Day 9: React Context API (State Management)

Today, you'll learn about **React Context API**, which helps manage **global state** without prop drilling. This is useful for managing **user authentication, themes, language preferences, and other shared states**.

---

1 What is the Context API?

Context API allows you to:

✅ **Share state** between multiple components without passing props manually
✅ Avoid **prop drilling** (passing data down through multiple levels)
✅ Manage **global state** (like authentication, themes, or user settings)

---

2 When to Use Context API?

Use it when:

✔ You need to share data **across multiple components**
✔ Passing props **becomes complex and repetitive**
✔ You want **centralized state management** without external libraries like Redux

---

3 How to Use Context API?

Step 1: Create Context

Create a file `ThemeContext.js`:

```
import { createContext } from "react";

export const ThemeContext = createContext(null);
```

✅ **`createContext()` creates a new context**

Step 2: Create a Context Provider

Wrap your app with a **Provider** to supply data.

```
import { useState } from "react";
import { ThemeContext } from "./ThemeContext";

function ThemeProvider({ children }) {
  const [theme, setTheme] = useState("light");

  return (
    <ThemeContext.Provider value={{ theme, setTheme }}>
      {children}
    </ThemeContext.Provider>
  );
}

export default ThemeProvider;
```

✅ **Stores the theme in `useState`**
✅ **Provides `theme` and `setTheme` to all children components**

Step 3: Wrap the App with the Provider

Modify `index.js` or `App.js`:

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";
import ThemeProvider from "./ThemeProvider";

ReactDOM.createRoot(document.getElementById("root")).render(
  <ThemeProvider>
    <App />
  </ThemeProvider>
);
```

✅ **Ensures all components inside `App` can access the theme context**

Step 4: Use Context in Components

Now, any component can access the theme without prop drilling.

```
import { useContext } from "react";
import { ThemeContext } from "./ThemeContext";

function ThemeSwitcher() {
  const { theme, setTheme } = useContext(ThemeContext);

  return (
    <div>
      <p>Current Theme: {theme}</p>
      <button onClick={() => setTheme(theme === "light" ? "dark" : "light")}>
        Toggle Theme
      </button>
    </div>
  );
}

export default ThemeSwitcher;
```

✅ **Uses `useContext(ThemeContext)` to access values**
✅ **Toggles between `light` and `dark` mode**

Using Context for Authentication

Context API is commonly used for **user authentication**.

Example: Auth Context

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

export const AuthContext = createContext(null);

function AuthProvider({ children }) {
  const [user, setUser] = useState(null);

  return (
    <AuthContext.Provider value={{ user, setUser }}>
      {children}
    </AuthContext.Provider>
  );
}

export default AuthProvider;
```

Example: Login Component

```jsx
import { useContext } from "react";
import { AuthContext } from "./AuthContext";

function Login() {
  const { user, setUser } = useContext(AuthContext);

  return (
    <div>
      {user ? (
        <p>Welcome, {user}!</p>
      ) : (
        <button onClick={() => setUser("Danish")}>Login</button>
      )}
    </div>
  );
}

export default Login;
```

✅ **Stores user login state in context**
✅ **Avoids prop drilling by using `useContext(AuthContext)`**

5️⃣ Summary of Day 9

✔ **Context API** helps manage global state
✔ **`createContext()`** creates a context
✔ **`Provider` wraps components and provides data**
✔ **`useContext()`** accesses context values
✔ Used Context API for **theme switching & authentication**

🚀 Next Step: Day 10 - React Hooks (useEffect & Lifecycle Methods)