<https://www.digitalocean.com/community/tutorials/how-to-manage-state-in-react-with-redux>

redux using hooks <https://react-redux.js.org/api/hooks>

https://www.youtube.com/watch?v=j942wKiXFu8&list=PL4cUxeGkcC9gZD-Tvwfod2gaISzfRiP9d

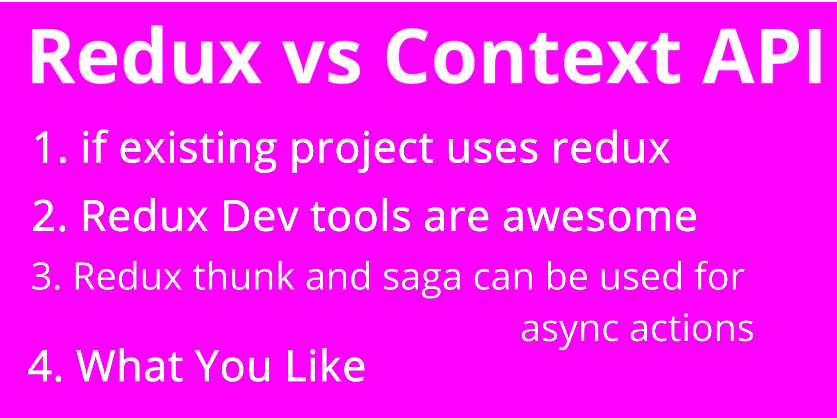
https://www.youtube.com/watch?v=ahZPLgj1tnY&list=PLB97yPrFwo5gpct1dOsirrotuPef1xBWS

<https://www.youtube.com/c/TheNetNinja/playlists>

For state management we can use two thing

1. Context Api
2. Redux

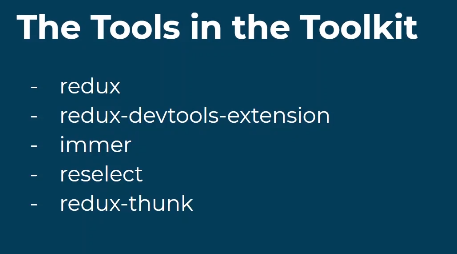




**React –redux toolkit- boliler code is reduced , bolier code means extra code which is not required**

[**https://www.youtube.com/watch?v=LZCZtFem74o**](https://www.youtube.com/watch?v=LZCZtFem74o)

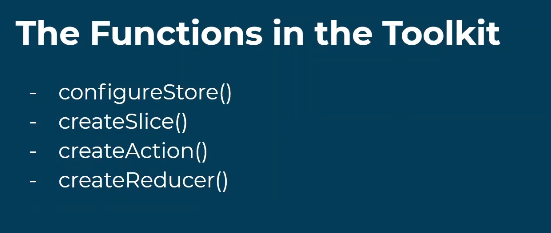
****

****

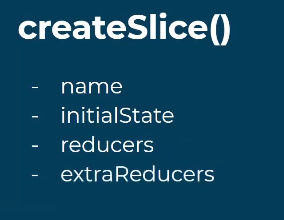
**Immer** offers a simple and elegant solution that allows developers to update their store by mutating it (changing values without creating a copy/new instance).

**Reselect** is a library for building memoized selectors. We define selectors as the functions that retrieve snippets of the **Redux** state for our React components. Using memoization, we can prevent unnecessary rerenders and recalculations of derived data which in turn will speed up our application – if value is same of state will not execute the code

**Redux Thunk** is a middleware that lets you call action creators that return a function instead of an **action object**. **That function receives the store's dispatch method**, which is then used to dispatch regular synchronous actions inside the function's body once the asynchronous operations have been completed







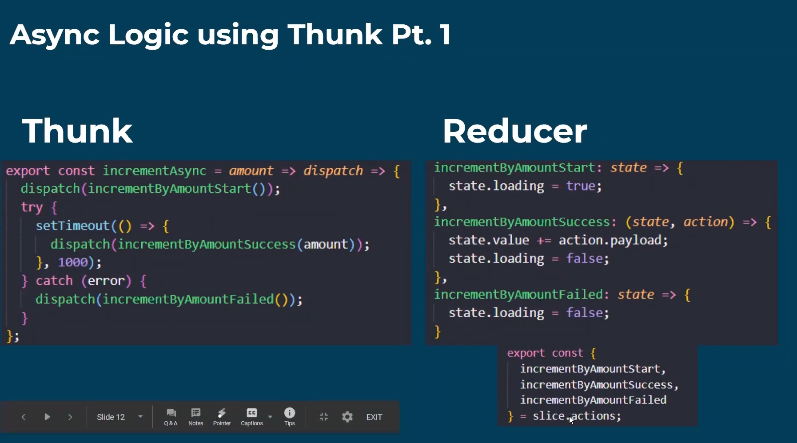
**In RTK- no need to create actions and constants**

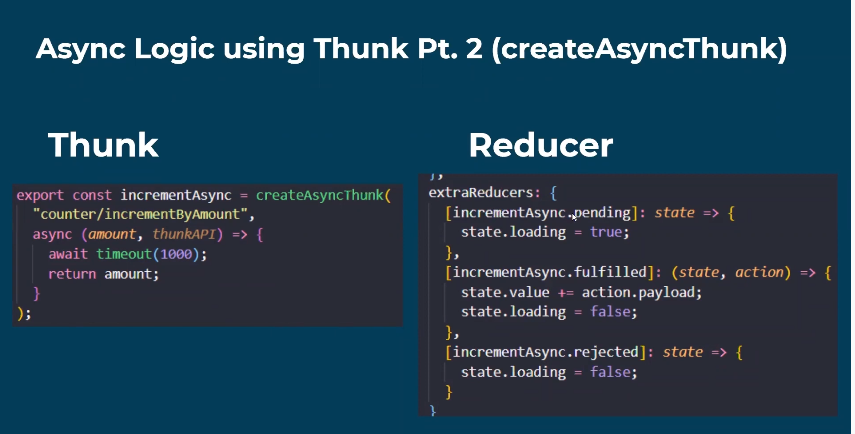


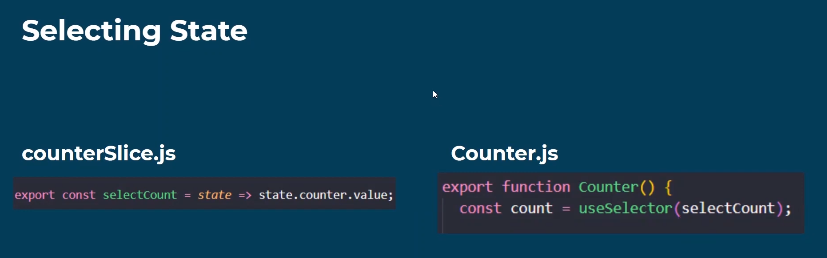
If we see we are using

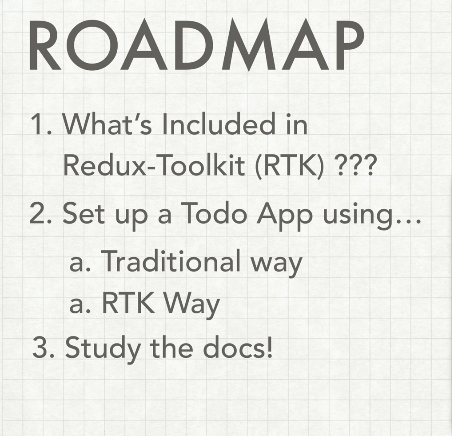
Export const {increament, decrement, incrementByAmount} = slice.actions

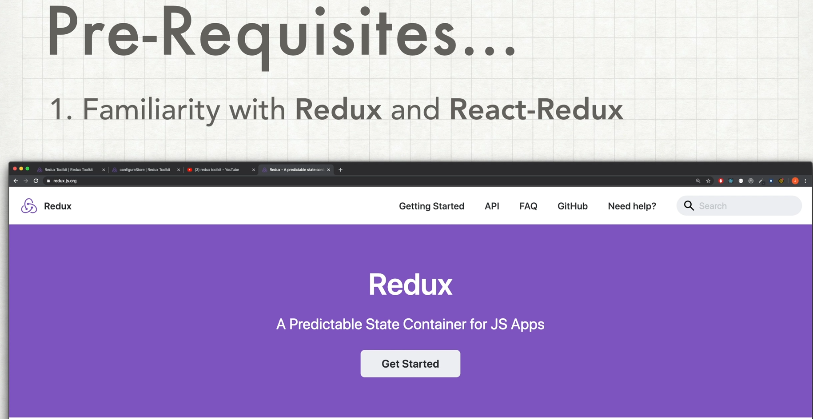
So slice.actions will create automatically action using destrturing name

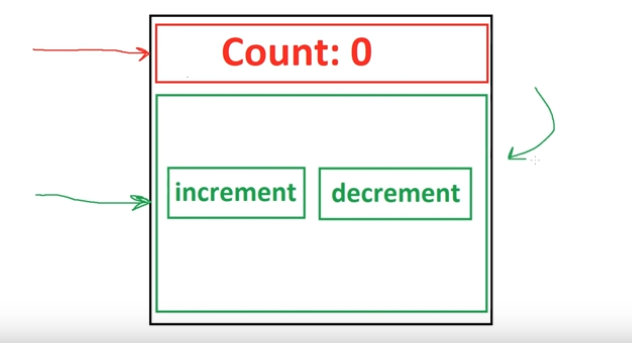












Increament and decrement functions in one Component & Count is different component, and we need to share data global, so we use redux

Npx create-react-app rtk

### Install Redux Toolkit and React-Redux[#](https://redux-toolkit.js.org/tutorials/quick-start" \l "install-redux-toolkit-and-react-redux" \o "Direct link to heading)

Add the Redux Toolkit and React-Redux packages to your project:

npm install @reduxjs/toolkit react-redux

after all installation, create folder in src/redux/configureStore.js

import { configureStore } from '@reduxjs/toolkit'

export default configureStore({

  reducer: {},

})

After this store creation we need to add this store into whole app in index.js only using providers

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

**import {Provider} from 'react-redux';**

**import store from './redux/configureStore';**

ReactDOM.render(

  <React.StrictMode>

**<Provider store={store}>**

    <App />

**</Provider>**

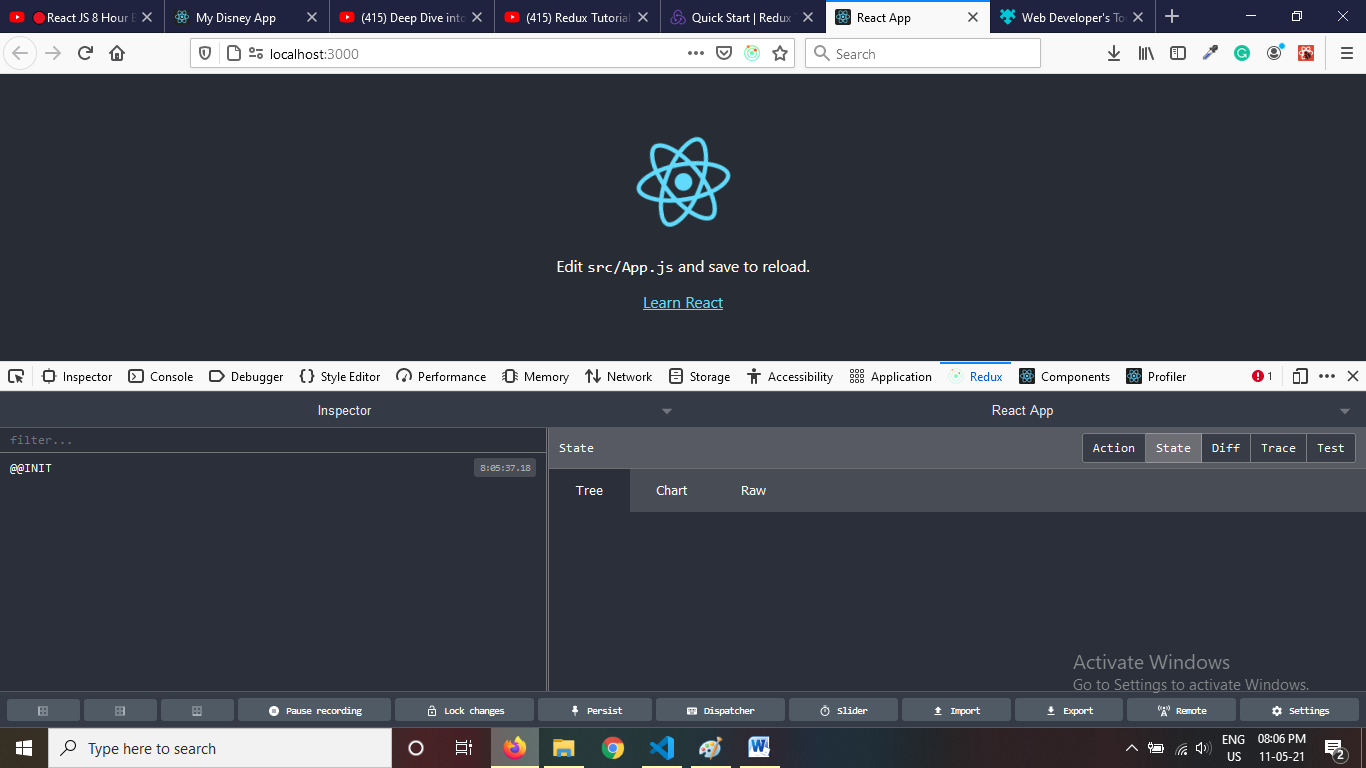
  </React.StrictMode>,

  document.getElementById('root')

);

reportWebVitals();

add redux toolbar extension



App.js

import logo from './logo.svg';

import './App.css';

function App() {

  return (

    <div className="App">

      <h1> The count is: </h1>

      <button >increment</button>

      <button >decrement</button>

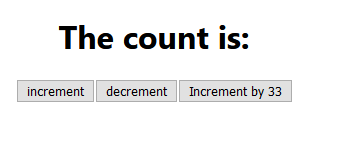
      <button >Increment by 33</button>

    </div>

  );

}

export default App;



### Create a Redux State Slice Src/redux/counter.js which is our state slice means

### it is a combination of name of reducer + initial state + reducer s action + return new state

import { createSlice } from '@reduxjs/toolkit'

export const counterSlice = createSlice({

  name: 'counter',

  initialState: {

    value: 0,

  },

  reducers: {

    increment: (state) => {

      // Redux Toolkit allows us to write "mutating" logic in reducers. It

      // doesn't actually mutate the state because it uses the Immer library,

      // which detects changes to a "draft state" and produces a brand new

      // immutable state based off those changes

      state.value += 1

    },

    decrement: (state) => {

      state.value -= 1

    },

    incrementByAmount: (state, action) => {

      state.value += action.payload //action.payload will get value as an argument in event function

    },

  },

})

// Action creators are generated for each case reducer function

export const { increment, decrement, incrementByAmount } = counterSlice.actionsexport default counterSlice.reducer

now we have to import this reducer into store

C:\rtk\src\redux\configureStore.js

import { configureStore } from '@reduxjs/toolkit'

import counterReducer from './counter'

export default configureStore({

  reducer: {

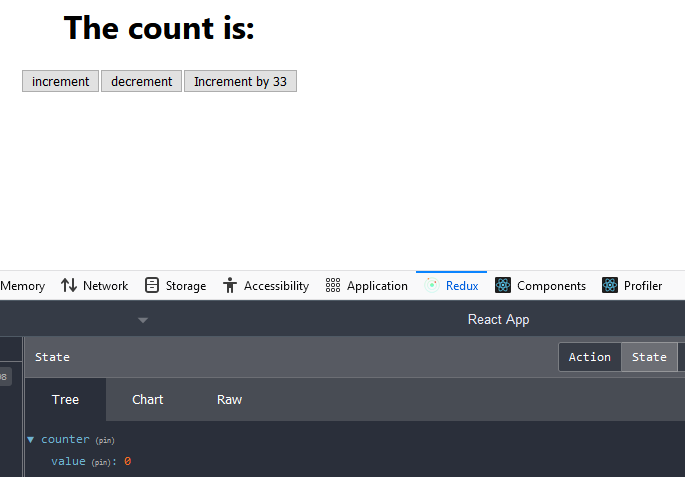
    /// name of reducer : import name of that reducer

    counter:counterReducers

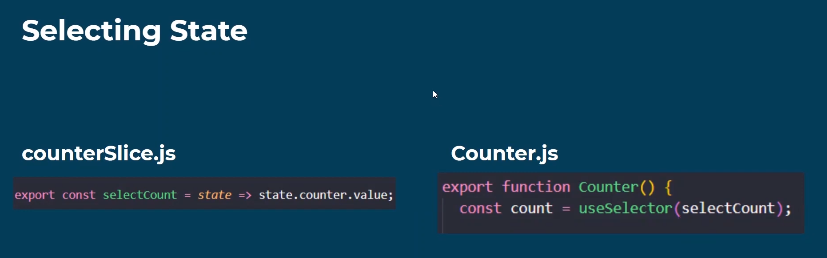
  },

})

Now this counter reducer is connected with application



Now we will use all actions name to bind events of app.js buttons using **useSelector hooks**



**App.js**

import logo from './logo.svg';

import './App.css';

import { useSelector } from 'react-redux';

import { useDispatch } from 'react-redux';

import { decrement, increment,incrementByAmount } from './redux/counter';

function App() {

// initalstate name we will use, Similar to mapStateToProps - use newstate=>state.nameofreducer

const {value}=useSelector(state=>state.counter);

//both two lines are sames

//initalstate name we will use, Similar to mapStateToProps - use newstate=>state.nameofreducer.initialstate key

//const value=useSelector(state=>state.counter.value);

//Now value is coming but we need to bind actions with buttons with help of dispatcher and using dispacter we call call action of reducer

const dispatch=useDispatch();

  return (

    <div className="App">

      <h1> The count is: {value} </h1>

      <button onClick={()=>dispatch(increment())}>increment</button>

      <button onClick={()=>dispatch(decrement())} >decrement</button>

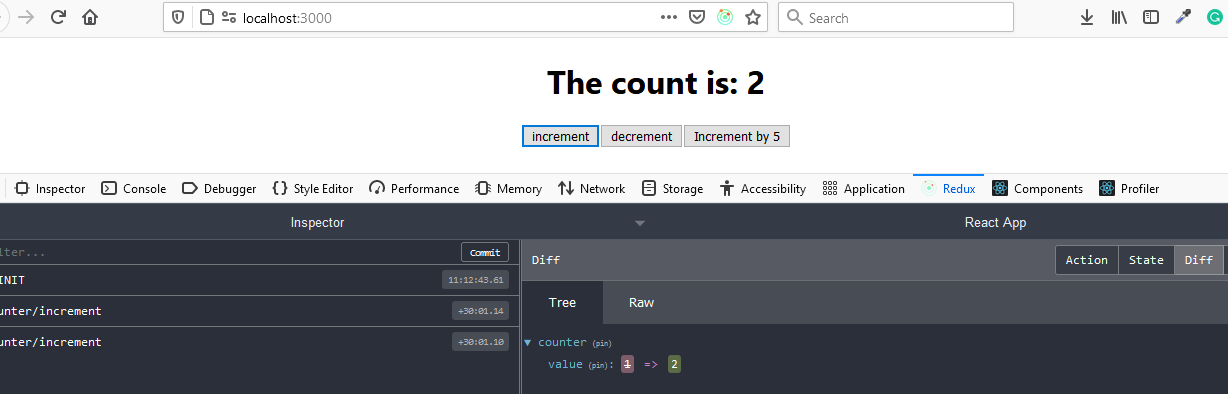
      <button onClick={()=>dispatch(incrementByAmount(5))}>Increment by 5</button>

    </div>

  );

}

export default App;



Finished

<https://www.youtube.com/watch?v=j942wKiXFu8&list=PL4cUxeGkcC9gZD-Tvwfod2gaISzfRiP9d>

Complete RTK <https://www.youtube.com/watch?v=9lCmbth63k0>

Clever https://www.youtube.com/watch?v=Dt3o4l\_OFa0