**Redux (Follow the redux toolkit method)**

****

Demanding ball and bat

**(The Image explains clearly the whole concept of the reduces, action and dispatcher where perfrectly (It helps to understand the concept nicely))**

1. Where we want to **change** the value of the data in the state than use the **dispatcher** and when we want to **use** the value of the data than use the **useSelector** hook
2. Redux is a Complex State Management Tool, with a single store as **CDS(Central Data Store)**
3. **Reducers** manages the state and returns the newly updated state
4. Actions have 2 properties type: which is a **unique identifier** and **payload which has the data**
5. **Dispatch** is used to send actions to update the data
6. Steps for making or using the redux technology :
7. Make an react application by the help of the **npx create-react-app <app-name>**
8. Then go into the terminal of that application by opening that folder in the vs code OR go via the cd <app-name> in terminal
9. Then install the redux by the help of the following codes in the terminal :

**npm install redux react-redux**

1. And the packages will be installed and do not take care about the **vulnerabilities** that comes
2. Then in the src folder of the react app, we have to make the new folder named the store and in it make a new file named the index.js
3. First import the createStore from the redux by the help of the following code

**import {createStore} from ‘redux’;**

Then we have to make the arrow function named the reducerFn which is has the two things the first is the state and the second is the payload, and in this case we have taken the action as the payload, Example

**const reducerFn=(state, action)=>{**

**}**

1. And then we have to create an store by the help of the createStore method which takes the reducerFn as the input, Example

**const store=createStore(reducerFn);**

1. And then we will give the initial property to an variable in the state in the reduerFn, suppose we take counter initially, Example

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

**return state;**

**};**

1. So now we can get easily the value of the counter in whole file where we want to use it which was given value in the state in the store
2. And we have to also export the store from the index.js by writing the following statement on the ending of the document

**export default store;**

1. Now, in the index.js of the src folder we will import the provider from the react-redux, example

**import {Provider} from ‘react-redux’**

1. And then in this file bellow the render function we have to write the tag of the alias of the provider and in it we have to pass the store as the store and then, at the end of the render method, we have to complete the alias of the provider, Example

**<Provider store={store}>**

**….**

**</Provider>**

🡪The import of the store will also been needed, if done automatically than its okay, otherwise do it mannualy by

**import store from “./store/index”**

1. And now we have to handle the state and show the state
2. Now we will go to the App.js,We will remove the everything present in the return method, only the div tags at the starting and the ending will be not removed
3. Now to access the value from the store, we will take help of the useSelector hook, and we will import it by

**import {useSelector} from “react-redux”;**

1. And then we will take the value of the counter by :

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

1. Now to see the value of the counter, we will write the simple statement, It can be anything, For example:

**<h2>{counter}</h2>**

1. As it is an react application, to run it we have to write the **npm start** in the terminal
2. Now, to change the value of the object or the variable stored in the redux store, We have to use the dispatcher, For using it we have to first import the following statement

**import {useDispatch} from ‘react-redux’;**

1. And by the use of the dispatcher, It will fetch the action and update the value in the store
2. And for doing so, we have to give the reference of useDispatch to the dispatch, in the App.js or where we want to use, Example

**const dispatch =useDispatch();**

1. Now, to change the value of any thing we will use the dispatch function and in it we will give the name of the function which is present in the redux store, For example there is an increment function which is been called on the click and in it we will use the dispatch function,

**const increment =()=>{**

**dispatch({type: ‘INC’});**

**};**

1. And the name INC will call an function(not exactly but via if statement) in the reducer function which is been already defined
2. There are some limitations for the react-reducer functions, They are :
   1. The function should be synchronised function
   2. The function should not mutate the original state
3. And as we know we have maked the store in index.js which is in the store folder, And in it we will write in the method like following in the reducerFn,

**const reducerFn=(state={counter: 10}, action)=>{**

**if(action.type === ‘INC’){**

**return {counter: state.counter +10}**

**}**

**}**

1. In the function like the INC, We can also pass the payload in the action to the store, For example

🡪And in the payload we can pass any value

**const addBy=()=>{**

**dispatch({type: ‘ADD’, payload: 10});**

**}**

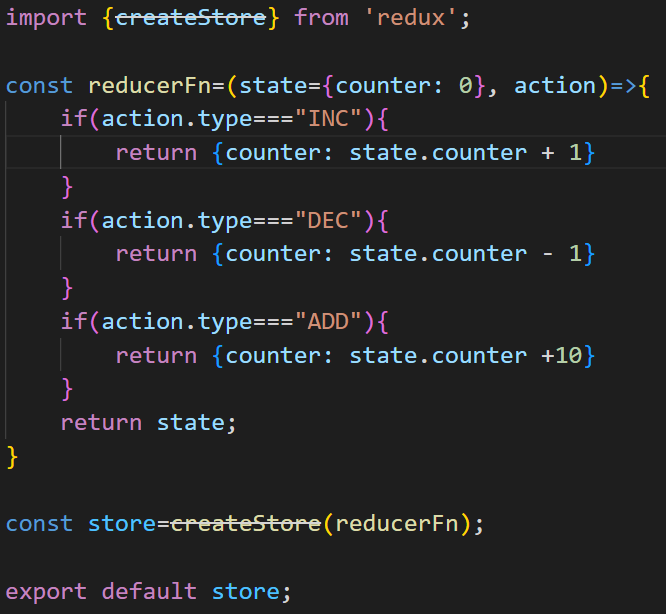
🡪And in the index.js/store we can use the payload value as the :

**if(action.type === ‘ADD’){**

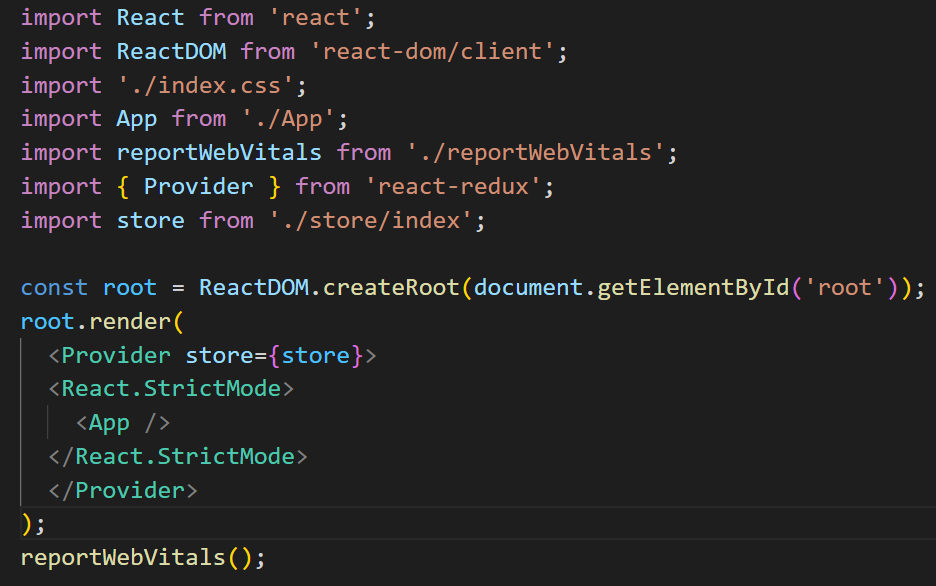
**return {counter: state.counter + action.payload};**

**}**

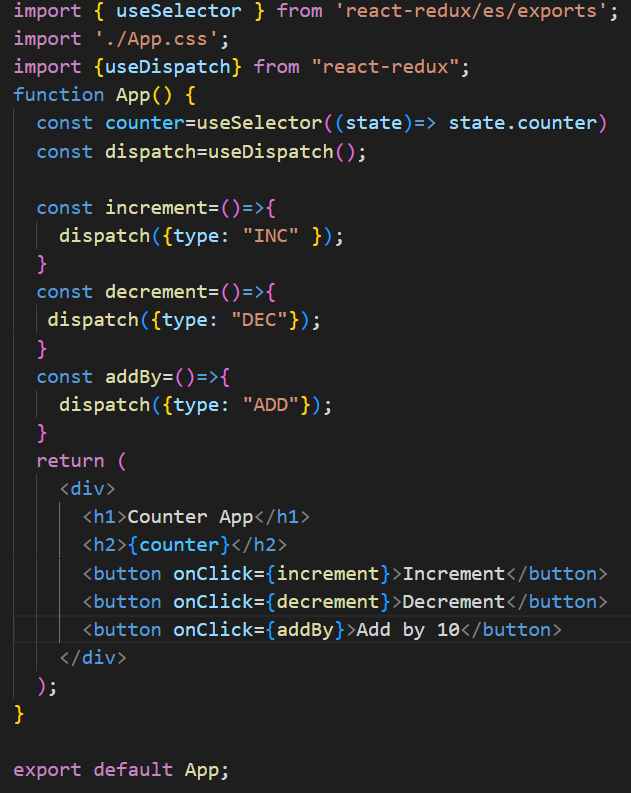
1. The screen shots of the basic program of the increment, decrement and add10 is as follow :



index.js/store



index.js/src



App.js

🡺Redux Tool-Kit(Follow this method)

🡪The redux tool-kit is being preferred over the simple redux libraray/package

🡪The redux tool-kit is simple, opinionated, powerful, effective

1. We can make an default redux-react app by using the following commands in the terminal where we want to make the project folder

**npx create-react-app <app-name> --template redux**

1. And we can also implement the redux toolkit to the exisiting react app by writing following statement in the terminal of the project

**npm install @reduxjs/toolkit**

1. For example we will make the some changes in the above app and will make it an redux tool-kit app
2. In the index.js of the store folder in the src we will write the following code(If app made in the simple redux than make the all thing remove and add the following thing to make it the app made by the use of the redux toolkit)🡪Follow the steps written bellow
3. Write the following import statement in the starting of the index.js/store

**import {configureStore, createSlice} from ‘@reduxjs/toolkit’**

1. Then we will make an function as shown bellow :

**const counterSlice=createSlice({**

**name: ‘counter’,**

**initialState:{counter: 0},**

**reducers: {**

**increment(state, action) {**

**state.counter++;**

**},**

**decrement(state, action) {**

**state.counter--;**

**},**

**addBy(state, action) {**

**state.counter+ action.payload;**

**}**

**}**

**})**

**export const actions=counterSlice.actions;**

**const store=configureStore({**

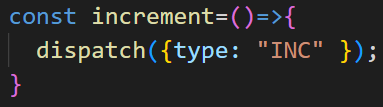
**reducer: counterSlice.reducer**

**})**

1. And in the App.js we have to import the following statement

import {actions} from ‘./store/index’

1. And the functions calling the store will also be changed from this to this :



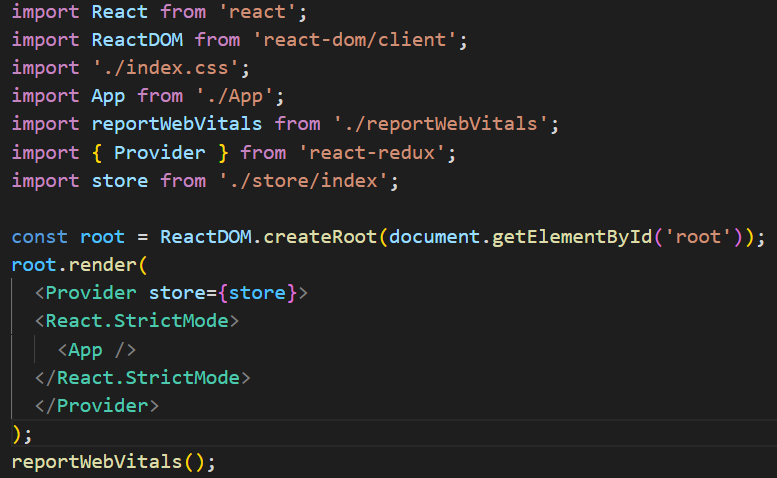
TO

🡪And if we want to pass the payload than we have to write :

1. The screen shorts of the app made by react redux toolkit is as follow :



index.js/store



index.js/src



App.js

1. In the redux toolkit, we can also make the different slice and than use it in the store and than access it’s state
2. The pure example of the usage of the redux toolkit of the login page is as follow:

🡪Steps to make such method is as follow:

1. Make an folder named the store in the src folder
2. Than in it we have to make an index.js file
3. And also another file which is an slice, which we will use to make different reducers
4. In this situation we have to make the authentication of the user, And hence we will make a file named Auth-slice.js(It is not mandatory to use the same name, we can any slice name as per our project requirement)
5. Now, In <name>-slice.js we will make the following additions(It is w.r.t login page, some things can be changed as per requirment)

**🡪Here the name authSlice, Auth used can be changed**

**🡪Here the name of the function and its state can be changed as per requirement**

import {createSlice} from “@reduxjs/toolkit”;

const authSlice=createSlice({

name: “Auth”,

initialState: {isLoggedIn: false},

reducers:{

login(state){

state.isLoggedIn=true;

},

logout(state){

state.isLogegdIn=false;

},

}

})

export const AuthActions=authSlice.actions;

export default authSlice;

**🡪Don’t forget to make the actions and export the AuthActions and authSlice**

1. We can make the as many as required slice in any project, but we have to also link it with the index.js in the store folder and this file is used as the store, We will make it in the next step
2. Now the following code is for making an store

import {configureStore} from “@reduxjs/toolkit”;

import authSlice from “./Auth-slice”;

const store=configureStore({

reducer: {

auth: authSlice.reducer

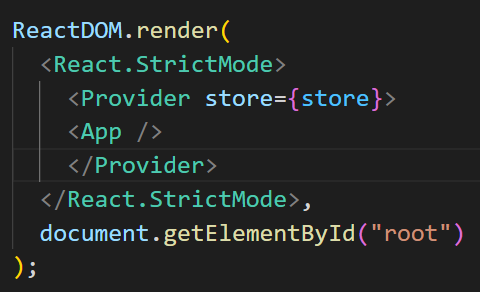
}

})

export default store;

🡪And in it we have to mention all the slice name in the reducer method

1. And in the index.js in the src folder, we have to make the import of the following two statements:
   1. **import {Provider} from ‘react-redux’;**
   2. **import store from ‘./store’;**
2. And at the ending and the starting of the react.StrictMode alias, we have to add the Provider alias with the state in it, Example



1. Now where we want to use the store we have to write the following statements
2. Write: **import {useSelector} from “react-redux”;**
3. **const isLoggedIn=useSelector(state=>state.auth.isLoggedIn);**

🡪Here the isLoggedIn is been defined in the auth state, If any other thing is defined in it, than use it’s name

🡪And now in code we can use the isLoggedIn and we can access it’s value by its name

🡪This can be used to decide in the App.js to transfer to which page as per the login situation

1. And we can also change its value(In this project we will do it in the login page after verifying the details we will change the isLoggedIn from false to true)
2. For making the change in the we assume that the form has the submit button and by clicking it the handleSubmit function will be called, Example(We are writing this in the component of the react app names Auth.js)

🡪The required imports and mehods are written in the following example, include them in the UI that you had maded

import {useDispatch} from ‘react-redux’;

import {AuthActions} from ‘../store/Auth-slice’;

const Auth=()=>{

const dispatch=useDispatch();

const handleSubmit=(event)=>{

event.preventDefault();

//dispatch

dispatch(AuthActions.login())

};

}

🡪And this was the last step and our login system is ready to be used

1. In the redux while we use the dispatch method than the whole App.js is been rendered again and whole thing is being repeated
2. To change the value, we use the dispatcher
3. And to get the value, we use the useSelector
4. We can also change the state to the opposite, if the variable is having the Boolean value by using the following logic :

setShowCart(state){

            state.showCart =!state.showCart;

        },

1. Reducers must be pure, side effect free and asynchronous functions always
2. So, if we have an HTTP call which has an asynchronous stance, So such type of the code would not go through the reducer functions
3. So we cant send the fetch and resend request from the one reducer function to the other
4. So it is strongly suggested that do not run the synchronous code here
5. Instead we have the two options to put the synchronous code :
   1. Inside the components using the useEffect hook
   2. Create an actionCreater which will allow us to run the asynchronous code(Thunk)
6. For backend we will use the firebase in both the cases for the sample code
7. And the example of the useEffect hook is <https://github.com/NeelDevenShah/Redux-Shopping-Cart-App.git> (1st commit)
8. The second method is of the thunk method

🡪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.

**🡪We can use the any patten but personally the first one is more liked by me**