|  |
| --- |
| Constredux=require('redux'); |
|  | const createStore=redux.createStore; |
|  |  |
|  | const intialState={ |
|  | numberOfBooks:10, |
|  | numberOfPens:15 |
|  | } |
|  |  |
|  | // action creator: wraping the action in a single function |
|  | function buyBook(){ |
|  | return { |
|  | type:"Buy\_Book", |
|  | payload:"My First Action" |
|  | } |
|  | } |
|  |  |
|  | function buyPen(){ |
|  | return { |
|  | type:"Buy\_Pen", |
|  | payload:"My Second Action"}} |
|  |  |
|  |  |
|  | const Reducer =(state=intialState,action)=>{ |
|  | switch(action.type){ |
|  | case "Buy\_Book":return{ |
|  | ...state, |
|  | numberOfBooks:state.numberOfBooks-1 |
|  | } |
|  |  |
|  | case "Buy\_Pen":return{ |
|  | ...state, |
|  | numberOfPens:state.numberOfPens-2 |
|  | } |
|  |  |
|  | default: return state;}} |
|  |  |
|  |  |
|  |  |
|  | const store =createStore(Reducer); |
|  | console.log("Initial State",store.getState()); |
|  | const unsubscribe=store.subscribe(()=>{console.log('Updated State Value', store.getState())}); |
|  | store.dispatch(buyBook()); |
|  | store.dispatch(buyBook()); |
|  | store.dispatch(buyBook()); |
|  | store.dispatch(buyPen()); |
|  | store.dispatch(buyPen()); |
|  | store.dispatch(buyPen()); |
|  | unsubscribe(); |

middleware

const redux=require('redux');

const createStore=redux.createStore;

const combineReducers=redux.combineReducers;

const applyMiddleware =redux.applyMiddleware;

const intialStateBooks={

numberOfBooks:10,}

const intialStatePens={

numberOfPens:15}

// action creator: wraping the action in a single function

function buyBook(){

return {

type:"Buy\_Book",

payload:"My First Action" }}

function buyPen(){

return {

type:"Buy\_Pen",

payload:"My Second Action" }}

// (prevState,action)=>newState

const booksReducer =(state=intialStateBooks,action)=>{

switch(action.type){

case "Buy\_Book":return{

...state,

numberOfBooks:state.numberOfBooks-1 }

default: return state; }}

const pensReducer =(state=intialStatePens,action)=>{

switch(action.type){

case "Buy\_Pen":return{

...state,

numberOfPens:state.numberOfPens-2 }

default: return state; }}

const reducer=combineReducers({

booksReducer,

pensReducer});

const logger=store=>{

return next=>{

return action=>{

const result=next(action);

console.log("middleware log",result);

return result; } }}

const store =createStore(reducer,applyMiddleware(logger));

console.log("Initial State",store.getState());

const unsubscribe=store.subscribe(()=>{console.log('Updated State Value', store.getState())});

store.dispatch(buyBook());

store.dispatch(buyBook());

store.dispatch(buyBook());

store.dispatch(buyPen());

store.dispatch(buyPen());

store.dispatch(buyPen());

unsubscribe();

thunk

const redux= require('redux');

const createStore= redux.createStore;

const applyMiddleware =redux.applyMiddleware;

const thunkMiddleware=require('redux-thunk').default;

const axios = require('axios');

const initialState={

loading:false,

users:[],

error:''}

const USER\_REQUEST='USER\_REQUEST';

const USER\_SUCCESS='USER\_SUCCESS';

const USER\_ERROR='USER\_ERROR';

const userRequest=()=>{

return{

type:USER\_REQUEST}}

const userSuccess=(users)=>{

return{

type:USER\_SUCCESS,

payload:users}}

const userError=(error)=>{

return{

type:USER\_ERROR,

payload:error}}

const reducer=(state=initialState,action)=>{

switch(action.type){

case "USER\_REQUEST": return{

...state, loading:true }

case "USER\_SUCCESS": return{

loading:false, users:action.payload,error:'' }

case "USER\_ERROR": return{

loading:false, users:[],error:action.payload } }}

const fetchUser=()=>{

return function(dispatch){

dispatch(userRequest())

axios.get('https://jsonplaceholder.typicode.com/users')

.then(response=>{

// response.data

const users =response.data.map(user=>user.name)

dispatch(userSuccess(users)) })

.catch(error=>{

// error.message

dispatch(userError(error.message)) }) }}

const store =createStore(reducer,applyMiddleware(thunkMiddleware));

const unsubscribe=store.subscribe(()=>{console.log(store.getState())});

store.dispatch(fetchUser());

//unsubscribe();

Apply with react

Npm Install redux react-redux

In component folder

Create redux folder

Create book folder

In book crate bookType.js

Export Const BUY\_BO0K=”BUY\_BOOK”;

Create BookAction.js file

Import {BUY\_BOOK} from ‘./bookType’;

Export Const buyBook=()=>{

Return {

Type:BUY\_BOOK

}

}

Create bookReducer.js

Import {BUY\_BOOK} from ‘./bookType’;

Const initialState={

NoofBook:10

}

Const bookReducer=(state=initialStaet,action)=>{

Switch(action.type){

Case BUY\_BOOK:RETURN{

…STATE,

NUMBEROFBOOKS:State.NUMBEROFBOOKS-1

}

DEFAULT:RETURN STATE;

}}

EXPORT DEFAULT BOOKREDUCER;

NOW WE HAVE REDUCER WHEre BOOK FOLDER AND ALL STUf are in side book

BUT STORE CREATE OUT SIDE BOOK AND INSIDE REDUX FLDER

Cratae store.js

Import {createStore} from ‘redux’;

Import bookReducer from ‘./book/bookReducer’;

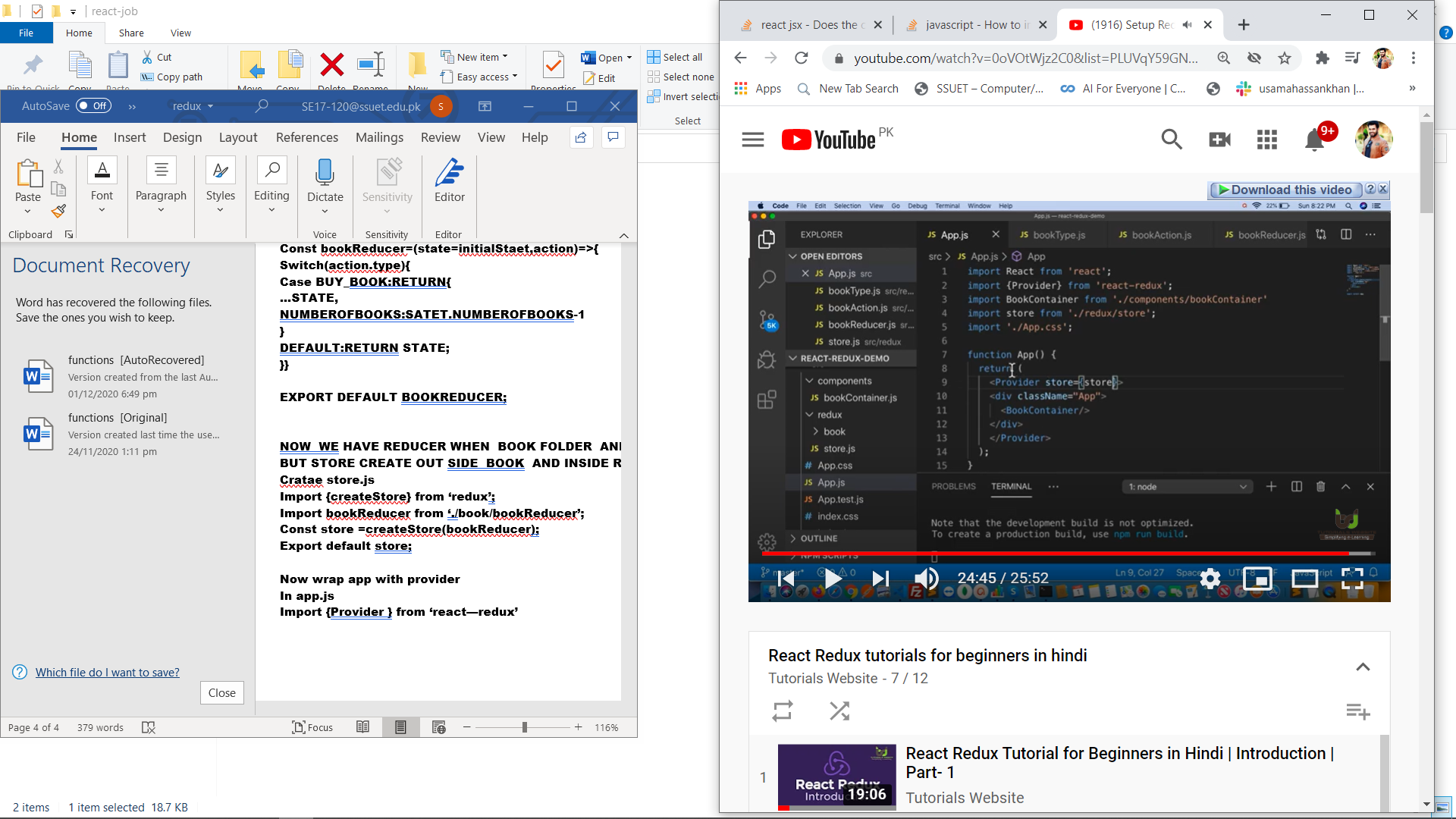
Const store =createStore(bookReducer);

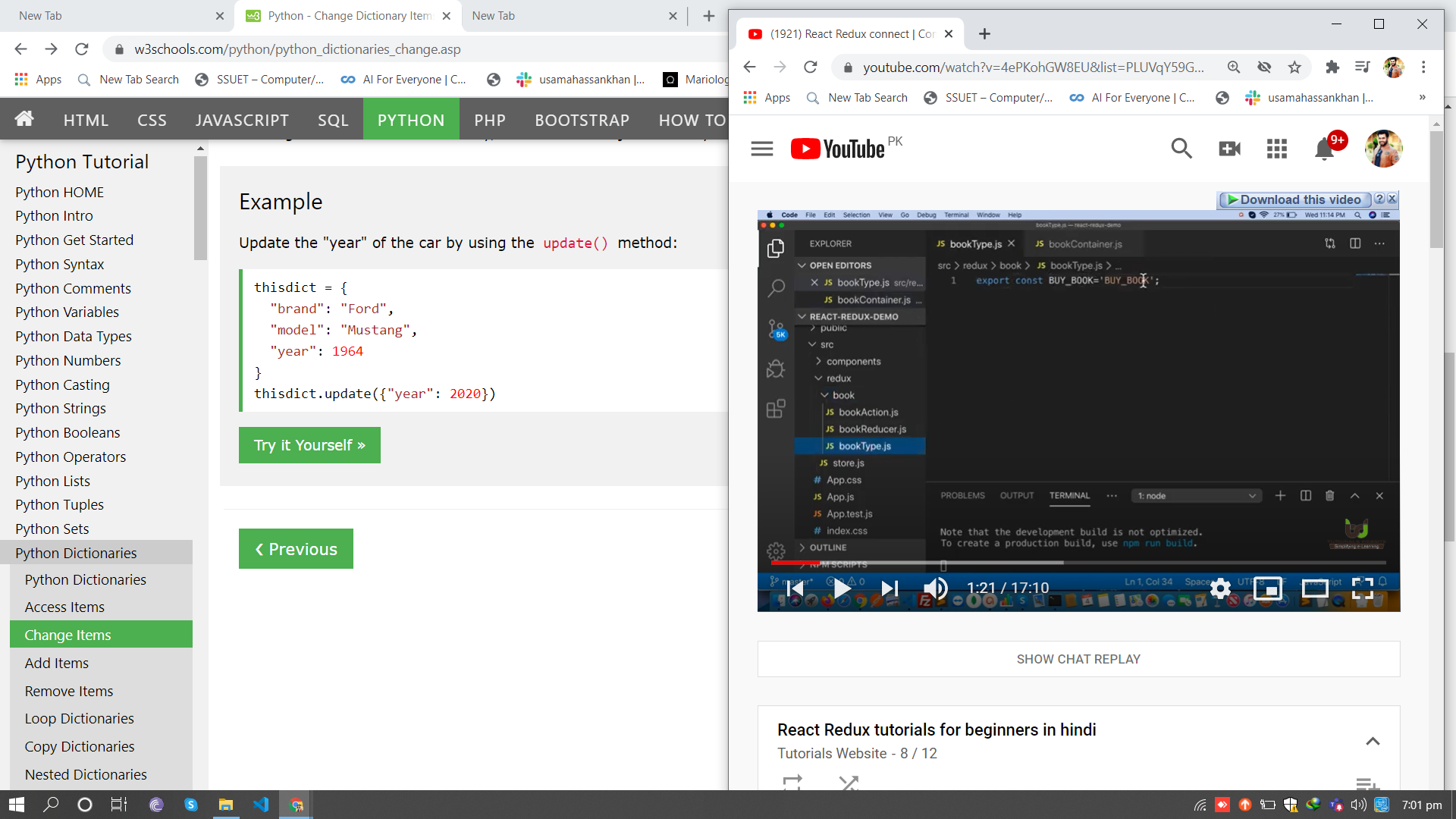
Export default store;

Now wrap app with provider

In app.js

Import {Provider } from ‘react—redux’





Now

In redux folder create index.js

Export {buyBook} from ‘./book/bookAction’;

In component

BookCntainers.js

Import {buyBook} from ‘../redux’/index.js ;//here don’t necessary to write index.js its pick default

Import React from ‘react’

Import {connect } from ‘react-redux’;

Function bookContainer (props){

Return (

<div>

<h1>number of books{props.numberofbooks}</h1>

<button onClick={props.buyBook}>buy book</button>

</div>

)

}

Const mapStatetoProps=(state)=>{

Return {

numberofBooks:state.numberofBooks

}}

Const mapDispatchtoProps=(dispatch)=>{

Return {

buyBook:function(){

dispatch(buyBook());

}}

}

Export default connect(mapStatetoProps,mapDispatchtoProps)(bookContainer);