Redux is interface & React-Redux is implementation

index.js

```
import './index.css'
import App from './App'
import React from 'react'
import { ReactDOM } from 'react'

import { store } from './app/store'
import { Provider } from 'react-redux'

const root = ReactDOM.createRoot(document.getElementById('root'))

root.render(
    <React.StrictMode>
        <Provider store={store}>   Making components aware of the store
        <App />
        </Provider>
    </React.StrictMode>
)
```

① Install redux & bindings of react & redux
Create a store
② Make all components aware of the store

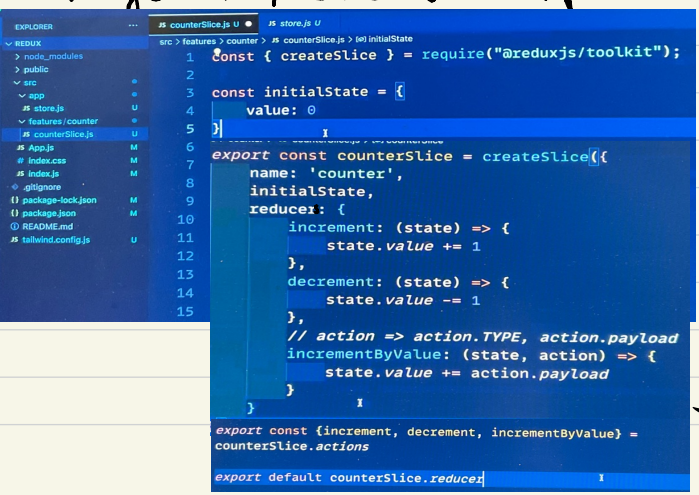
store.js

```
import { configureStore } from "@reduxjs/toolkit";
                                  ↳ store can be created with this store
export const store = configureStore({
  reducer: {}   → no need to define the type and then match it. Can directly call a method
})
```

* state is a built in React Object that is used to contain Data or information about the component
Whenever it changes the component rerenders

& Slice means Functionality

* Payload means value / parameter



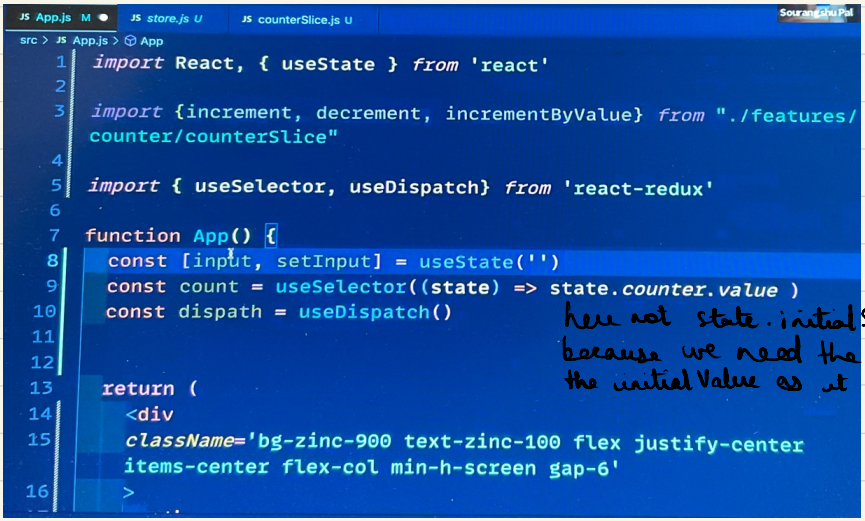

```
import { configureStore } from "@reduxjs/toolkit";
import counterSlice from "../features/counter/counterSlice";
                                  is made the store aware of methods
export const store = configureStore({
    reducer: {
        counter: counterSlice
    } // Video: VideoPlayerSlice
})
```

any random name

These are the only ways / methods by which my State can be manipulated

* **Useselector** : Selects the current value from the store
* **usedispatch** : after I imported the methods for my state manipulation in App.js and I want to use them. I access them via usedispatch

*



here not state.initialState.value (from store) because we need the real time value and not the initialValue as it will always be 0.

① · installed redux js/toolkit which is independent
· used react-redux library (binding) to attach with react

Step 1 : Create a store
↓
configure store which creates the store
↓
create object and inside it create reducer

Step 2 : Making every component aware that there exists a store
↓
import store
import Provider
Wrapped the entire App with Provider & pass on a prop store

* Now our store is ready we have to write some rules how are going to manipulate the values in our store and in term of redux we call them Slice (functionalities)

Step 3: We create createSlice, which provides a lot of abstraction to us. First we had to match the actions and payloads (values) so inorder to avoid these. so we use CreateSlice to avoid this matching

now we create functionalities:

* Inner Library helps us to prevent in mutation of the state

```
const initialState = {
        value : 0;
}
export const counterSlice = createSlice ({
      name: 'counter',
      initialState,
      reducers: {
            increment: (state) => {
                  state.value += 1
            },
            decrement: (state) => {
                  state.value -= 1
            }, // action : action.type , action.payload
            incrementByValue: (state, action) => {
                  state.value += action.payload
            }
      }
})
export const {increment, decrement, incrementByValue} =
 counterSlice.actions
export default counterSlice.reducer
```

I dont have to use it because of counterSlice

in App.js

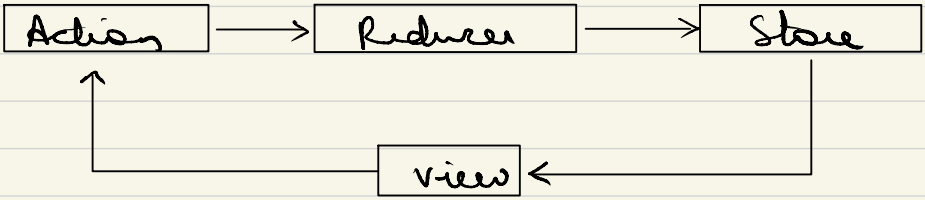
```
import {increment, decrement, incrementByValue} from "./features/
counter/counterSlice"
import {useSelector, Usedispatch} from 'react-redux'

function App() {
    const [input, setInput] = useState(' ');
    const count = useSelector((state) => state.counter.value)
    const dispatch = usedispatch()
    return (
      :
    <button onclick={() => dispatch(increment())}>
    </button>
      :
    <button onclick={() => dispatch(decrement())}>
    </button>
```

(local scope) (from store.js)

* An Action is a plain Object that describes the intention to cause Change.

* A reducer is a function that determines changes to an applications state. return the new state and tell the store how to do.

* it uses the action it receives to determine this change.



* Action
  Type: 'BRING'
  Payload: 1 ball + 1 bat