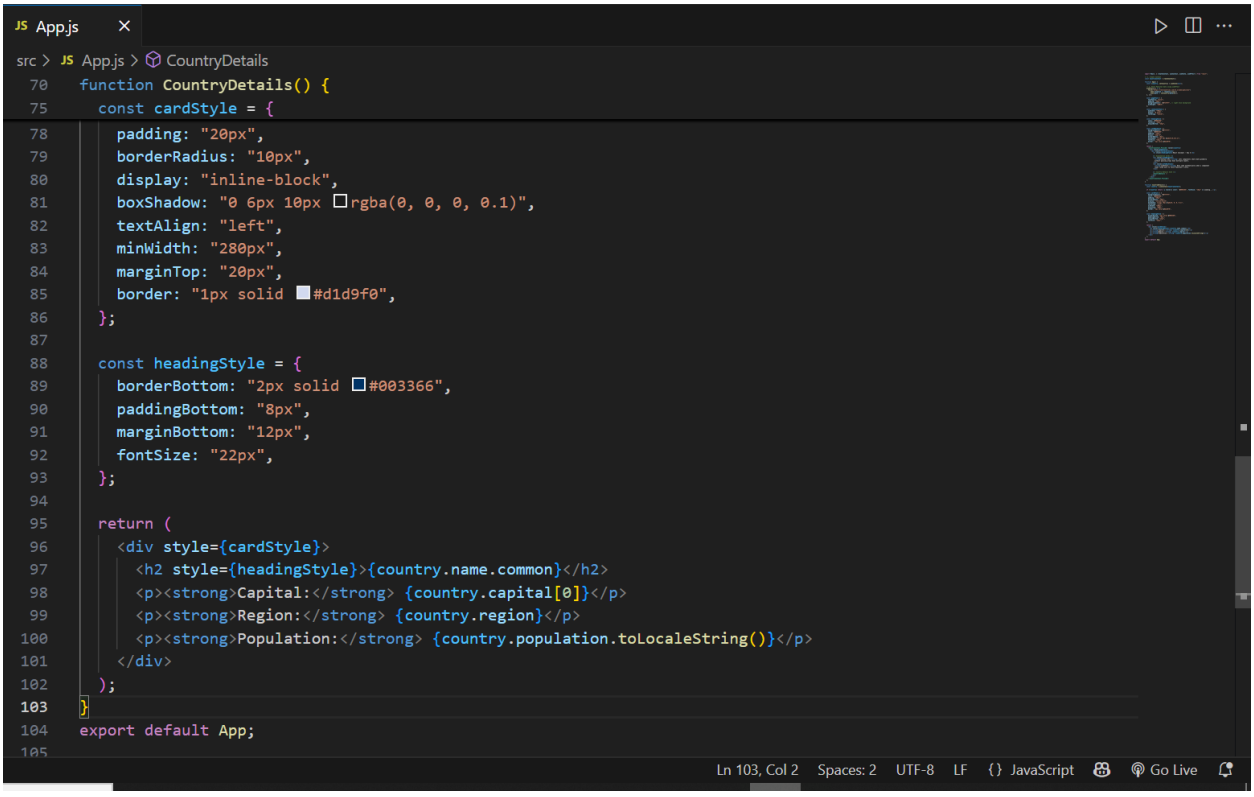

App.js
src > App.js > CountryDetails
function App() {
{/* Explanation Cards */}

Context API: Lets components share data globally without passing props down multiple layers.

useEffect: Runs code automatically when a component loads (used here to fetch Pakistan's info).

{/* Country Details Card */}
<CountryDetails />

</CountryContext.Provider>
);
}
function CountryDetails() {
const country = useContext(CountryContext);
if (!country) return <p style={{ color: "#003366", fontSize: "18px" }}>Loading... ;
const cardStyle = {
backgroundColor: "#ffffff",
color: "#003366",
padding: "20px",
borderRadius: "10px",
display: "inline-block"
Ln 103, Col 2 Spaces: 2 UTF-8 LF {} JavaScript Go Live


App.js
src > App.js > CountryDetails
function CountryDetails() {
const cardStyle = {
padding: "20px",
borderRadius: "10px",
display: "inline-block",
boxShadow: "0 6px 10px rgba(0, 0, 0, 0.1)",
textAlign: "left",
minWidth: "280px",
marginTop: "20px",
border: "1px solid #d1d9f0",
};
const headingStyle = {
borderBottom: "2px solid #003366",
paddingBottom: "8px",
marginBottom: "12px",
fontSize: "22px",
};
return (

<h2 style={headingStyle}>{country.name.common}</h2>
Capital: {country.capital[0]}
Region: {country.region}
Population: {country.population.toLocaleString()}

);
}
export default App;
Ln 103, Col 2 Spaces: 2 UTF-8 LF {} JavaScript Go Live

Conclusion:

Today, I learned how to manage global state in React using the **Context API** and how to fetch and display data using the **useEffect hook**. These concepts help in creating scalable applications where data is shared efficiently and loaded dynamically from servers, making the app more powerful and user-friendly.