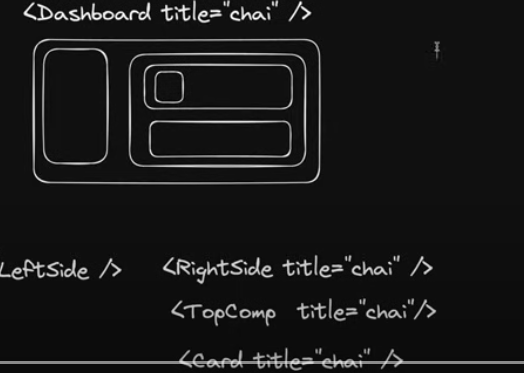
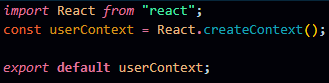
**Context API**

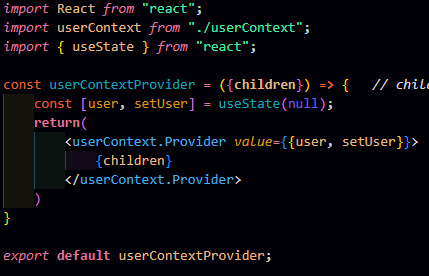
* Context API in react is used to manage the state and share data across the components without need to pass the props manually through every level of the component level.



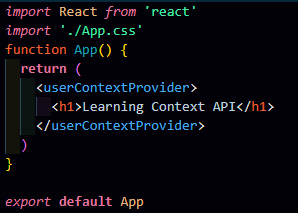
* Only the card component need the data but due to this we have to pass the data to all the levels like right side component and then top component to reach the card component.
* To fix this we use Context API.
* It is used when you have a global data and many components need this data.
* **Steps to use Context:**
  + First we will create a user context for which create a user context **js** file in context named folder in the src.



* + After creating context we have to make a provider to provide the access to the different components.
  + To make a provider create user context provider **jsx** file in the same folder.



* This is how we provide the context.
* After calling the user context provider in **app.jsx** we will now have the access of all the data.

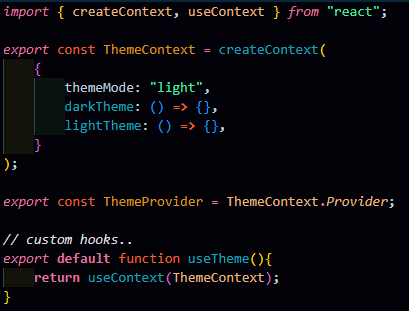


* Now we will create 2 components: **Login.jsx** & **Profile.jsx** to see how the context work.
* In **Login.jsx** we have 2 input boxes for username and password and in **Profile.jsx** we are checking if there is no user then show please login otherwise show their username.

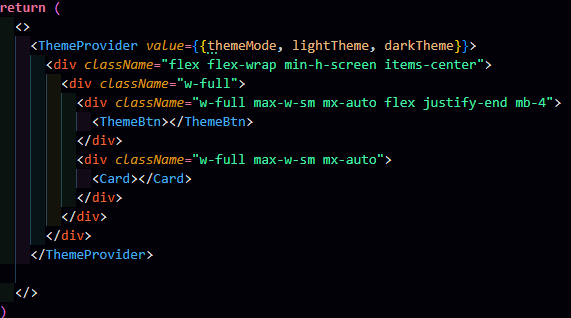
**Theme Switcher Project**

In this project we will see the different approach for making the context.

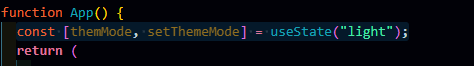
1. Created a context folder with **theme.js** file.
2. In which we will import the create context and use context.
3. Then we will create a theme context name variable in which we will provide some values by default.
   1. **In previous project we haven’t provided any default value to the context while creating it.**
4. We have created an object with theme mode, a function for dark theme and another function for light theme in the theme context function.
5. While creating context we can pass anything like variable or methods.
6. **Earlier we have created another file for context provider but in this** we will directly provide the context in the same file.



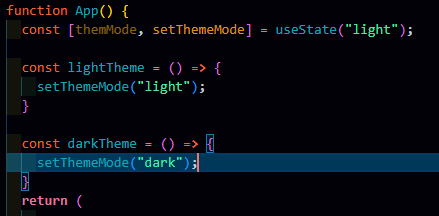
1. Now, in the **App.jsx** file we will wrap our whole app UI with theme provider.
   1. We did this to provide the access of theme context to all the components, it is necessary to provide the access of theme provider to use theme context values such as theme mode, dark theme, light theme.



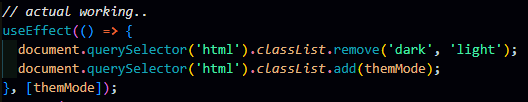
1. Now, we will create a state of theme mode and set theme mode to keep the track of theme mode across the component.



1. Now, we will design the functionality of the value provided in the Them Provider to use them.

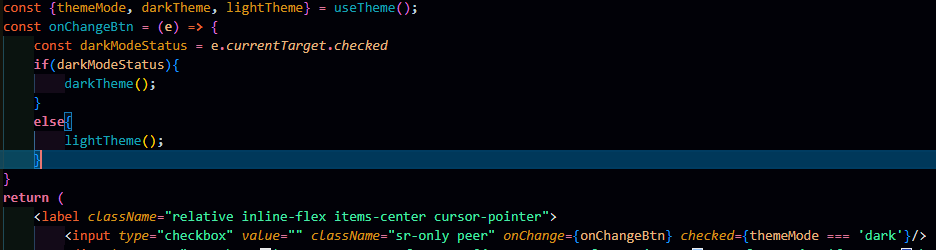


1. After designing the functionality, now we will write code for actual working of the theme switching.



* 1. Here we have used the use effect hook because, use effect hook used when we have to update the DOM or we have to fetch something and in this case we are updating the theme of the DOM.
  2. Firstly, we have removed the light or dark word from the html, because initially we don’t know what the theme of the application is.
  3. Finally, we have add the theme mode state.
  4. And we have one dependency of theme mode, because all the things depends on the change of this only.

1. **Now, everything is working fine, when we inspect or application there is “light” and “dark” but our application does get the access to change,** we are changing manually from inspect option.
2. To fix this we will import the necessary things form use theme custom hook we have created. **We are importing this use theme custom hook to access the theme context and this will allows the them button to interact with the theme context such as toggling theme**
3. Now, we will define the functionality of the theme button. We will add a method on change in which we will have a function and we use checked in built method to **change the theme to dark mode**.



1. **To enable theme toggling we have add dark mode class in tailwind configuration file.**

