Chapter 12: Context API

1 Introduction

Is chapter mein humne **Context API** ke baare mein seekha, jo React mein global state manage karta hai. Yeh prop drilling se bachata hai aur data ko multiple components mein share karta hai.

2 12.1 Why Context API?

Jab multiple components ko same data chahiye (jaise theme, user info), toh Context API global store banata hai.

2.1 Real-World Example

Food delivery app mein theme (dark/light) ya user data (name, email) ko globally share karna, jaise Header, Menu, Cart components mein.

3 12.2 Creating and Using Context

Context API ke teen parts:

- `createContext`: Context object banata hai.
- `Provider`: Data provide karta hai.
- `useContext`: Data access karta hai.

3.1 Code Example: Theme Toggle with Context

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Theme Context</title>
  <script
     src="https://cdn.jsdelivr.net/npm/react@18.2.0/umd/react.development.js
  <script
     src="https://cdn.jsdelivr.net/npm/react-dom@18.2.0/umd/react-dom.develo
  <script
     src="https://cdn.jsdelivr.net/npm/@babel/standalone@7.20.6/babel.min.js
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel">
    // Creating Context
    const ThemeContext = React.createContext();

    // Header Component
    function Header() {
```

```
  const { theme, toggleTheme } = React.useContext(ThemeContext);
  return (
    <div className={`p-4 ${theme === 'dark' ? 'bg-gray-800 text-white' : 'bg-gray-100 text-black'}`}>
      <h1 className="text-2xl font-bold">Food App</h1>
      <button
        className="bg-blue-500 text-white px-4 py-2 mt-2 rounded"
        onClick={toggleTheme}
      >
        Toggle Theme ({theme === 'dark' ? 'Light' : 'Dark'})
      </button>
    </div>
  );
}

// Menu Component
function Menu() {
  const { theme } = React.useContext(ThemeContext);
  return (
    <div className={`p-4 ${theme === 'dark' ? 'bg-gray-700 text-white' : 'bg-white text-black'}`}>
      <h2 className="text-xl font-bold">Menu</h2>
      <ul className="list-disc list-inside">
        <li>Chai - Rs 10</li>
        <li>Samosa - Rs 20</li>
      </ul>
    </div>
  );
}

// Main App Component
function App() {
  const [theme, setTheme] = React.useState('light');

  const toggleTheme = () => {
    setTheme(theme === 'light' ? 'dark' : 'light');
  };

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      <div className="min-h-screen">
        <Header />
        <Menu />
      </div>
    </ThemeContext.Provider>
  );
}

```

```
    const root = ReactDOM.createRoot(document.getElementById('root'));
    root.render(<App />);
  </script>
</body>
</html>
```

3.2 Output

Browser mein yeh dikhega:

- **Light Mode**:
  - **Header**: Gray background (`bg-gray-100`), black text, heading "Food App", blue button "Toggle Theme (Dark)".
  - **Menu**: White background, black text, heading "Menu", list: "Chai - Rs 10", "Samosa - Rs 20".
- **Dark Mode** (button click pe):
  - **Header**: Dark gray (`bg-gray-800`), white text, button "Toggle Theme (Light)".
  - **Menu**: Darker gray (`bg-gray-700`), white text, same list.

3.3 Explanation

- `ThemeContext`: Context object.
- `Provider`: Theme state aur `toggleTheme` provide.
- `useContext`: Theme data access in `Header`, `Menu`.
- No prop drilling.

4 12.3 Combining with Routing and API Calls

Context API ko React Router aur API calls ke saath combine karte hain.

4.1 Code Example: User Context with Routing

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>User Context with Routing</title>
  <script src="https://cdn.jsdelivr.net/npm/react@18.2.0/umd/react.development.js
  <script src="https://cdn.jsdelivr.net/npm/react-dom@18.2.0/umd/react-dom.develo
  <script src="https://cdn.jsdelivr.net/npm/@babel/standalone@7.20.6/babel.min.js
  <script src="https://cdn.jsdelivr.net/npm/react-router-dom@6.3.0/dist/umd/react-
  <script src="https://cdn.jsdelivr.net/npm/axios@1.4.0/dist/axios.min.js"></scri
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <div id="root"></div>
  <script type="text/babel">
    const { BrowserRouter, Routes, Route, Link } = ReactRouterDOM;

    // Creating User Context
    const UserContext = React.createContext();

    // Header Component
    function Header() {
      const { user } = React.useContext(UserContext);
      return (
        <div className="p-4 bg-gray-100">
          <h1 className="text-2xl font-bold text-blue-600">Food App</h1>
          <p className="text-lg">Welcome, {user ? user.name : 'Guest'}!</p>
          <nav className="flex justify-center space-x-4 mt-2">
            <Link to="/" className="text-blue-500 hover:underline">Home</Link>
            <Link to="/profile" className="text-blue-500 hover:underline">Profile</Link>
          </nav>
        </div>
      );
    }

    // Profile Component
    function Profile() {
      const { user } = React.useContext(UserContext);
      return (
        <div className="text-center p-4">
          <h2 className="text-xl font-bold text-blue-600">User Profile</h2>
          {user ? (
            <div>
              <p className="text-lg">Name: {user.name}</p>
              <p className="text-lg">Email: {user.email}</p>
            </div>
          ) : (
            <p className="text-lg">No user data</p>
          )}
        </div>
      );
    }

    // Main App Component
    function App() {
      const [user, setUser] = React.useState(null);
      const [loading, setLoading] = React.useState(true);

      React.useEffect(() => {
        axios.get('https://jsonplaceholder.typicode.com/users/1')
          .then((response) => {
            setUser(response.data);
            setLoading(false);
          })
          .catch((err) => {
            console.error(err);
            setLoading(false);
          });
      }, []);

      if (loading) return <div className="text-center p-4">Loading...</div>;

      return (
        <UserContext.Provider value={{ user }}>
          <BrowserRouter>
            <Header />
            <Routes>
              <Route path="/" element={<h1 className="text-center text-3xl font-bold text-blue-600 p-4">Home</h1>} />
              <Route path="/profile" element={<Profile />} />
            </Routes>
          </BrowserRouter>
        </UserContext.Provider>
      );
    }

    const root = ReactDOM.createRoot(document.getElementById('root'));
    root.render(<App />);
  </script>
</body>
</html>
```

4.2 Output

Browser mein yeh dikhega:

- **Initially**: "Loading..." (center-aligned).
- **After Fetch**:
  - **Header**: Gray background, heading "Food App", text "Welcome, Leanne Graham!" (or similar), links "Home", "Profile" (blue, hover pe underline).

  - **URL:** `/`: Heading "Home" (large, bold, blue).
  - **URL: /profile**: Heading "User Profile", text "Name: Leanne Graham", "Email: Sincere@april.biz".

4.3 Explanation

- `UserContext`: User data store.
- `Provider`: API se user data provide.
- `useContext`: User data access in `Header`, `Profile`.
- `axios`: User data fetch.
- Routing: `/`, `/profile`.

5 Common Mistakes

- **Missing `Provider`**: Context data ke liye zaroori.
- **Overusing Context**: Local state ke liye `useState` use karo.
- **No Loading States**: API-based Context mein loading handle karo.

6 Interview Tips

- **Context API kya hai?**
  - Global state manage, prop drilling avoid.
- **Kab use karte hain?**
  - Multiple components ko same data chahiye.
- **`useContext` vs `useState`?**
  - `useContext`: Global, `useState`: Local.

7 Assignment: Practice Time

7.1 Task 1: Theme Switcher

- `ThemeApp` banao jisme `ThemeContext`.
- Dark/light theme toggle, `Header`, `Menu` mein apply.
- Tailwind se style.

7.2 Task 2: Cart Context

- `CartApp` banao jisme `CartContext`.
- Cart items globally manage (add/remove).
- `CartList`, `CartTotal` mein data use.

- Tailwind se style.

7.3 Task 3: User Context with API and Routing

- `UserApp` banao jisme `UserContext`.
- JSONPlaceholder se user data (`/users/1`).
- `/home`, `/profile` routes, user data globally.
- Tailwind se style.