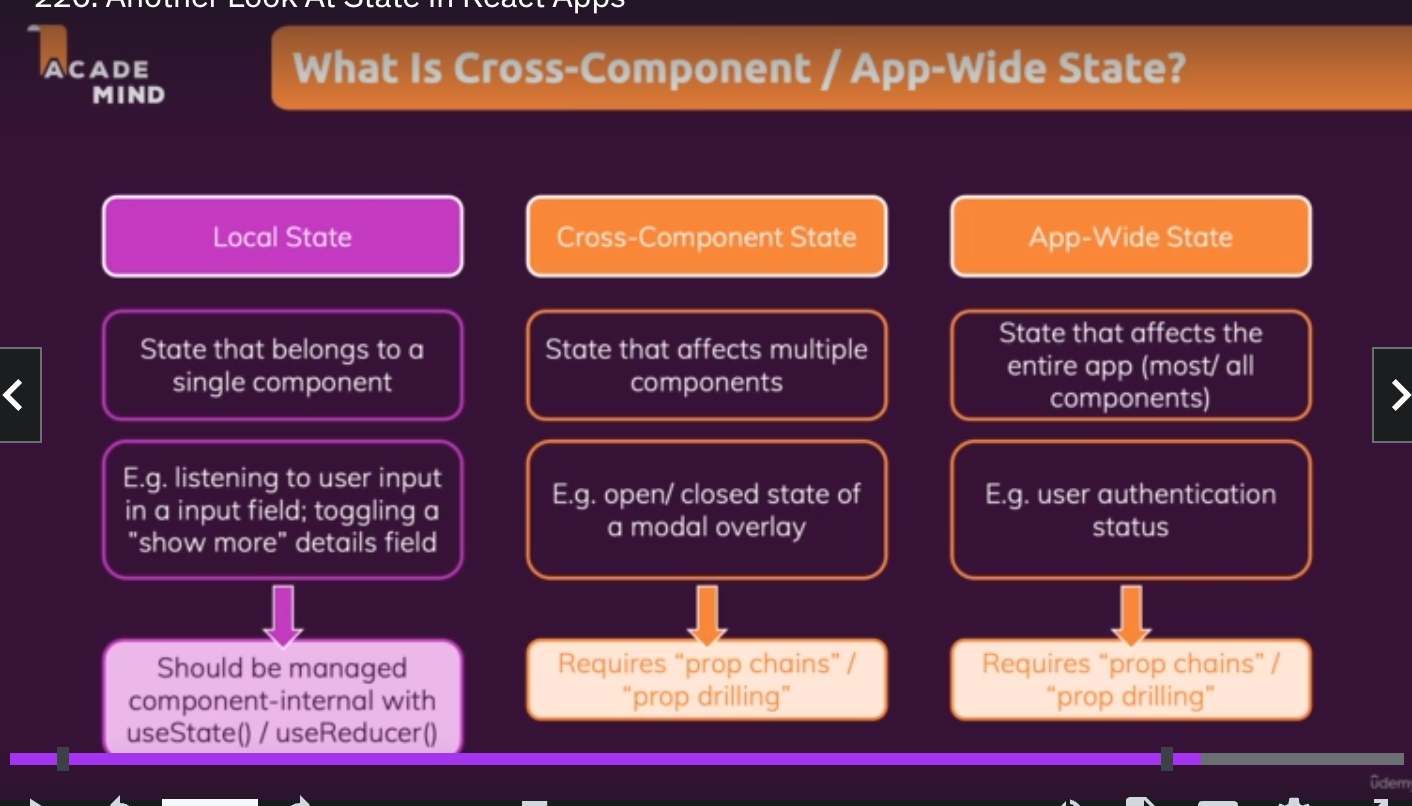
What is redux and why we use it and redux toolkit for handling appwide state.

Redux is a state management system for cross-component or app-wide state.

Different state can be handled as shown below in the picture.



Redux is a state management system for app wide state. Redux has potential disadvantages.

React context has complex setup and is difficult to manage.

We can end up with deeply nested JSX code as shown below.



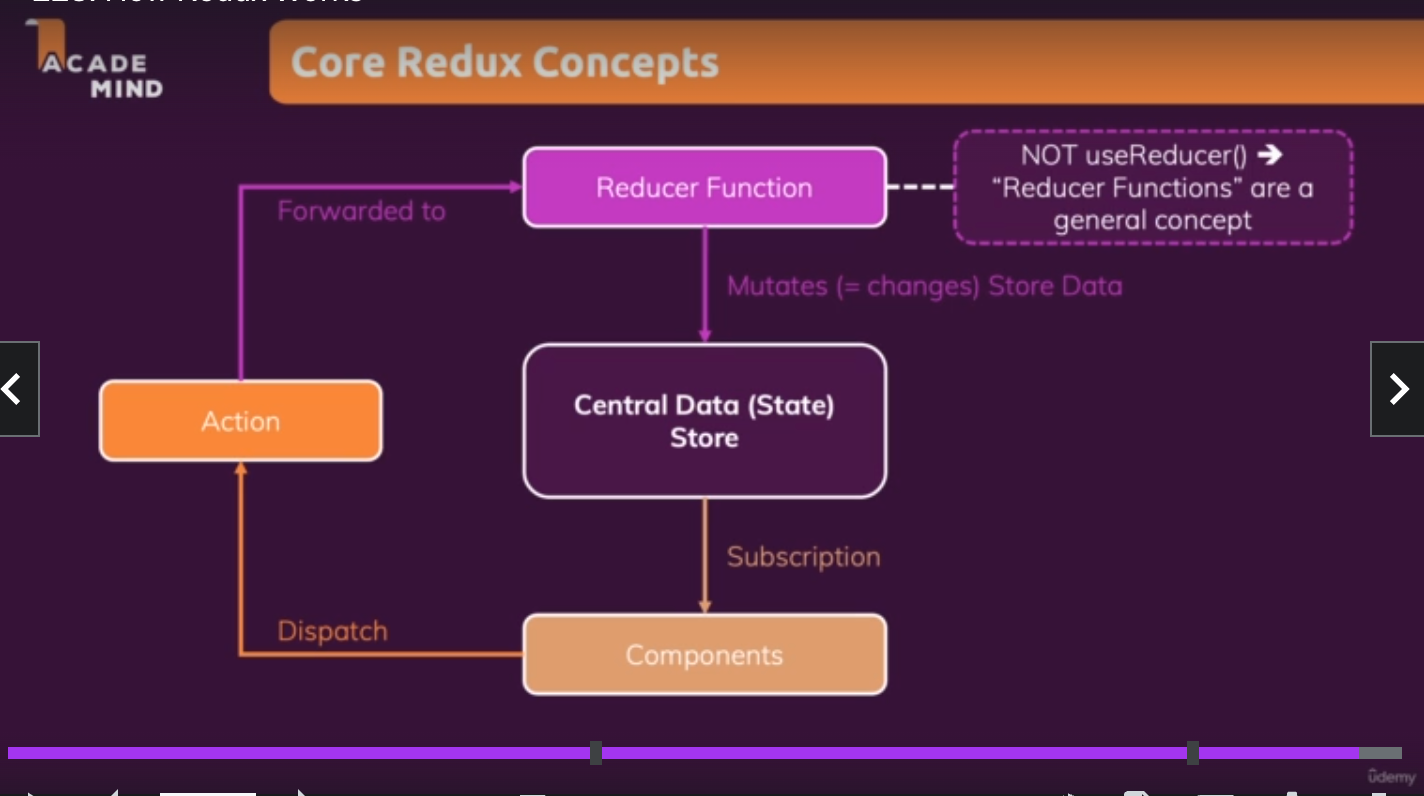
React context is not good for high frequency changes and not really a great substitute.

We will have a potential performance issues.

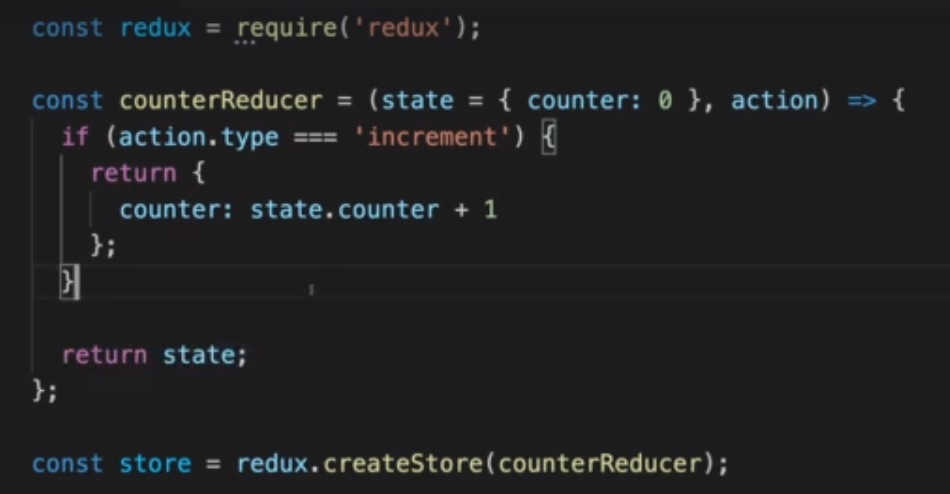
Redux is an alternative to react context and why we might want to use it to our applications.

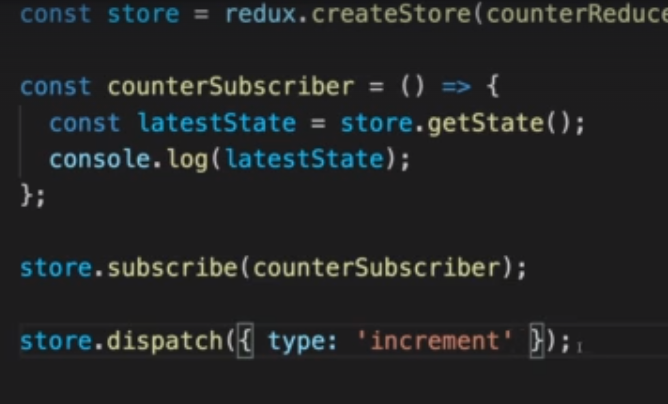
Entire appwide state is stored in central Data (State) store. Components are subscribing to the central Data (state). Components never directly alter the state of the items present in store.

Reducers are responsible for making such changes.



Exploring core redux basics.





The code is the heart of redux.

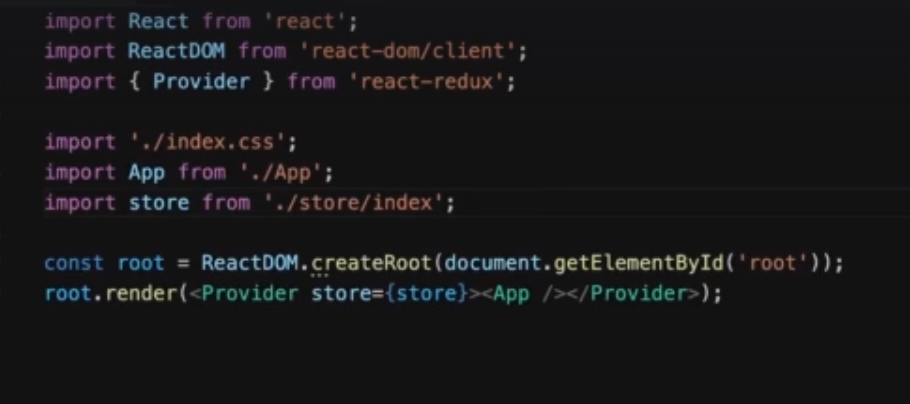
1. Create a store.
2. Create reducer functions and point them inside the store.
3. Create subscriber function and use store to subscribe the function.
4. Call store.dispatch functions to trigger state changes via reducer functions.

Steps to use redux in our project ➖

1. Install redux -> npm install redux
2. Npm install react-redux
3. Create a folder called store and create a new index.js file
4. Then createStore write the code as below.



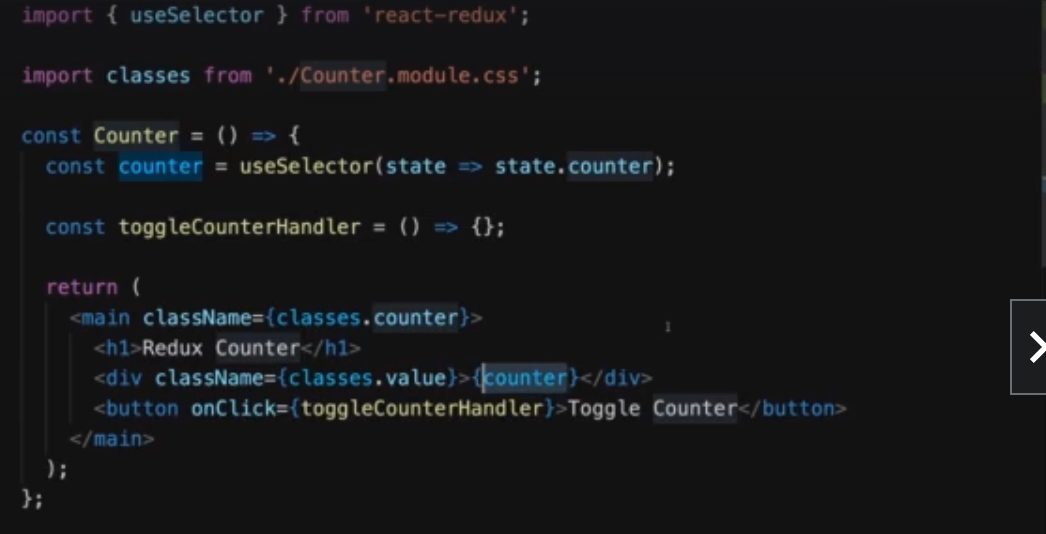
We now need to provide our store since we can have only one store.



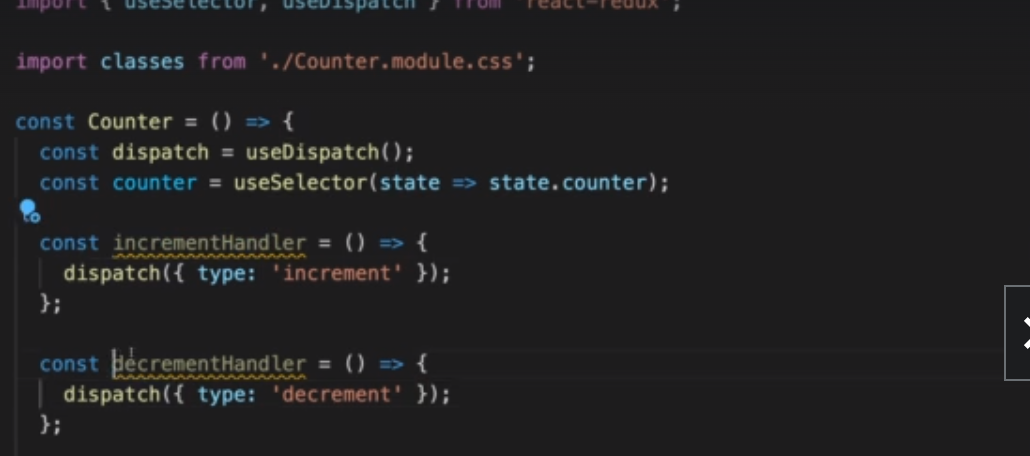
Add provider to the main app for subscription and provide store = {store} prop to the provider.

Now use this store for dispatching action and subscribing to the store.

To subscribe to the state present in the store we use useSelector hook as shown below.

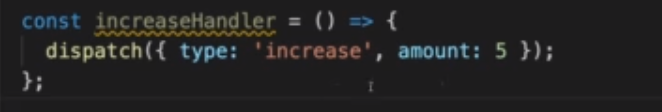


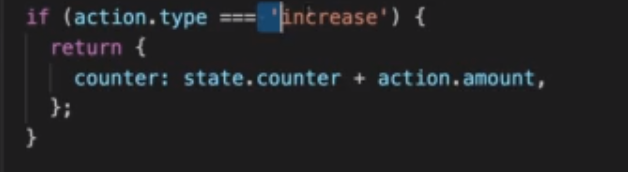
Now we need to dispatch the action.



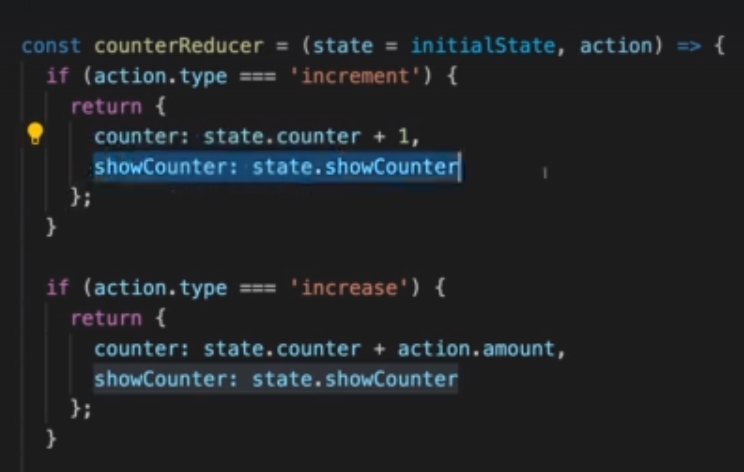
Now we can use useDispatch() hook to dispatch the action from the components to change the states. As shown in the above code.

We can also add a payload to the action function as shown to the left.





We can handle multiple state in the same object as shown below:-



We must always update the other states in reducer function because we overwrite the states. Never mutate existing state always overwrite the state.

React redux has some disadvantages. Reducer function action type names should be unique and we cannot afford typos while working with them. If the state that needs to be maintained is a very large object then , it becomes quite repetitive to use the same object again and again increases the size of reducer function.

We also has to respect state immutability. For all those problems we use constants file to fix the issue. Instead there is another library. If we use redux toolKit it get easier to work with.

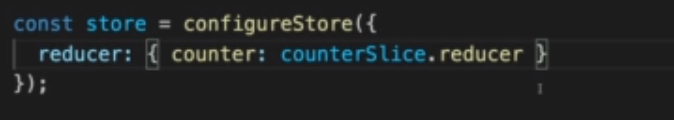
Important redux tool kit library which should be used for easy of working with react.

Installing redux toolkit :-

1. Npm install @reduxjs/toolkit
2. In Store file use import createSlice from @reduxjs/toolkit
3. CreateSlice helps us in creating a slice of a global state.
4. With createSlice we are allowed to mutate the state of the slice.(Because this is automatically done by the redux toolkit.
5. Continue with provider code used in the above example and add store to the main App component.
6. We can access dispatch by using counterSlice.actions as shown below.
7. Then import counterActions inside a component where we want to access counterActions.

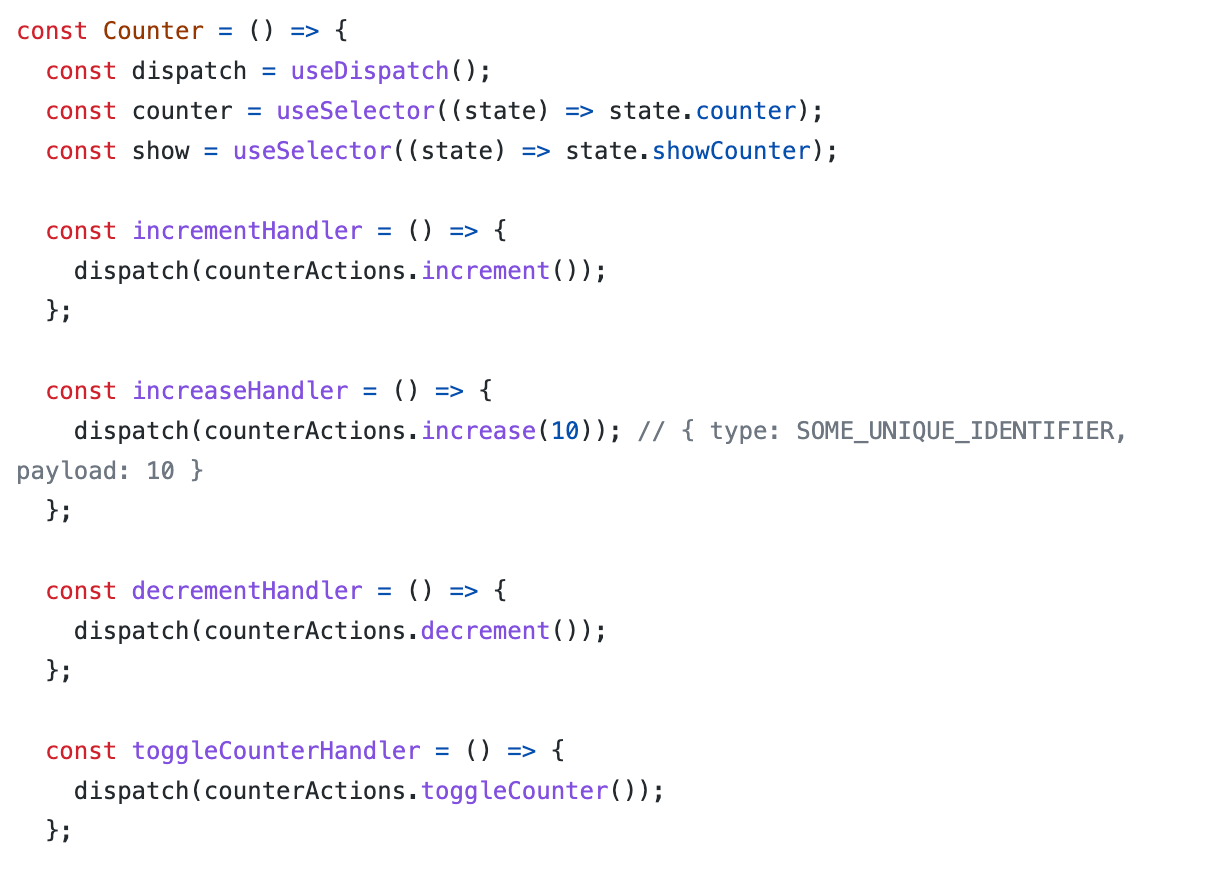


We can import configureStore from redux toolkit for handling multiple slices. It makes merging of multiple stores into one easier.(Because there would be only one store for the entire application).



Incase if we have multiple stores we can map it as shown above.

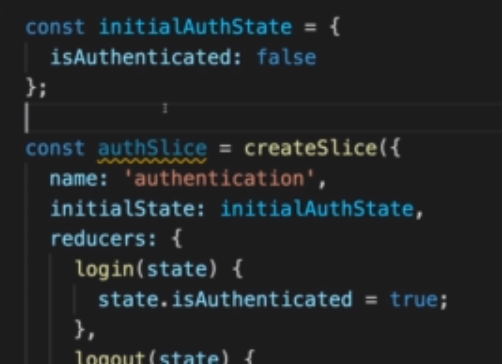
For dispatching actions createSlice automatically creates unique identifiers for us so we dont have to assign any new values to it.

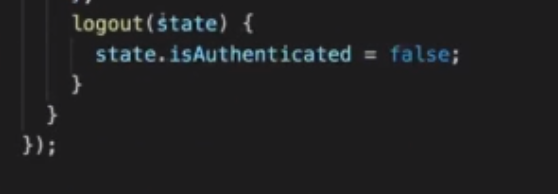


Whenever we want to pass payload, that is already by default taken care of by redux toolkit by convert the passed parameter into the commented statement shown above.

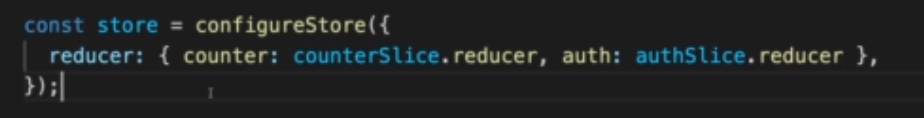
Working with multiple slices.

Let us take an example of adding another slice say authentication slice that needs to be added in index.js





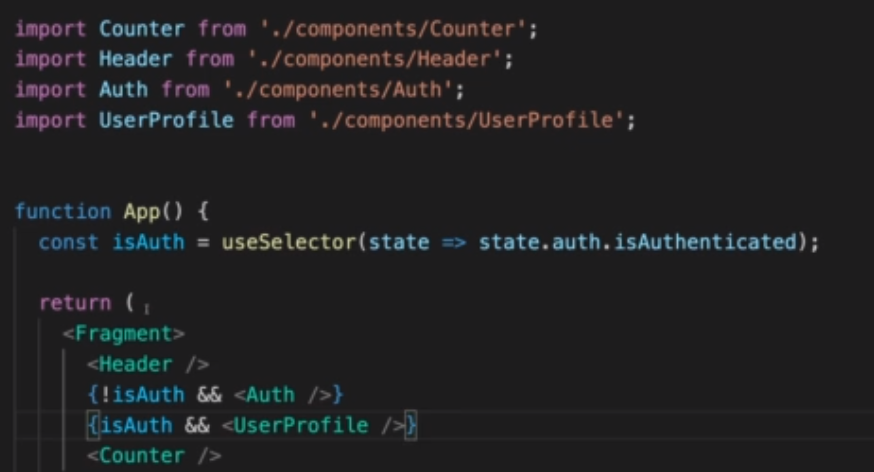
Again in order to create a new slice we can create a new state called initialAuthState and create a slice called authSlice ( can have any name).



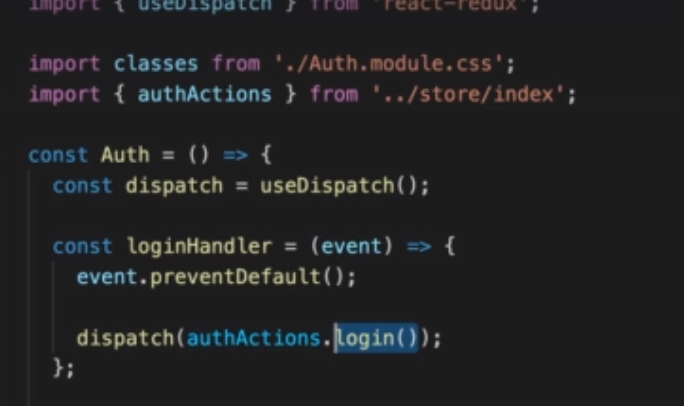
The above code then combines them into one store .

We can also expose the actions of this newly created slice by using authSlice.actions.

Now whenever we want to access the authentication value we can access it by using useSelector.



Lets dispatch the isAuthenticated state after user clicks on login button and that can be done as follows.



We can use useDispatch() for triggering reducer functions as shown above.

Similarly we can dispatch logout for logout functionality.

For large enterprise applications it makes sense to create separate file for every file and it becomes easy to work with.