REDUX:

Redux is a predictable state container for JavaScript apps.

REDUX TOOLKIT:

Redux toolkit is our official approach (according to redux website) for writing Redux logic. It wraps around the Redux core, and contains packages and functions that we think are essential for building a Redux app. Redux Toolkit builds in our suggested best practices, simplifies most Redux tasks, prevents common mistakes, and makes it easier to write Redux applications.

I am following redux toolkit in this example

Dependencies to install:

yarn add @reduxjs/toolkit

yarn add redux

yarn add react-redux

yarn add axios (for async createAsyncThunk if needed)

STEPS:

1. Create slice.js file containing all states and actions
2. Create store
3. Wrapp index.js file in provider with store
4. Creating userSlice.js file:

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

export const userSlice = createSlice({

    name:"user",

    initialState:{

        name: "leadastaire",

        email: "muneebwaseem@leadastaire.com"

    },

    reducers:{

        update: (state,action) =>(

            state.name= action.payload.name,

            state.email= action.payload.value

        ),

        remove: (state) => (

            state= {}

        )

    }

})

export const {update, remove} = userSlice.actions;

export default userSlice.reducer;

1. Creating store using the above create reducer:

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

import userReducer from './userSlice'

export const store = configureStore({

    reducer: {

        user: userReducer

    }

})

To use name and email anywhere, state.storename.name or something like that

In our case;

State.user.name and state.user.email

1. Wrapping index.js file

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

import {store} from './redux/store'

import { Provider } from "react-redux";

ReactDOM.render(

  <React.StrictMode>

    <Provider store={store}>

    <App />

    </Provider>

</React.StrictMode>,

  document.getElementById("root")

);

IMP HOOKS:

To hooks are very important when working with redux

1. useSelector(state => state.storename.name)

Example:

Const name = useSelector(state=>state.user.name)

It is used to use any value from store

1. useDispatch()

Example:

Const dispatch = useDispatch()

dispatch(update({name, email})

useDispatch used to update store states

here we create update function in reducer file and it requires both name and email, so in dispatch we use update function and update store by providing some new name and email

Demonstration of useSelector hook

import { useSelector } from "react-redux";

const name = useSelector(state => state.user.name)

<span className="navbarName">{name}</span>

Second demonstration:

const user = useSelector(state => state.user) //return till store

placeholder={user.name}

Demonstration of useDispatch:

import React, { useContext } from "react";

import Warning from "../warning/Warning";

import "./update.css";

// import { update, remove } from "../../redux/userSlice";

import { useState } from "react";

import { useSelector } from "react-redux";

import { useDispatch } from "react-redux";

import { remove, update } from "../../redux/userSlice";

export default function Update() {

  const dispatch = useDispatch()

  const [name, setName] = useState('');

  const [email, setEmail] = useState('')

  const user = useSelector(state => state.user) //return till store

  const handleClick = (e)=>{

    e.preventDefault();

    dispatch(update({name, email}))

  }

  const handleDelete = (e)=>{

    e.preventDefault();

    dispatch(remove())

  }

  return (

    <div className="update">

      <div className="updateWrapper">

        <h3 className="updateTitle">Update Your Account</h3>

        <Warning />

        <button className="delete" onClick={handleDelete}>Delete Account</button>

        <div className="updateContainer">

          <form>

            <div className="formItem">

              <label>Profile Picture</label>

              <div className="profilePic">

                <img

                  className="avatar"

                  src="https://images.pexels.com/photos/3024627/pexels-photo-3024627.jpeg?auto=compress&cs=tinysrgb&dpr=2&w=500"

                  alt=""

                />

                <span className="change">Change</span>

              </div>

            </div>

            <div className="formItem">

              <label>Username</label>

              <input

                className="formInput"

                type="text"

                placeholder={user.name}

                onChange = {(e)=>setName(e.target.value)}

              />

            </div>

            <div className="formItem">

              <label>Email</label>

              <input

                className="formInput"

                type="text"

                placeholder={user.email}

                onChange={(e) => setEmail(e.target.value)}

              />

            </div>

            <div className="formItem">

              <label>Password</label>

              <input className="formInput" type="password" />

            </div>

            <button

              className="updateButton"

              onClick={handleClick}

            >

              Update

            </button>

          </form>

        </div>

      </div>

    </div>

  );

}

In this file, we are update app based on entering name and email in the form. For that we created const of name and email using useState() hook

After that in form , we update both name and email using onChange method which updates name and email, now to update name and email in store requires useDispatch, so we run it on handleClick, and inside that, we pass ;

dispatch(update({name, email}))

thus our store will be updated with new name and email, and all places using these values will automatically gets updated

similarly, handleDelete runs dispatch(remove()) from reducer file that will set state to empty object, means remove any name and email set in the store

WHEN WORKING WITH API OR ASYNC OPERATION:

In this case, we need to use createsyncthunk() method

1. Store remains the same
2. Wrapper remains the same
3. Update slice file or userSlice.js file

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

import axios from 'axios';

export const updateUser2 = createAsyncThunk("users/update", async (user)=>{

     const response = await axios.post("http://localhost:8800/api/users/1/update", user);

     return response.data

});

//In above we define const, users/update is just dummmy

//we pass user from update.jsx file to it, it will post that to api

//and in return it will send us response back

//and we can run pending, success and rejected cases on this api in extra reducers

export const userSlice = createSlice({

    name: "user",

    initialState:{

        userInfo:{

            name: "Jack",

            email: "jack@gmail.com"

        },

        pending: null,

        error: false

    },

    reducers: {},

    //use extra reducers for async thunk, consist of pending, fulfilled(success), rejected(error)

    extraReducers:{

        [updateUser2.pending]: (state)=>{

            state.pending = true;

            state.error = false

        },

        //when pending false and api run successfully, we send data to that api using action.payload from update.jsx file using dispatch

        [updateUser2.fulfilled]: (state, action)=>{

            state.pending = false;

            state.userInfo = action.payload

        },

        [updateUser2.rejected]: (state)=>{

            state.pending = null;

            state.error = true

        }

    }

})

export default userSlice.reducer;

As we place name and email inside userInfo object, so our useSelector will be like;

useSelector(state => state.user.userInfo.name) (state.storename.userInfo.name)

Now code for update.jsx file where we use updateUser2 method to pass user data to api

import React, { useContext } from "react";

import Warning from "../warning/Warning";

import "./update.css";

// import { update, remove } from "../../redux/userSlice";

import { useState } from "react";

import { useSelector } from "react-redux";

import { useDispatch } from "react-redux";

import { remove, update, updateUser2 } from "../../redux/userSlice";

export default function Update() {

  const dispatch = useDispatch()

  const [name, setName] = useState('');

  const [email, setEmail] = useState('')

  const {userInfo, pending, error} = useSelector(state => state.user) //return till store

  const handleClick = (e)=>{

    e.preventDefault();

    dispatch(updateUser2({name,email}))

  }

  const handleDelete = (e)=>{

    e.preventDefault();

    // dispatch(remove())

  }

  return (

    <div className="update">

      <div className="updateWrapper">

        <h3 className="updateTitle">Update Your Account</h3>

        <Warning />

        <button className="delete" onClick={handleDelete}>Delete Account</button>

        <div className="updateContainer">

          <form>

            <div className="formItem">

              <label>Profile Picture</label>

              <div className="profilePic">

                <img

                  className="avatar"

                  src="https://images.pexels.com/photos/3024627/pexels-photo-3024627.jpeg?auto=compress&cs=tinysrgb&dpr=2&w=500"

                  alt=""

                />

                <span className="change">Change</span>

              </div>

            </div>

            <div className="formItem">

              <label>Username</label>

              <input

                className="formInput"

                type="text"

                placeholder={userInfo.name}

                onChange = {(e)=>setName(e.target.value)}

              />

            </div>

            <div className="formItem">

              <label>Email</label>

              <input

                className="formInput"

                type="text"

                placeholder={userInfo.email}

                onChange={(e) => setEmail(e.target.value)}

              />

            </div>

            <div className="formItem">

              <label>Password</label>

              <input className="formInput" type="password" />

            </div>

            <button

              className="updateButton"

              onClick={handleClick}

              disabled={pending}

            >

              Update

            </button>

            {error && <span className="error">Something went wrong!</span>}

            {pending === false && (

              <span className="success">Account has been updated!</span>

            )}

          </form>

        </div>

      </div>

    </div>

  );

}





