REACT NOTES (10)

Context API in reactjs

## What is Prop Drilling?

*Prop drilling occurs when a parent component generates its state and passes it down as props to its children components that do not consume the props – instead, they only pass it down to another component that finally consumes it.*

*And for solving prop drilling problem, there is concept of* ***Context API***

* ***Context API*** *is purely associated with react. It doesn’t exist outside react.*
* *But, similar problem like prop drilling exists outside react too (i.e in vue.js)*
* *And to solve this problem outside react, there is similar concept like context api*

*Named as* ***Redux.***

* ***Redux is a standalone library.***
* ***Redux*** *handles transferring of data in an organized way through state management.*
* *And, in react, it is used with the name* ***react-Redux.***
* ***In market, easy version of redux is available named as Redux-toolkit (RTK)***
* *There are also other similar libraries for state management like redux**(i.e.* ***zustand****, etc)*
* ***What is React Context?***

*React Context is a method to pass props from parent to child component(s), by*

*storing the props in a store(similar in Redux) and using these props from the store*

*by child component(s) without actually passing them manually at each level of the*

*component tree*

* **Context API**

*CONTEXT IS NOTHING BUT A GLOBAL VARIABLE.*

***STEPS:***

1. *Make a folder named* ***context*** *in src. Then, make file as per your use (i.e. UserContext.js) . Similarly, we can make context for login, product, cart.*

***.js extension is used instead of .jsx because pure javascript used in it.***

1. *Then, in file UserContext.js , make a context using method createContext and store it in a variable and throw it in output.*

***Import React from ’react’***

***Const UserContext = React.createContext()***

***export default UserContext;***

1. ***Context gives a provider.***

***As it provides a variable , hence each context is a provider.***

1. *In the end,* ***we will use it as a wrapper. And, the components that we put inside it will have access of the global user context*** *defined in context file.*

*Example: <UserContext>*

*<Login/>*

*<Card>*

*<Data/>*

*</Card>*

*</UserContext>*

1. *In next step, we also have to make a provider. For that make a file named* ***UserContextProvider.jsx***

*(.jsx because we will use it a s wrapper react component)*

***import React from ‘react’;***

***import UserContext from ‘./UserContext’;***

***const UserContextProvider = ({children}) => {***

***const [user, setUser] = React.useState(null)***

***return(***

***<UserContext.Provider>***

***{children}***

***</UserContext.Provider>***

***)***

***}***

1. ***Now, we will make 2 files ( components ) ,***

***First one to see how to send data => Login.jsx***

***Second one to see how to receive data => Profile.jsx***

***Login.jsx***

*Import React from, {useState, useContext} from ‘react’*

*Import UserContext from ‘../context/UserContext’*

*function Login(){*

*const [username, setUsername] = useState(‘’)*

*const [password, setPassword] = useState(‘’)*

***const {setUser} = useContext(UserContext)***

*const handleSubmit = (e) =>{*

***e.preventDefault()***

***setUser({username, password })***

*// example of context k through data bhjna kse h*

*}*

*return(*

*<div>*

*<h2>Login</h2>*

*<input type=’text’ value={username} onChange={(e)=> setUsername(e.target.value)}*

*placeholder=’username’/>*

*<input type=’text value={password} onChange={(e)=> setPassword(e.target.value}*

*placeholder=’password’/>*

*<button onClick={handleSubmit}> Submit </button>*

*</div>*

*)*

*}*

*export default Login*

***Now we will make another component, (example of how to receive data through context)***

*import React , {useContext} from ‘react’*

*import UserContext from ‘../context/UserContext’*

*function Profile() {*

***const {user} = useContext(UserContext)***

*if (!user) return <div>please login </div>*

*return <div> Welcome {user.username} <div>*

*}*

*export default Profile*

1. ***Now, add components inside wrapper component (i.e. Context Provider)***

*App.jsx*

***<UserContextProvider>***

***<Login/>***

***<Profile/>***

***</UserContextProvider>***

# NOW WE WILL DO ONE MORE PROJECT TO UNDERSTAND CONTEXT API AND ITS WORKING DEEPLY

***STEPS:***

1. *Make a folder named* ***contexts*** *in src. Then, make file as per your use (i.e.theme.js) .Here, we used .js extension because we’re not gonna return context.*
2. *Then, in file theme.js , make a context using method createContext and store it in a variable and throw it in output. We will also put default values in createContext method this time.*

***In this method, we can pass variables and methods both as default values(args) .***

***// Import React from ’react’***

***export const ThemeContext = React.createContext({***

***themeMode: “light”,***

***darkTheme: () =>{},***

***lightTheme: () =>{}***

***})***

***export const ThemeProvider = ThemeContext.provider;***

1. *Here, for provider we won’t make a file named* ***ThemetProvider.jsx.***

*(.jsx because we will use it a s wrapper react component)*

**Because there is no necessity to make it by making a diff component (file),**

**there are diff styles to make provider. And this is another way to make provider.**

***// Import React from ’react’***

***export const ThemeContext = React.createContext({***

***themeMode: “light”,***

***darkTheme: () =>{},***

***lightTheme: () =>{}***

***})***

***export const ThemeProvider = ThemeContext.provider;***

1. **We can also make and export custom hooks here,**

**As we have to**

***// Import React from ’react’***

***export const ThemeContext = React.createContext({***

***themeMode: “light”,***

***darkTheme: () =>{},***

***lightTheme: () =>{}***

***})***

***export const ThemeProvider = ThemeContext.provider;***

***export default function useTheme(){***

***return useContext(ThemeContext)***

***}***

***}***

**Will complete when needed.**

**=> Important points**

**1. localstorage.getItem(key) - returns value in string format and we need that in json so we always convert string into json by using JSON.parse**

**Eg: const todos = JSON.parse(localstorage.getItem(key)**

**Localstorage.setItem(key,value) - we have to set item values in string format so we have to convert our response or array in string by using JSON.stringify**

**Eg: localStorage.setItem("todos",JSON.stringify(todos))**

* **SUMMARY (FOR INTERVIEWS)**

**### What is React Context API?**

\*\*React Context API\*\* is a way to manage state and pass data through your React app without manually passing props down through each level of the component tree.

**### How Does it Work?**

**1. \*\*Create Context**:\*\* First, you define a context using `React.createContext()`. This sets up a space where data can be stored and accessed by any component in your app.

**2. \*\*Provide Context:\*\*** You wrap parts of your app that need access to this data with a `<Context.Provider>` component. This "provides" the context data and functions to those parts of the app.

**3. \*\*Consume Context:\*\*** Components that need the context data can access it using the `useContext` hook. This hook allows components to tap into the context and use the data without needing to pass props down manually.

**### Why Use Context API?**

Context API is useful for managing "global" state that many components need to access. It simplifies passing data between components and reduces the need for prop drilling, making your code cleaner and more maintainable.

**### Example:**

Imagine you have a to-do list app. Instead of passing the list and functions to add or delete to-dos through each component, you can use Context API to make this data available throughout the app.

**### Interview Response:**

"I'd describe React's Context API as a way to manage and share state across your app without prop drilling. You create a context to hold your data, wrap parts of your app with a Provider to make that data available, and then consume it in components that need it using the useContext hook. It's a handy tool for managing global state and simplifying data flow in React applications.