Redux.

React context disadvantage:

End with deeply nexted jsx code. Or huge context provider

Difficult to maintain.

Performance

Not optimized for high frequency changes

Having central Data(State) Store

Reducer is just a general concept

Node redux:

const redux = require('redux')

const couterReducer = (state = {counter:0},action)=>{

    if(action.type =="increment"){

        return {

            counter: state.counter + 1

        };

    }

    if(action.type =="decrement"){

        return {

            counter: state.counter - 1

        };

    }

    return state;

};

const store = redux.createStore(couterReducer)

const counterSubscriber = ()=>{

   const latestState= store.getState();

   console.log(latestState)

}

store.subscribe(counterSubscriber);

store.dispatch({type:"increment"})

store.dispatch({type:"decrement"})

npm i redux react-redux

react redux:

import React from 'react';

import ReactDOM from 'react-dom/client';

import {Provider} from 'react-redux';

import './index.css';

import App from './App';

import store from './store/index';

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

root.render(

<Provider store={store}>

<App />

</Provider>);

import { createStore } from "redux";

const counterReducer = (state={counter:0},action) =>{

    if(action.type=="increment"){

        return {counter:state.counter+1}

    }

    if(action.type=="decrement"){

        return {counter:state.counter-1}

    }

    return state

}

const store = createStore(counterReducer)

import classes from './Counter.module.css';

import { useSelector,useDispatch } from 'react-redux';

const Counter = () => {

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

  const dispatch = useDispatch();

  const toggleCounterHandler = () => {

  };

  const incrementHandler = () => {

    dispatch({type:'increment'})

  };

  const decrementHandler = () => {

    dispatch({type:'decrement'})

  };

  return (

    <main className={classes.counter}>

      <h1>Redux Counter</h1>

      <div className={classes.value}>{counter}</div>

      <div>

        <button onClick={incrementHandler}>

          increment

        </button>

        <button onClick={decrementHandler}>

          decrement

        </button>

      </div>

      <button onClick={toggleCounterHandler}>Toggle Counter</button>

    </main>

  );

};

export default Counter;

payload to action:

  const increaseHandler =()=>{

    dispatch({type:'increment',amount:5})

  }

const counterReducer = (state={counter:0,value:0},action) =>{

    if(action.type=="increment"){

        return {counter:state.counter+action.amount}

    }

Never mutate the state

CombineReducer to combine multiple reducers

Redux toolkit:

npm i @reduxjs/toolkit

after installing we don’t need ‘redux’ package in store.

Avoid action type conflicts

Here we can mutate the state

configureStore()

to manage multiple reducers

import { createSlice,configureStore } from "@reduxjs/toolkit";

const initialState = { counter: 0, showCounter: true };

const counterSlice = createSlice({

    name:'counter',

    initialState:initialState,

    reducers:{

        increment(state){

            state.counter++;

        },

        decrment(state){

            state.counter--

        },

        increase(state,action){

            state.counter= state.counter+action.payload

        },

        toggleCounter(state){

            console.log(state)

            state.showCounter = !state.showCounter

        }

    }

});

const store = configureStore({

    reducer:counterSlice.reducer

});

For multiple reducer

const store = configureStore({

    reducer:{counter:counterSlice.reducer}

});

counterSlice.actions.decrment()

this will automatically create action identifiers

export const counterActions = counterSlice.actions;

  const counter = useSelector(state=> { console.log(state);return state.counter.counter});

  const showCounter = useSelector(state=>state.counter.showCounter);

   dispatch(counterActions.toggleCounter())

redux and side effects