3 StephenGrider **Modeling Application State**

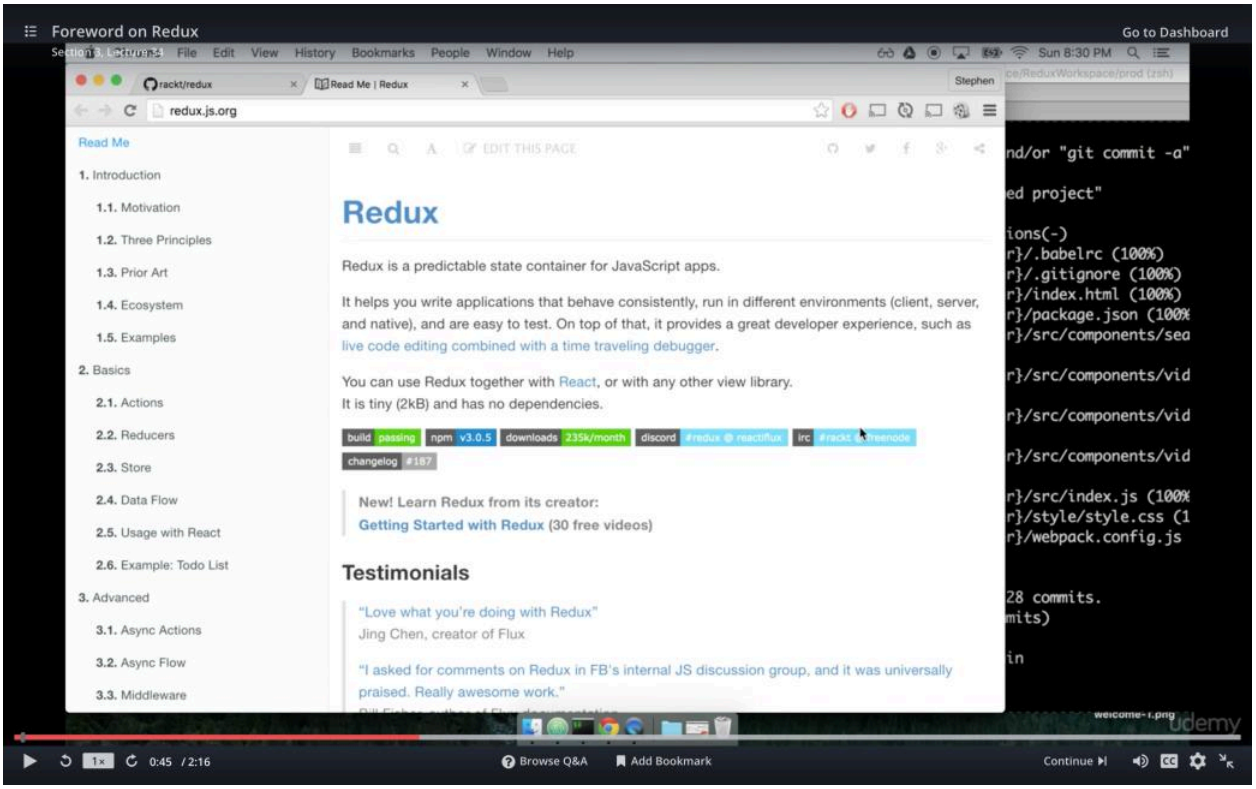
book detail fork
https://codesandbox.io/s/486zn8n5j9

https://github.com/StephenGrider/BookList

Redux predictable state container for JavaScript applications

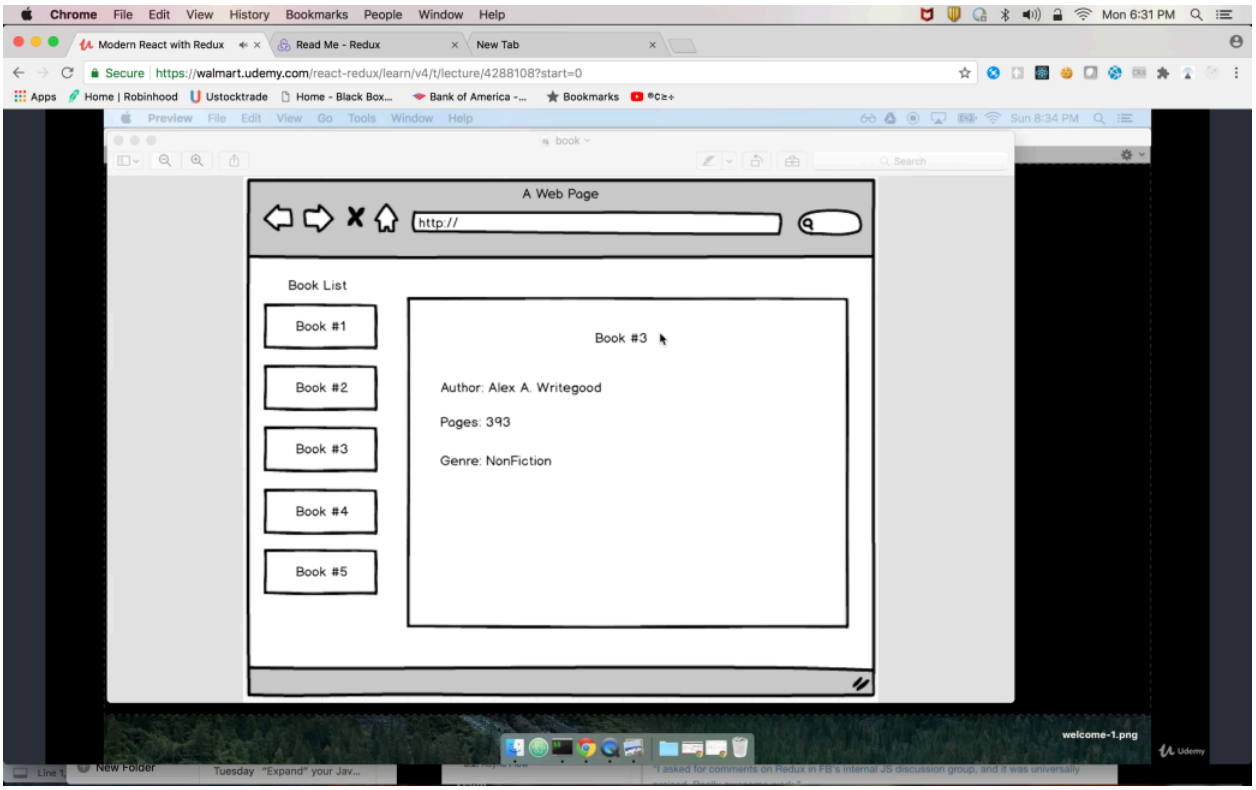
https://redux.js.org/

On top of redux you have
webpack
redux router
react router
redux promise
redux thunk
hot reloading



show list of books on left hand side

if use clicks on any book it shows details

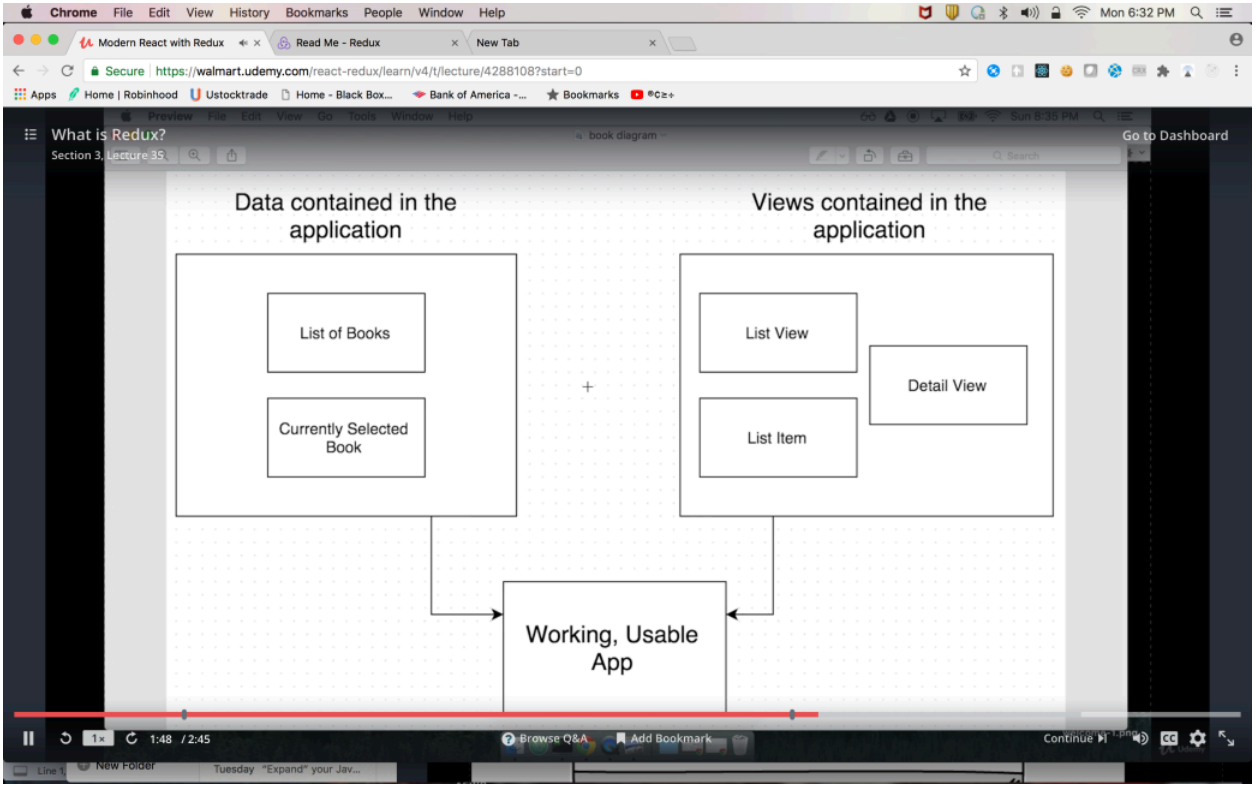


first separate data and view

Redux is data container (State Container) in application box
React is view container in application box]

Redux centralized all the application data in single object, which referred to state

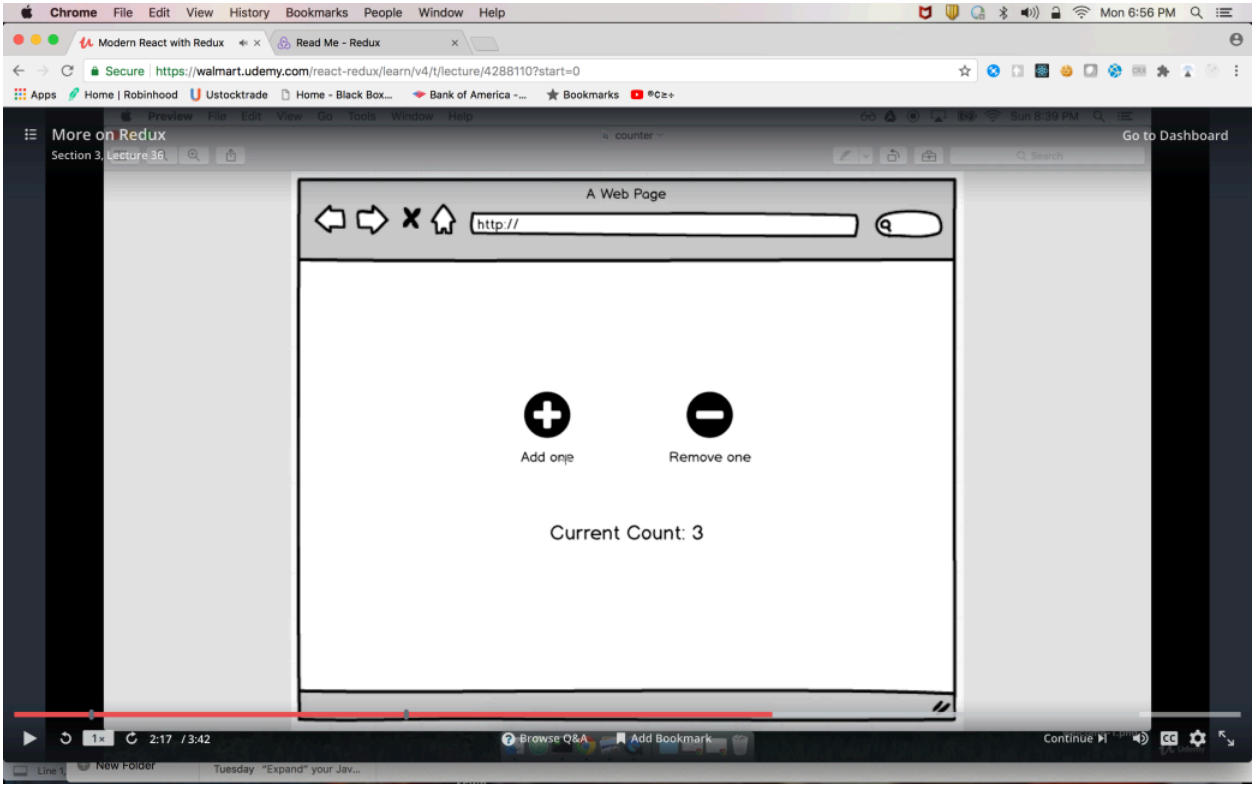
Redux maintain application level state
React view level state



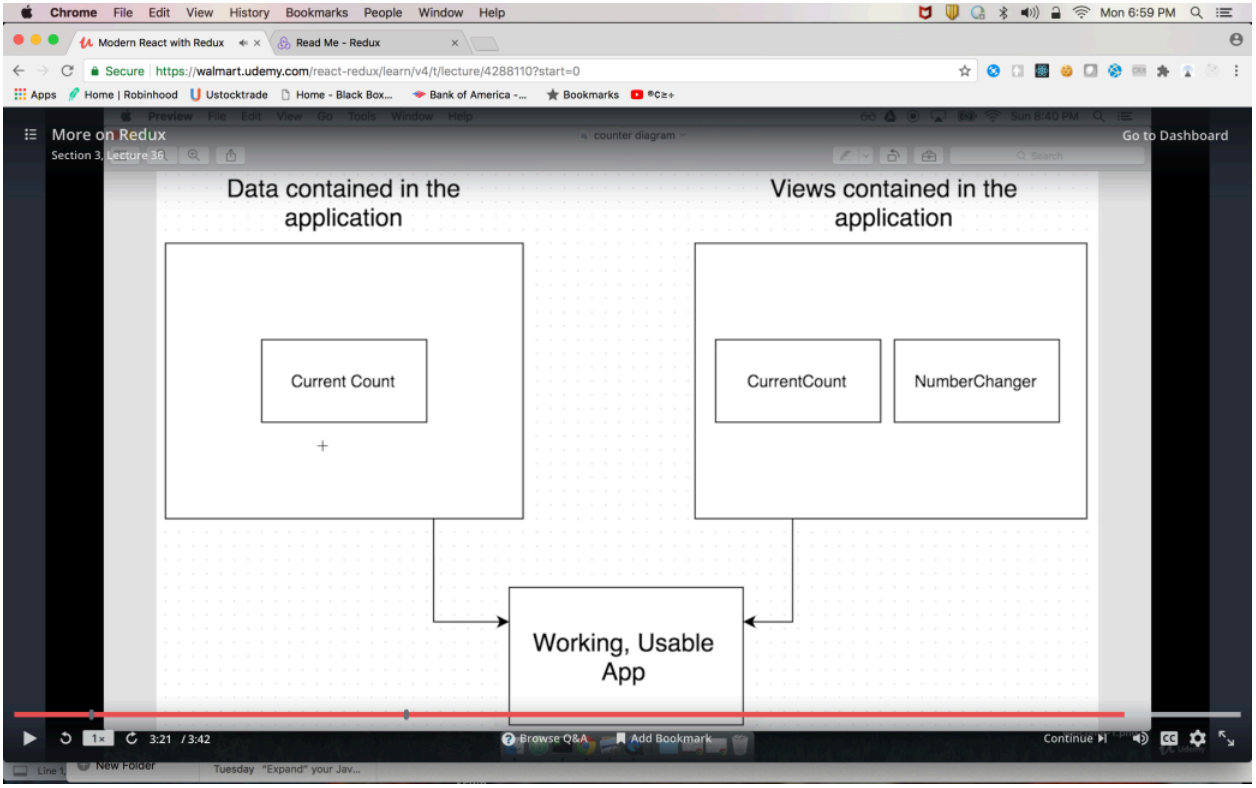
In other javascript lib they have separate collections of data.
backbone has collections, flux has stores

Example of counter application

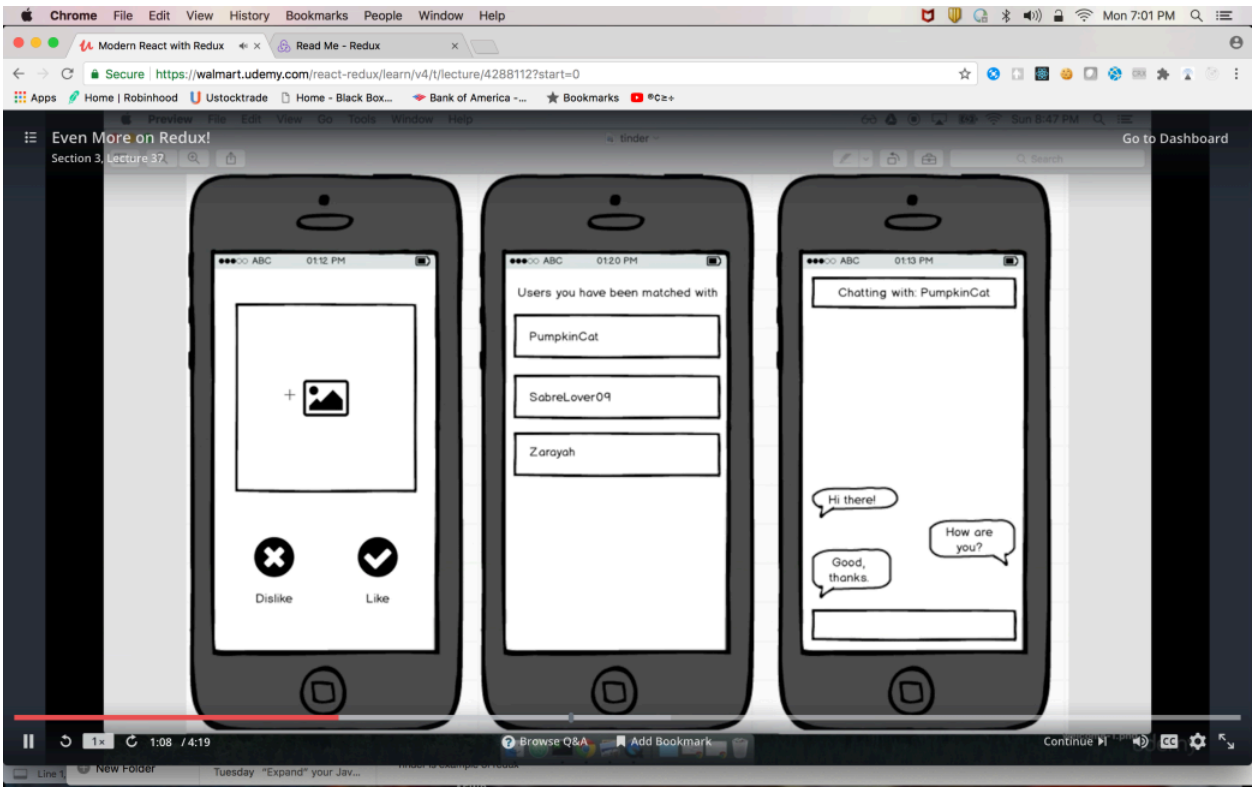
at bottom counter start at 0
when user press + plus button number goes up by 1
when user press - plus button number goes down by 1

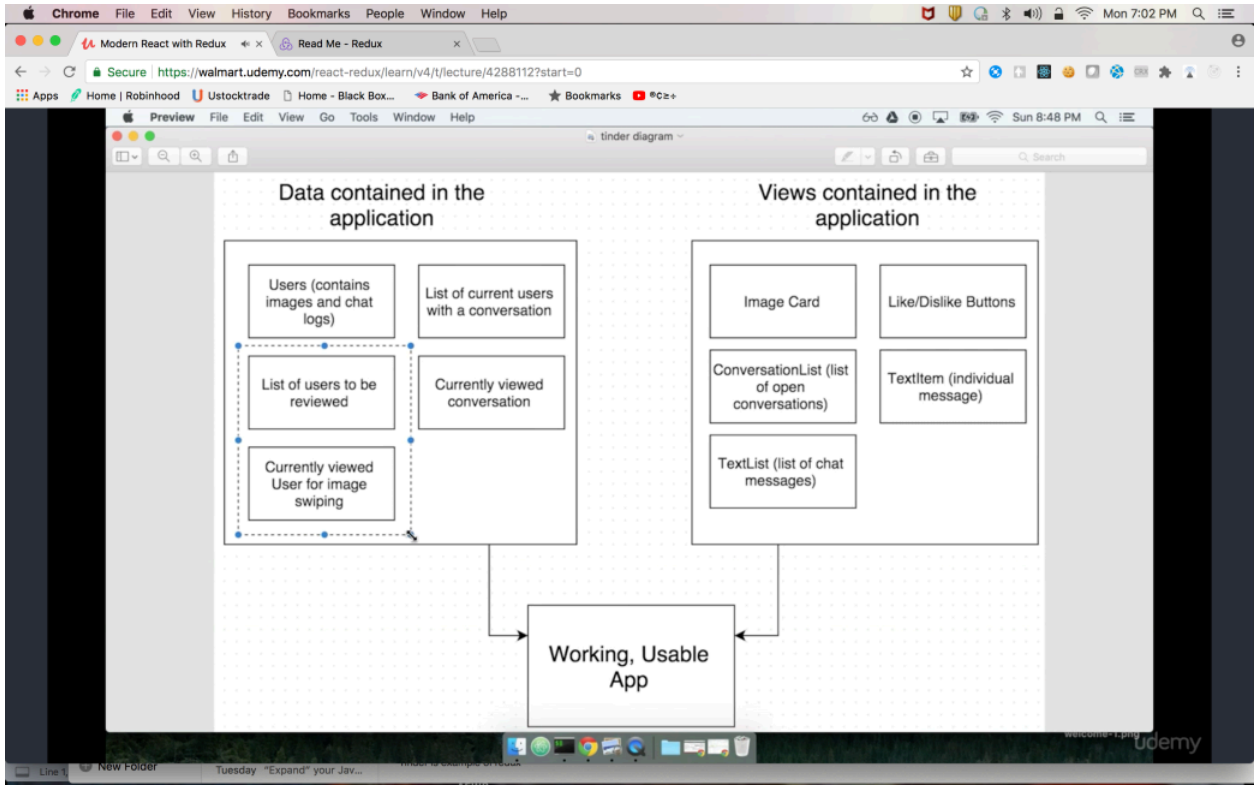


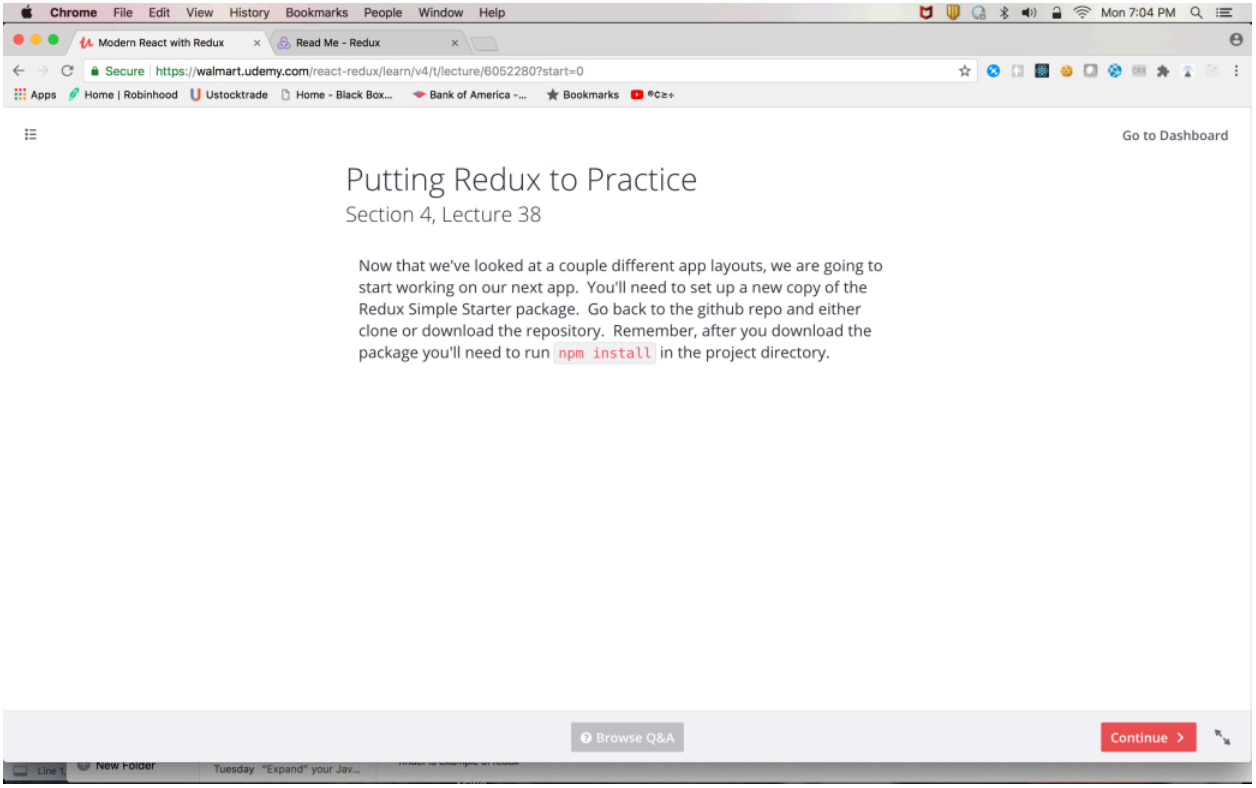
React will be showing 2 components
Redux will maintain state of counter



Tinder is example of redux



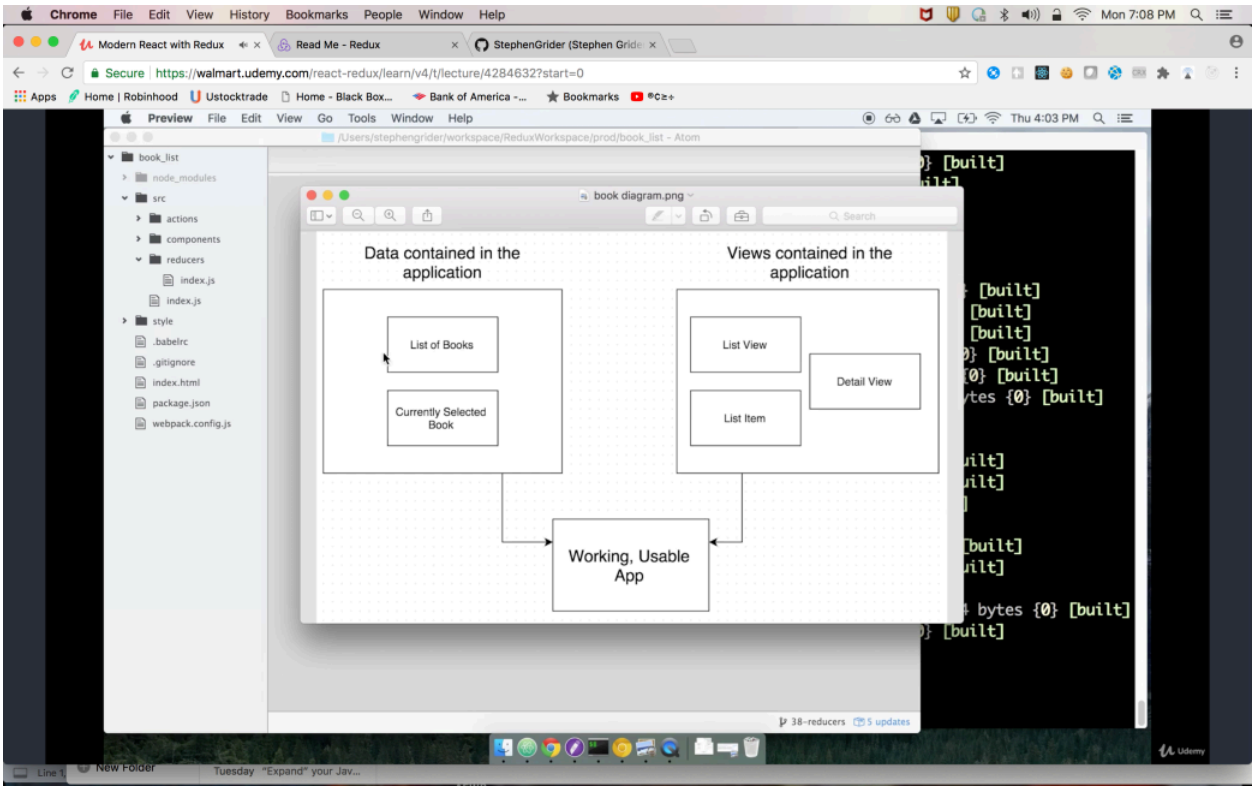

Data contained in the application
Users (contains images and chat logs)
List of current users with a conversation
List of users to be reviewed
Currently viewed conversation
Currently viewed User for image swiping
Views contained in the application
Image Card
Like/Dislike Buttons
ConversationList (list of open conversations)
TextItem (individual message)
TextList (list of chat messages)
Working, Usable App




```
import React from "react";
import ReactDOM from "react-dom";

import { Provider } from "react-redux";
import { createStore, applyMiddleware } from "redux";

import reducers from "./reducers";

import App from "./components/app";

import "./styles/base.css";

const createStoreWithMiddleware = applyMiddleware()(createStore)(reducers);

ReactDOM.render(
<Provider store={createStoreWithMiddleware}>
<App />
</Provider>,
document.getElementById("root")
);
```

Reducer is function that returns the piece of applications state

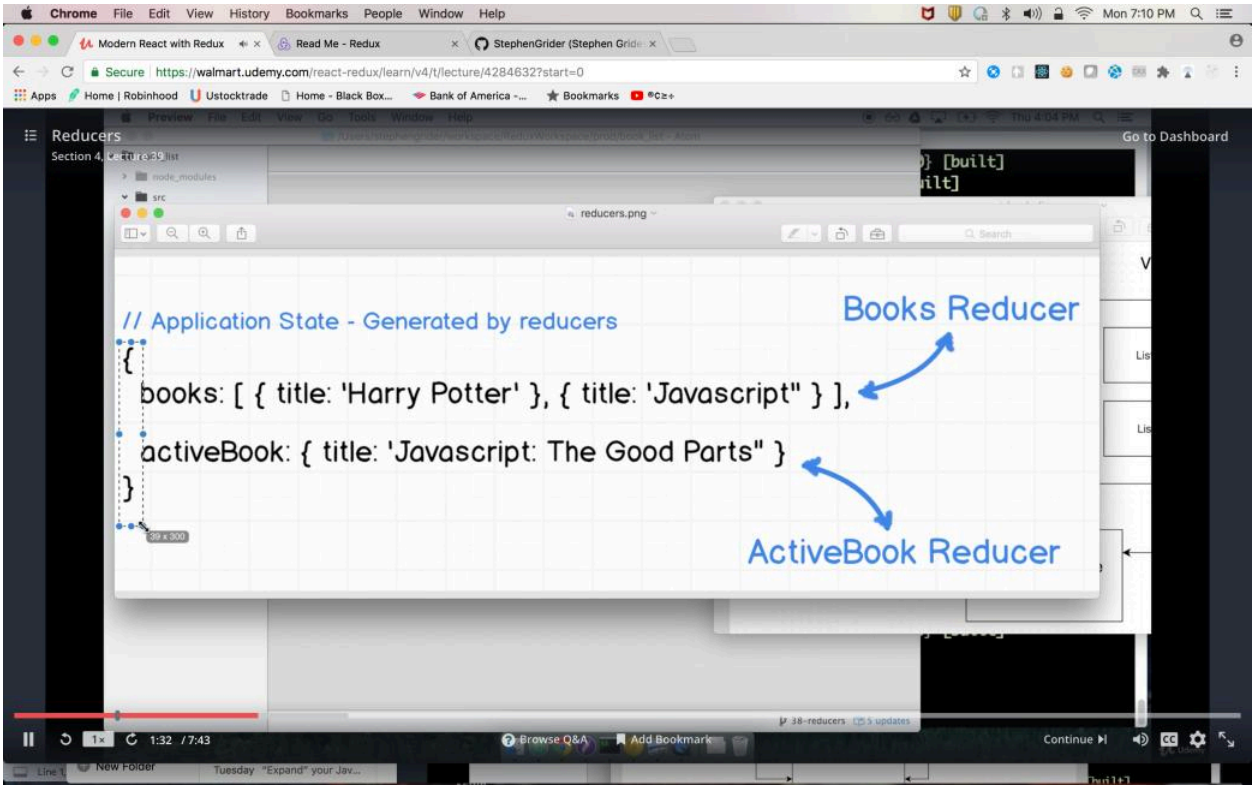
since our application has 2 pieces of state
we will need 2 reducers in below example



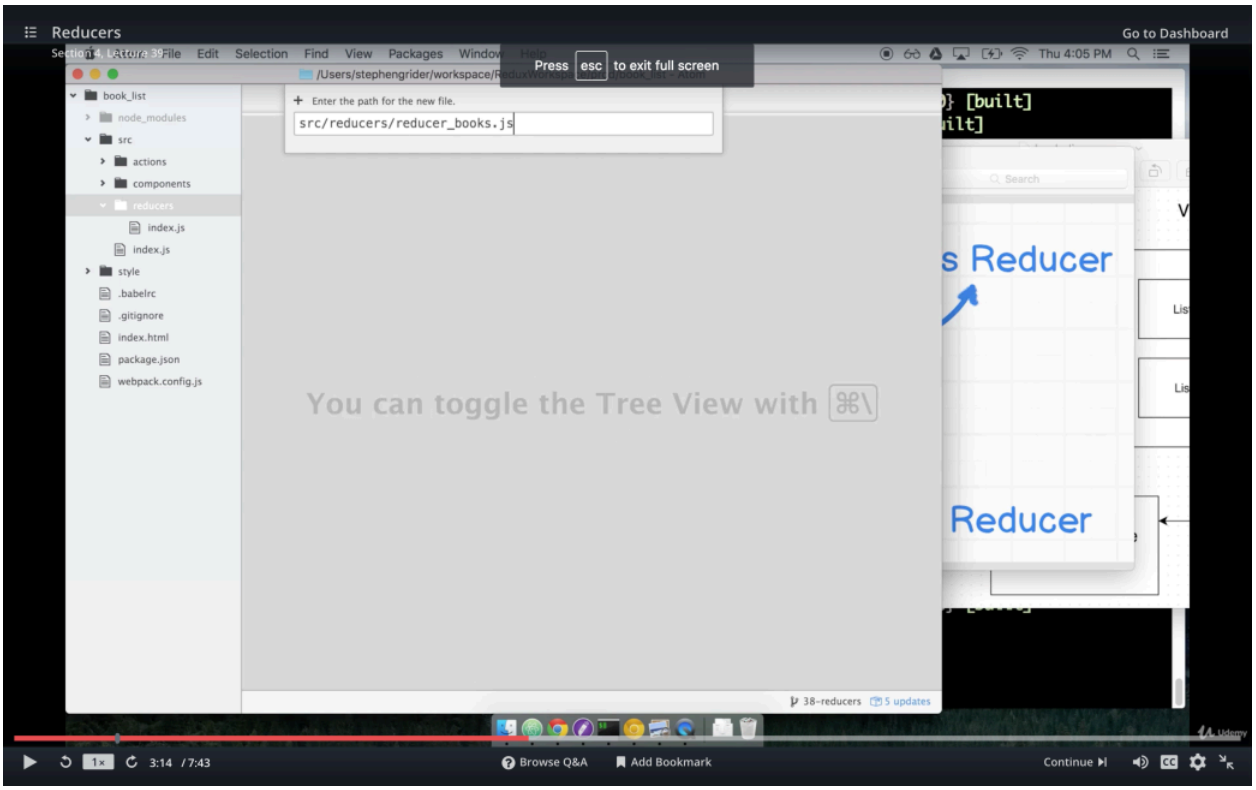
Application state is a pure javascript object

it has 2 keys
2 pieces of state

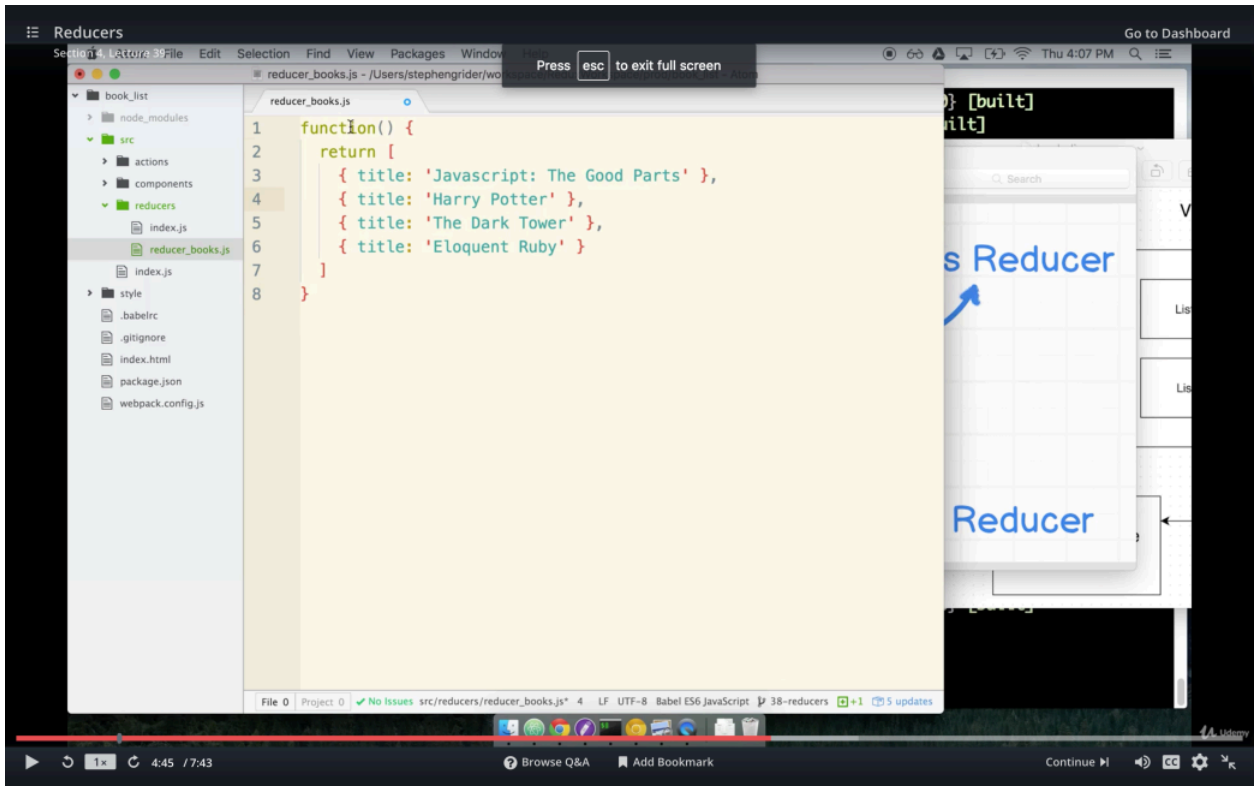
2 Reducers will produce value of state



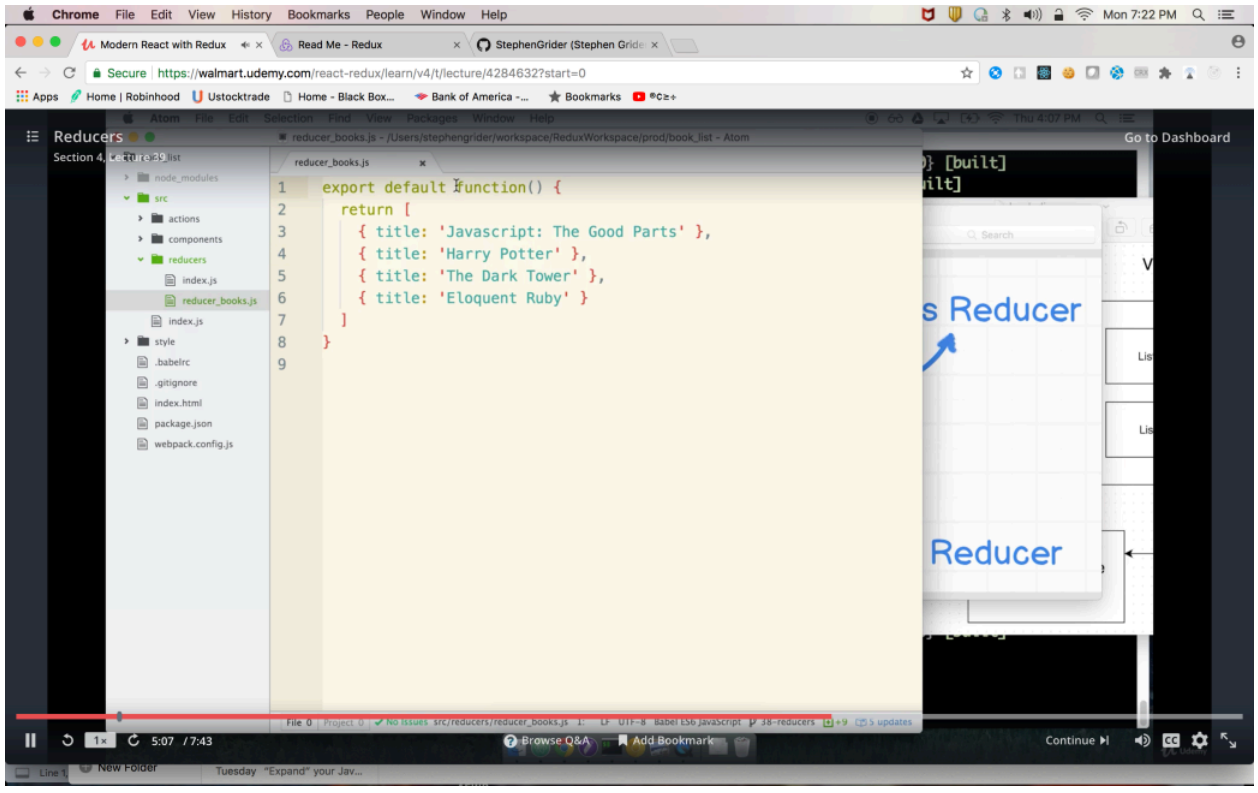
create a reducer
reducer_books.js



start with simple function
that just return array of books



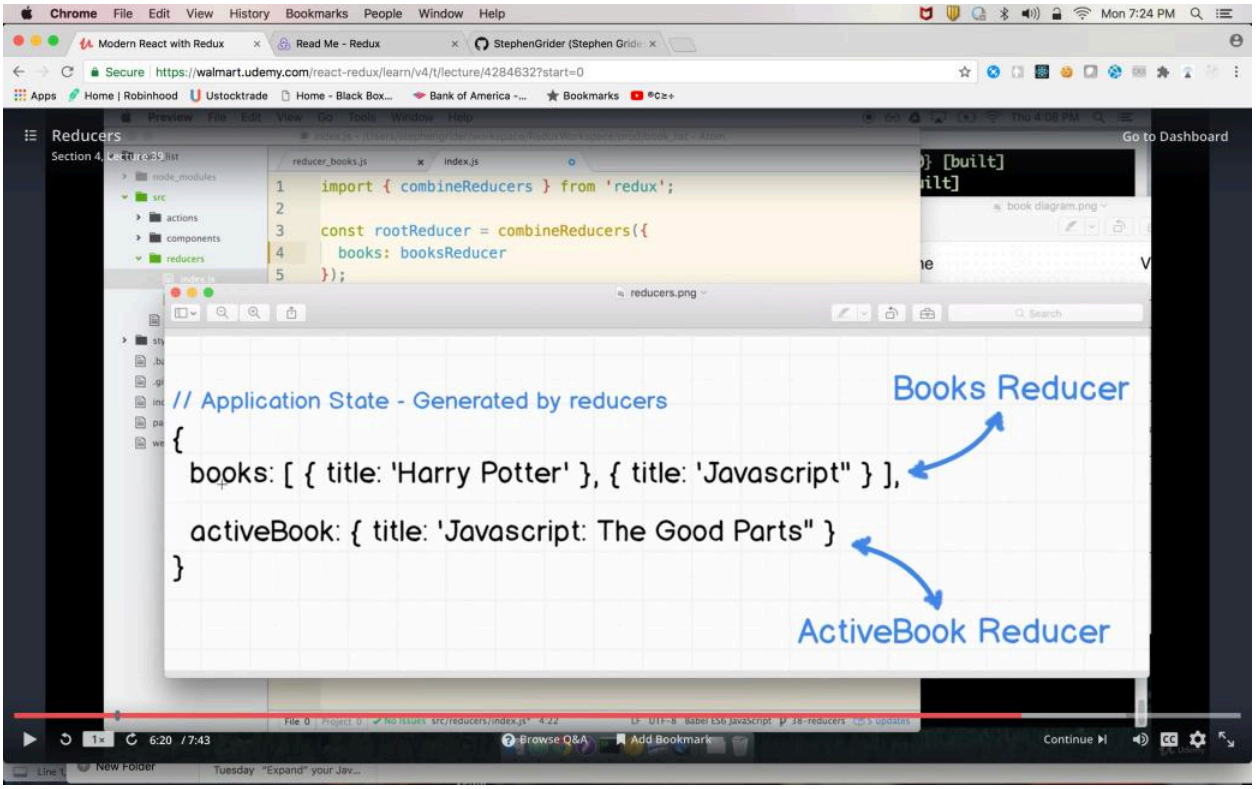
export the function
so it can use used any where else in project



now wire this to our application
we will do into index.js
where we import combineReducers from ‘redux’

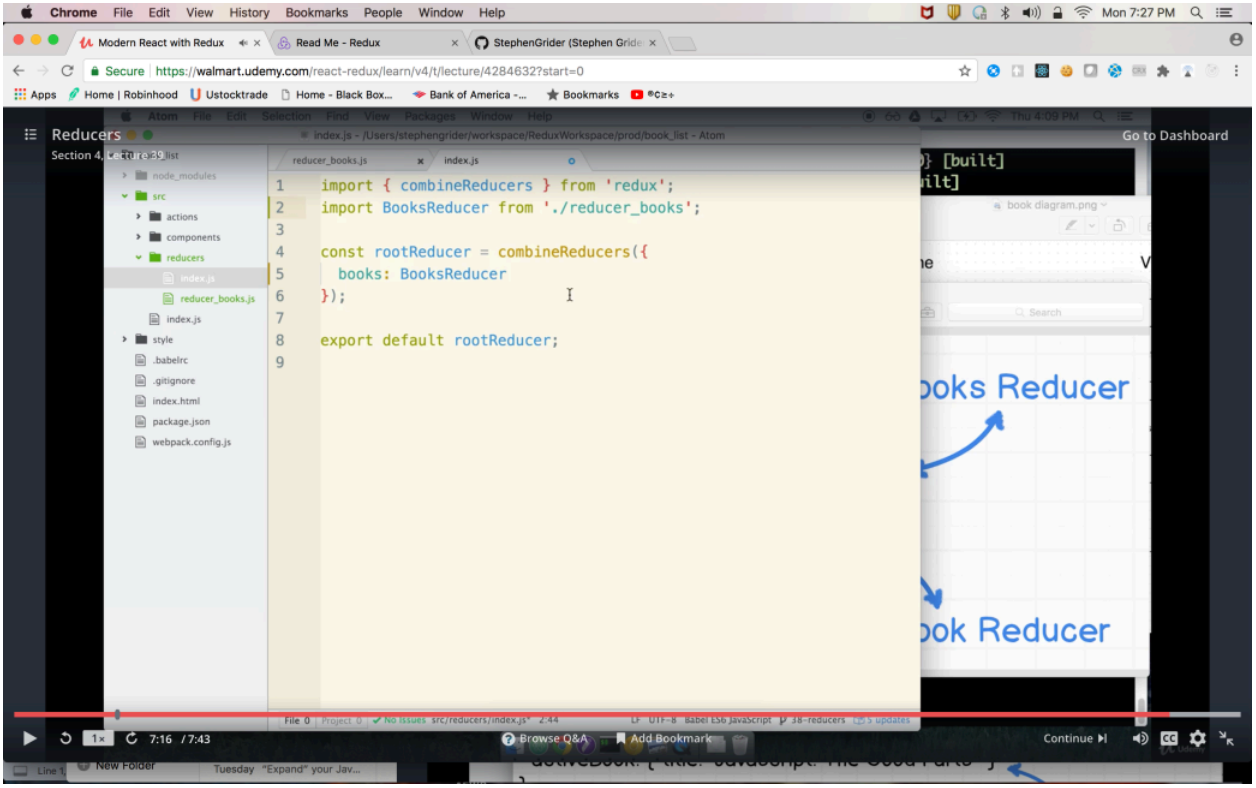
combineReducers({state: reducer})
in combineReducers function we will pass a object {state: reducer}

we will really map the state to reducer
books is state
booksReducer is reducer

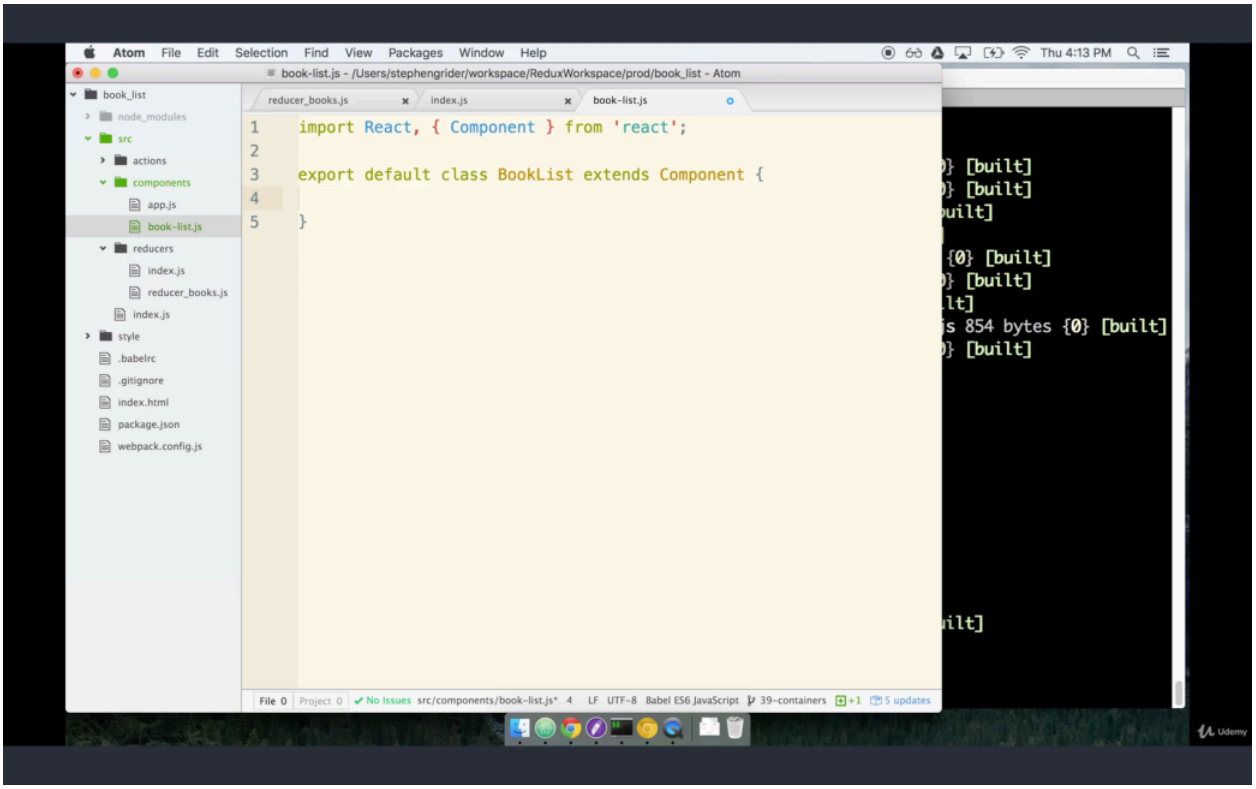


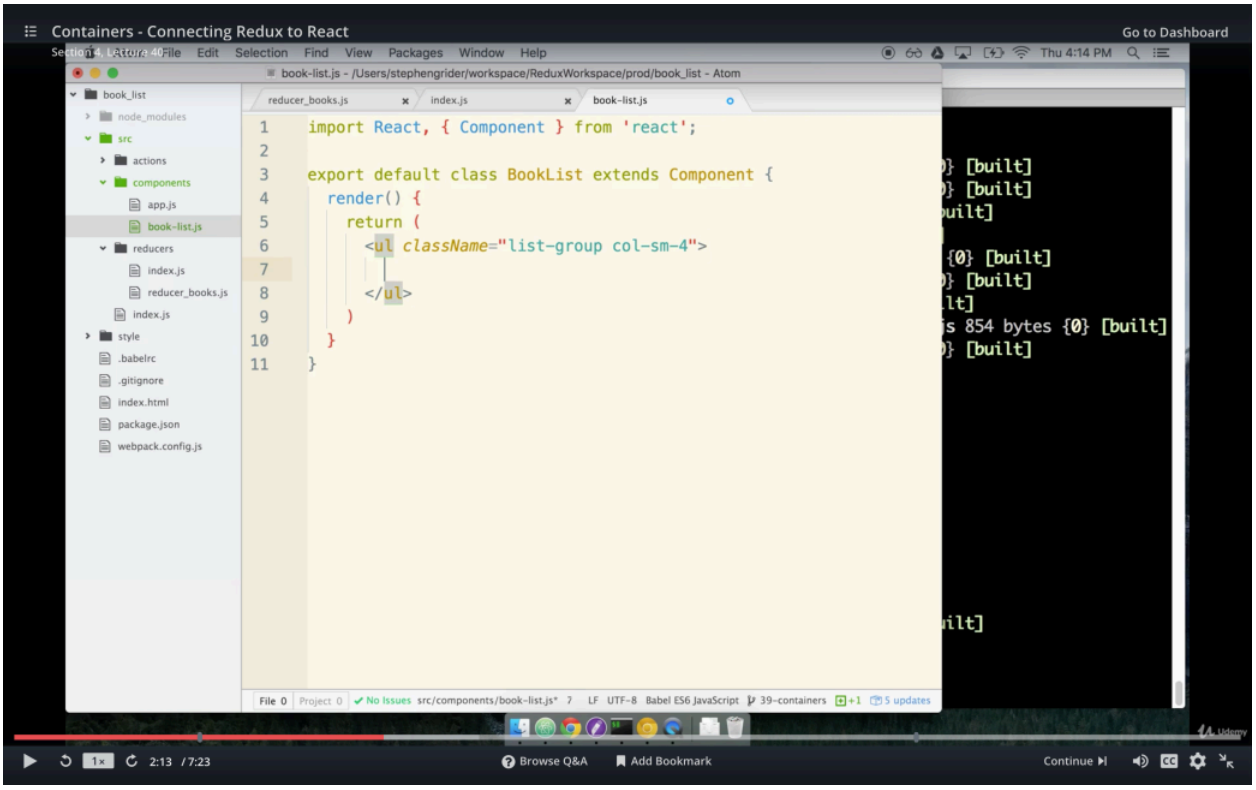
import the reducer_books

and capitalize the BooksReducers



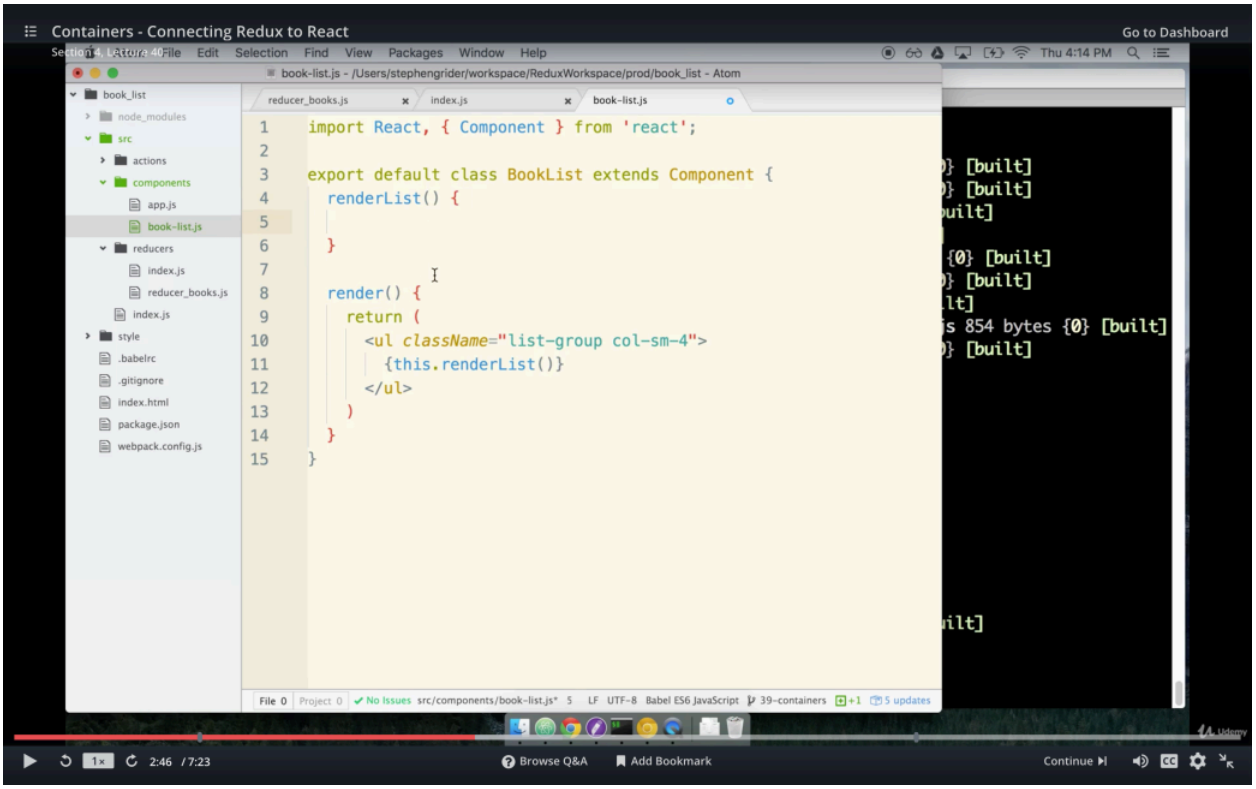
To test this we will create book-list component in React



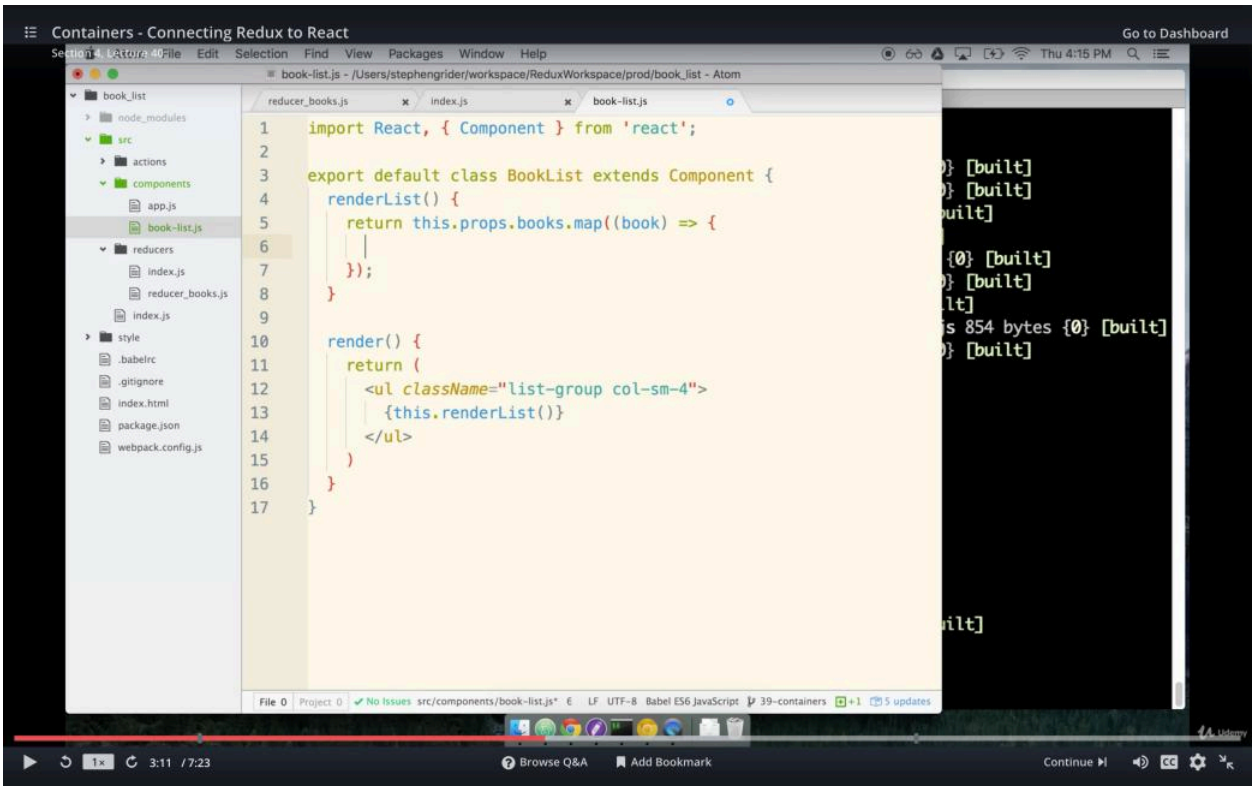


if want to call a separate javascript function in jsx
we will wrap in curly braces

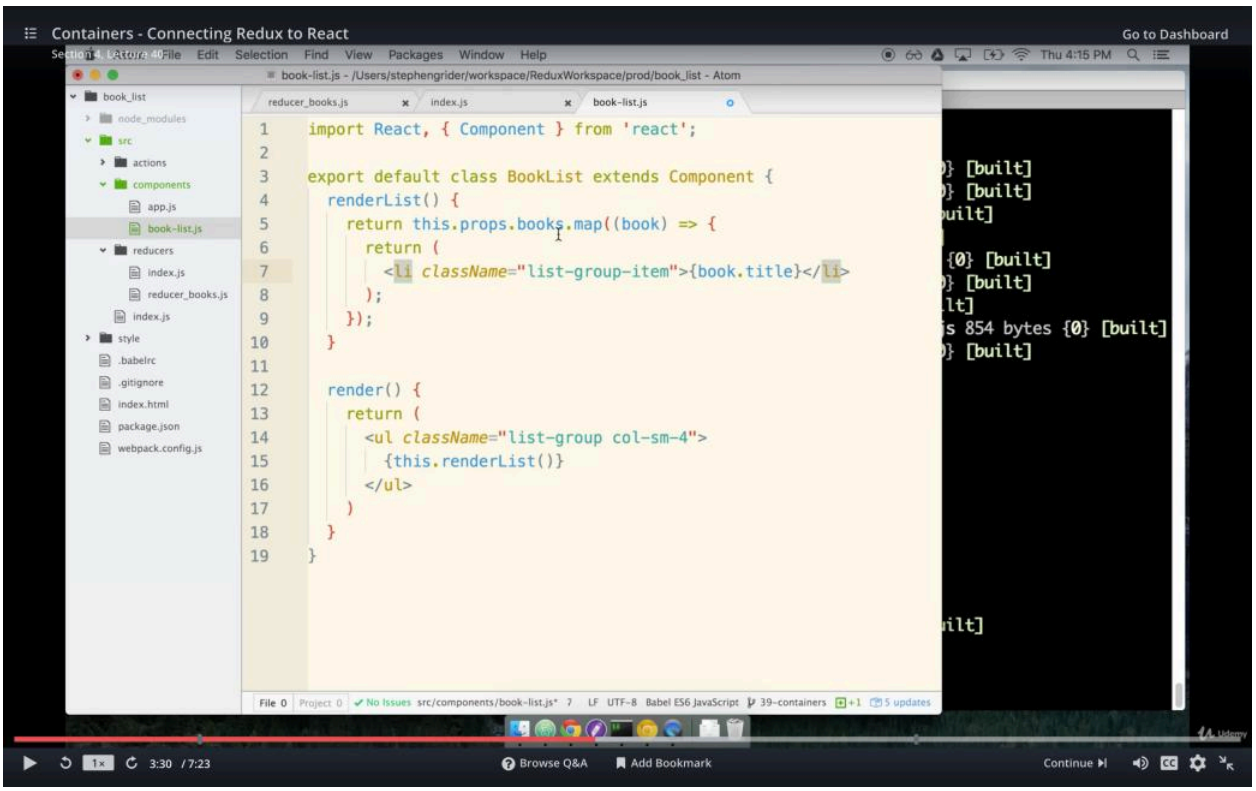
{this.renderList()}



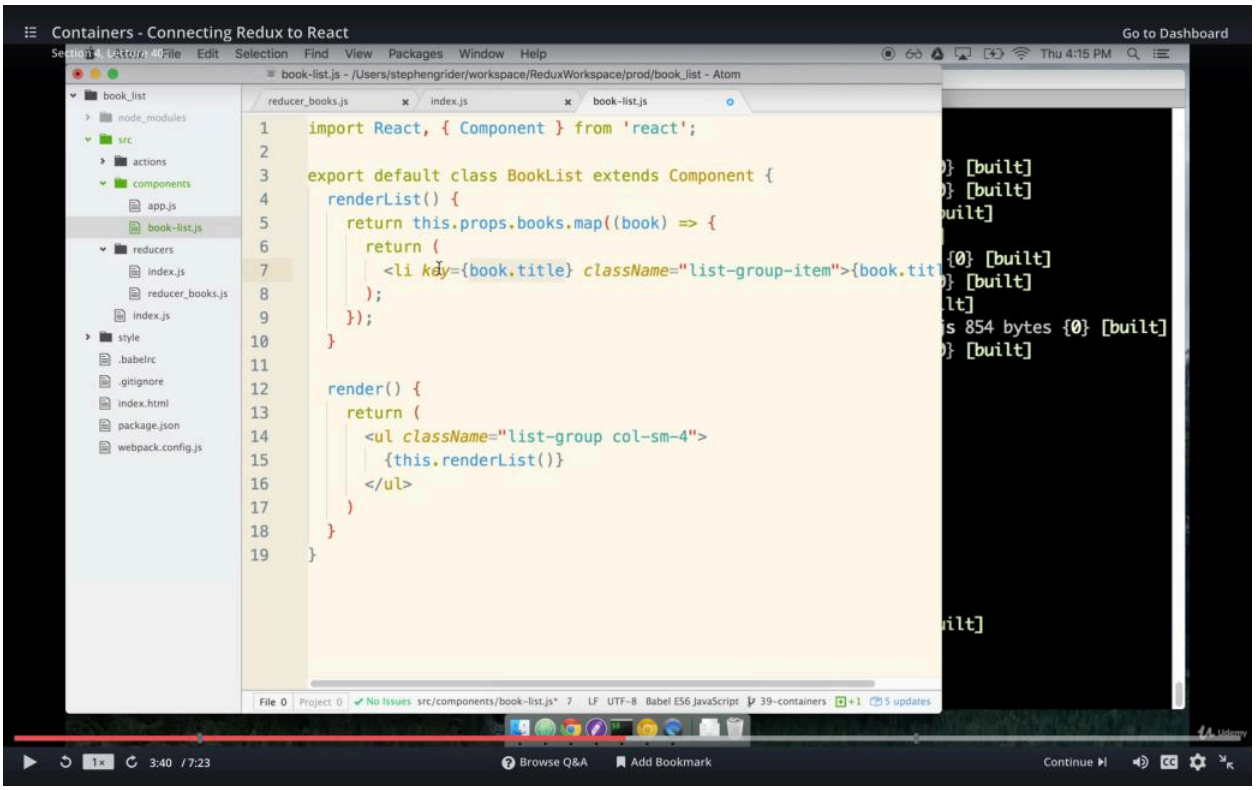
we will assume this.props will return books array
map over that array
for each element in array we will return a li.
don’t forget to add key because it is a list . we will title to give unique value to keys


Containers - Connecting Redux to React
Go to Dashboard
book-list.js - /Users/stephengrider/workspace/ReduxWorkspace/prod/book_list - Atom
reducer_books.js
index.js
book-list.js
import React, { Component } from 'react';
export default class BookList extends Component {
renderList() {
return this.props.books.map((book) => {
});
}
render() {
return (
<ul className="list-group col-sm-4">
{this.renderList()}

)
}
}
3:11 / 7:23
Browse Q&A
Add Bookmark
Continue


Containers - Connecting Redux to React
Go to Dashboard
book-list.js - /Users/stephengrider/workspace/ReduxWorkspace/prod/book_list - Atom
reducer_books.js
index.js
book-list.js
import React, { Component } from 'react';
export default class BookList extends Component {
renderList() {
return this.props.books.map((book) => {
return (
<li className="list-group-item">{book.title}
);
});
}
render() {
return (
<ul className="list-group col-sm-4">
{this.renderList()}

)
}
}
3:30 / 7:23
Browse Q&A
Add Bookmark
Continue



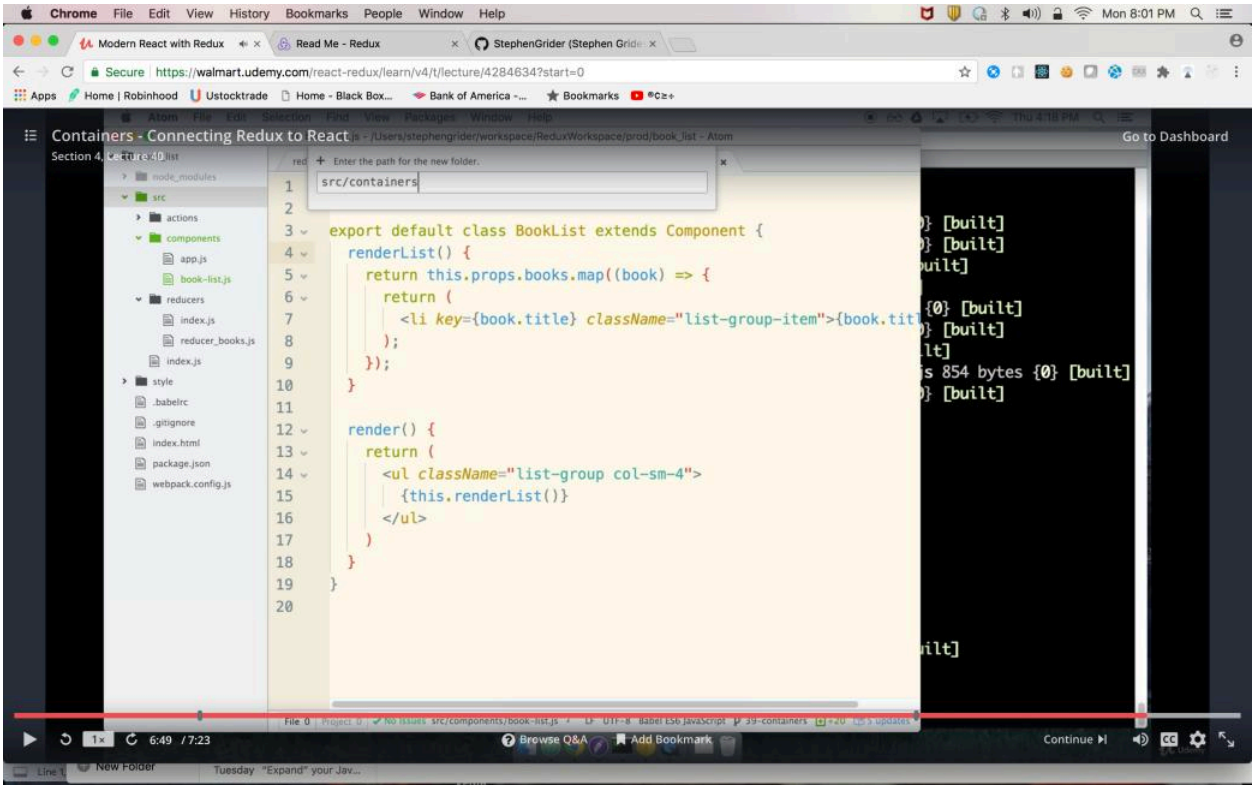
now we need to connect
react views with redux state

connecting react and redux lib done with separate lib react-redux

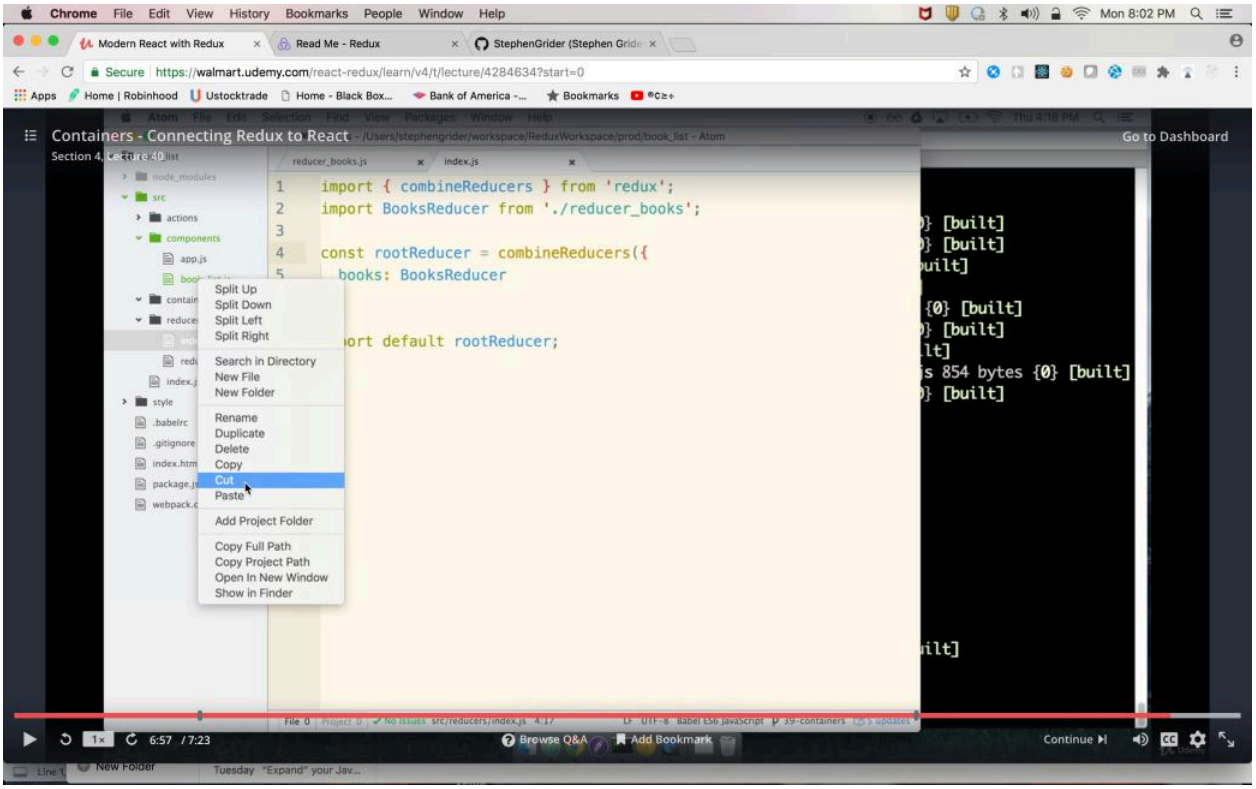
we will define one of our components as container

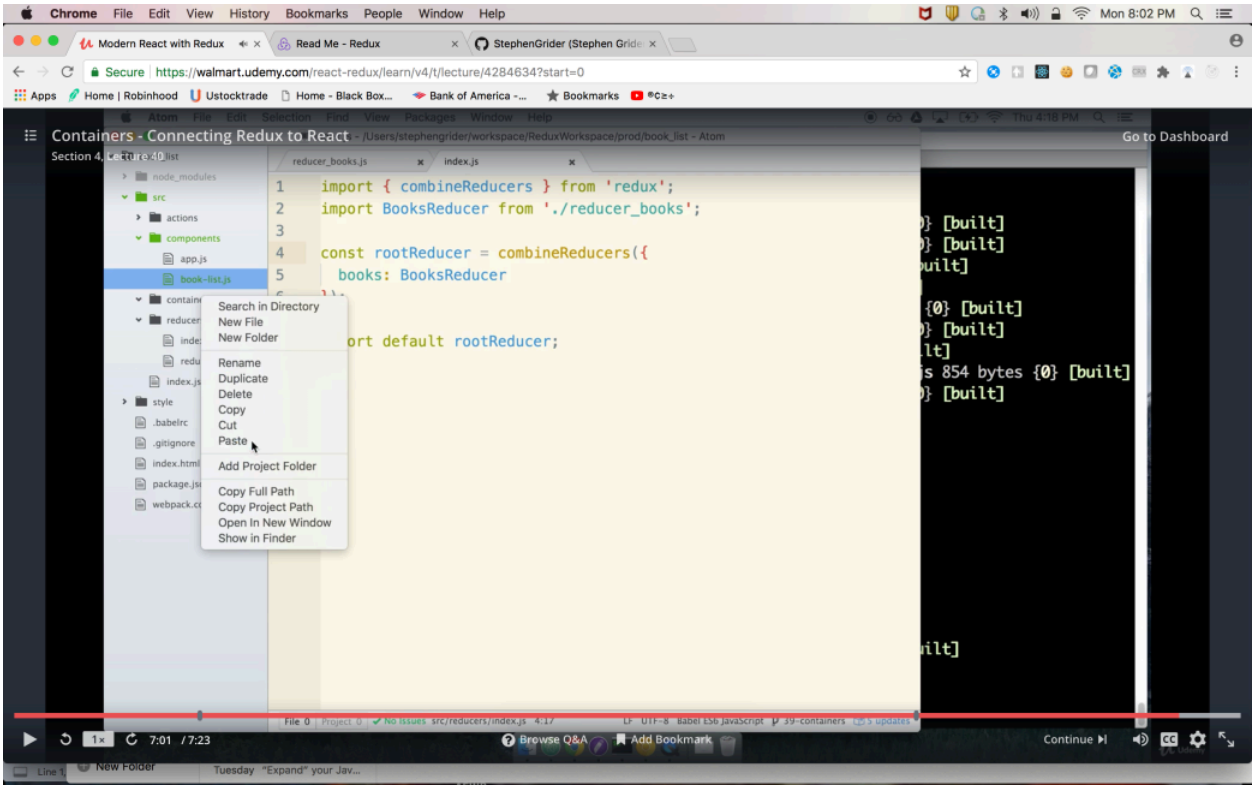
A Container is react component that has a direct connection with state manage by redux

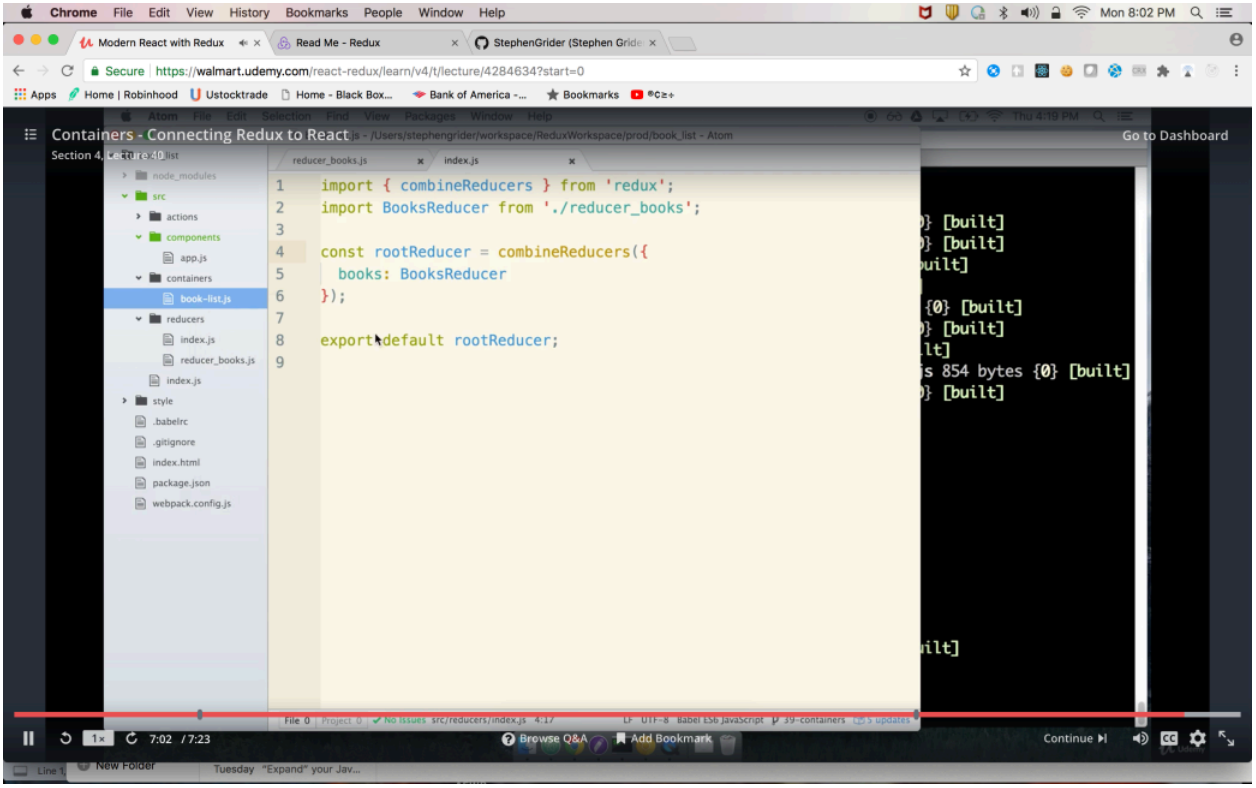
we will put all containers in a separate dir calls Conainer



cut our book-list and from components and past it containers

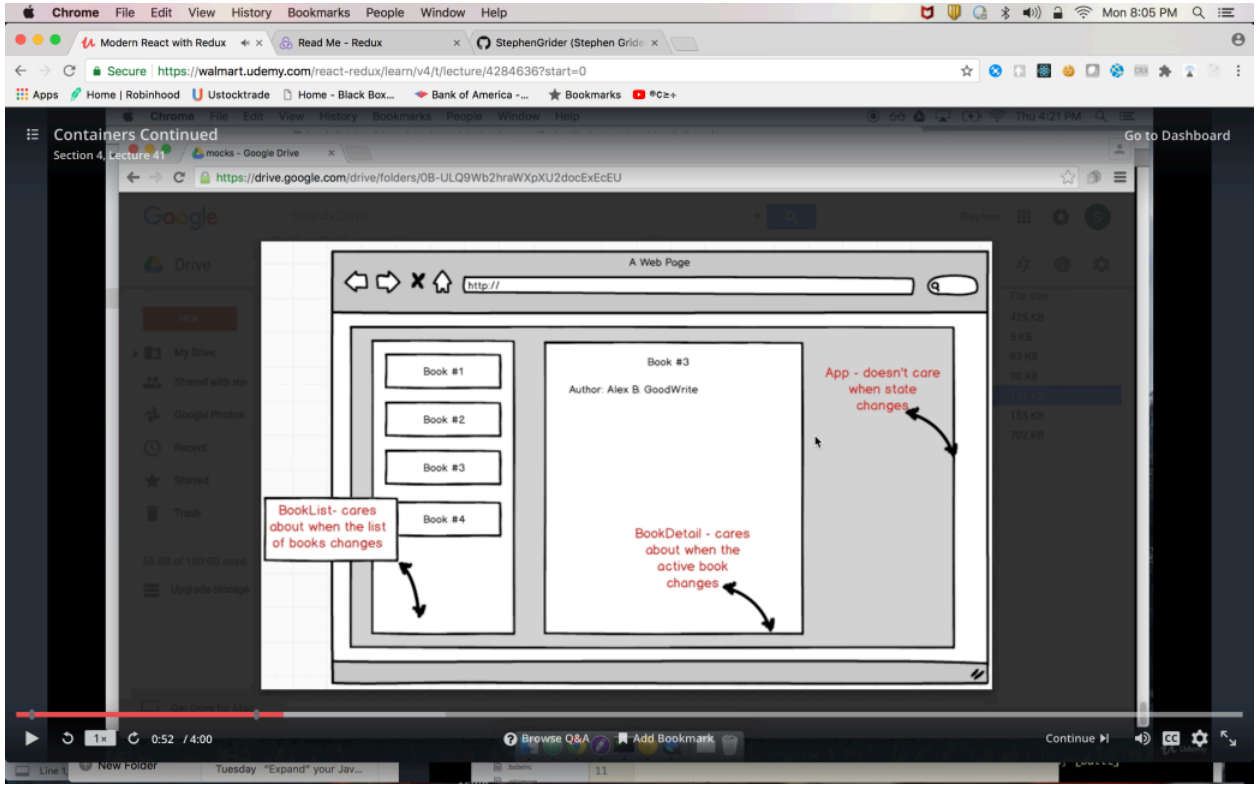

Containers - Connecting Redux to React
Go to Dashboard
import { combineReducers } from 'redux';
import BooksReducer from './reducer_books';
const rootReducer = combineReducers({
books: BooksReducer
Split Up
Split Down
Split Left
Split Right
Search in Directory
New File
New Folder
Rename
Duplicate
Delete
Copy
Cut
Paste
Add Project Folder
Copy Full Path
Copy Project Path
Open In New Window
Show in Finder
6:57 / 7:23
Browse Q&A
Add Bookmark
Continue


Containers - Connecting Redux to React
Go to Dashboard
import { combineReducers } from 'redux';
import BooksReducer from './reducer_books';
const rootReducer = combineReducers({
books: BooksReducer
Search in Directory
New File
New Folder
Rename
Duplicate
Delete
Copy
Cut
Paste
Add Project Folder
Copy Full Path
Copy Project Path
Open In New Window
Show in Finder
7:01 / 7:23
Browse Q&A
Add Bookmark
Continue

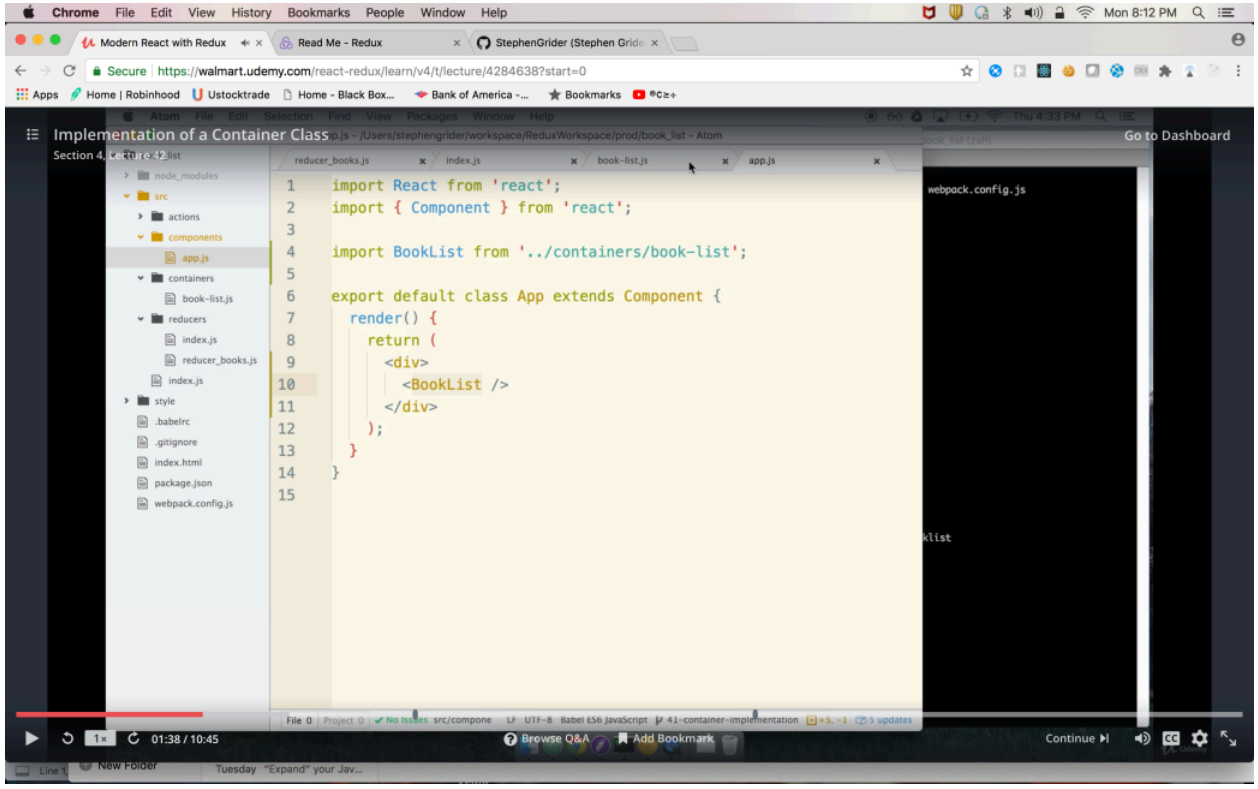


Now q : which component we want to move to container
in general we keep most parent component in container

App does

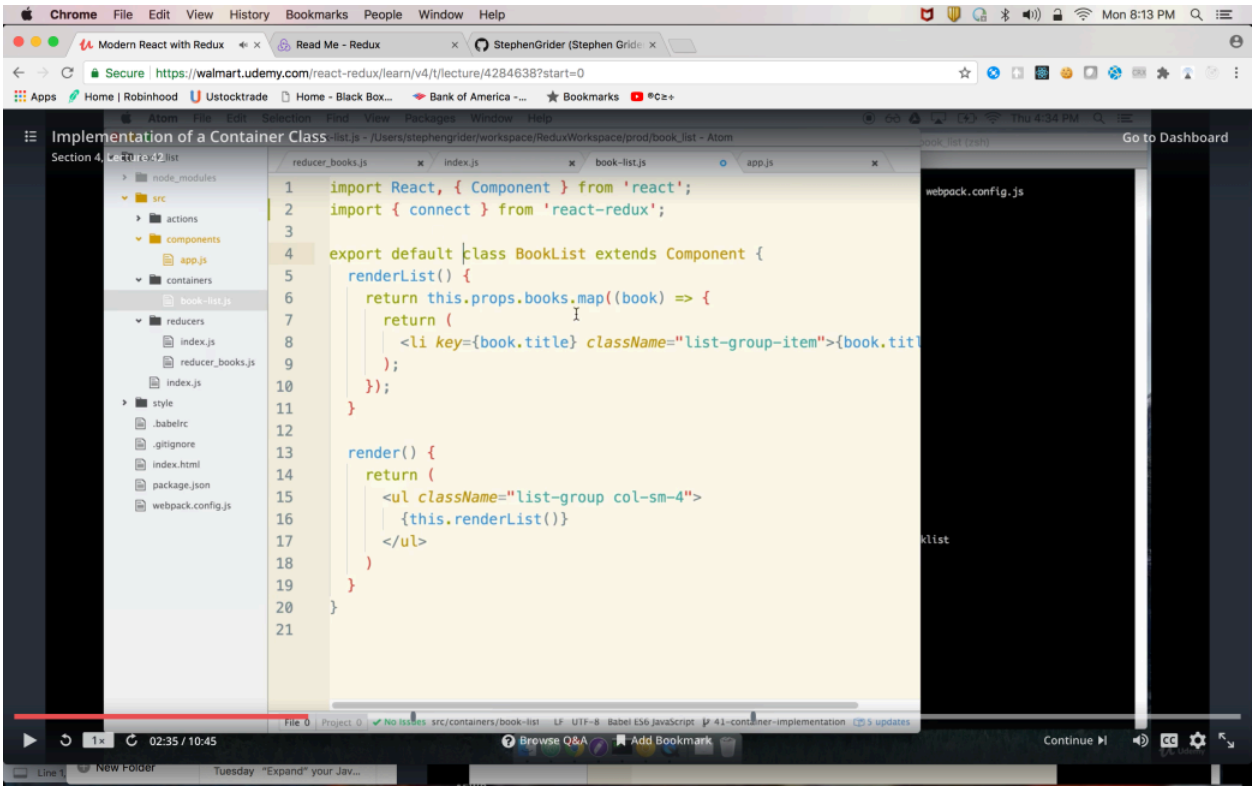


Now first add BookList to app.js

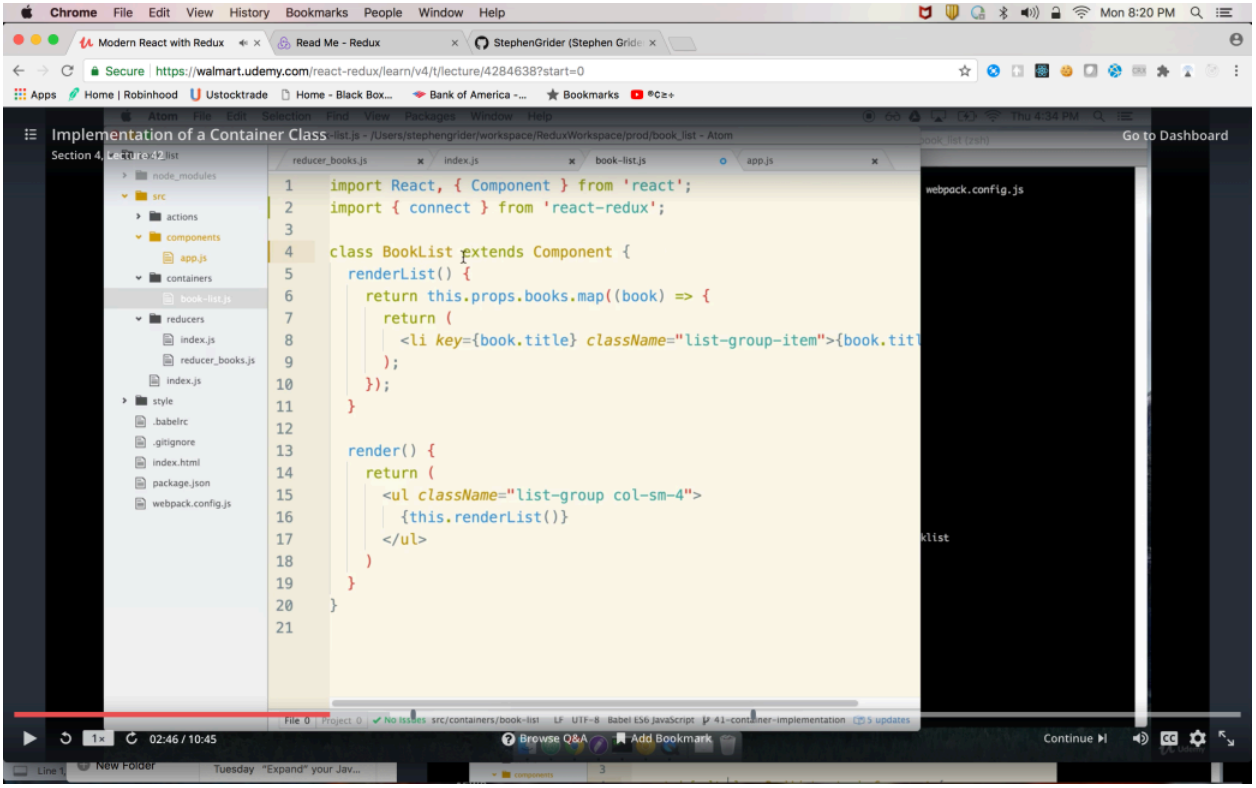


```
import queryString from "query-string";
import extend from "lodash/extend";
import isEmpty from "lodash/extend";
import Promise from "bluebird";

import { fetchJSON } from "@walmart/electrode-fetch";
import {
  transformItemBasePayload,
  transformCustomerBasePayload
} from "../helpers/apollo/apollo-helper";
```

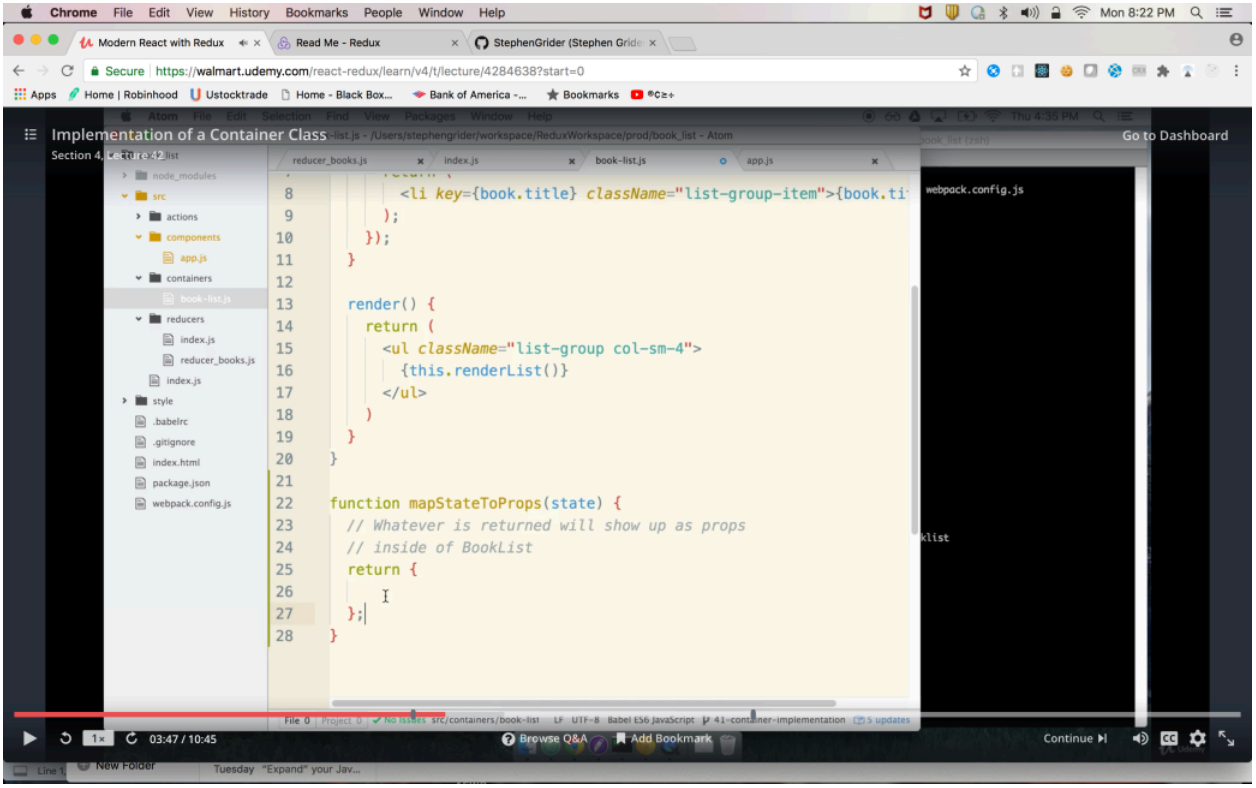
{connect} import a single property from ‘react-redux’
if import a word like **React** it imports **entire object** from file

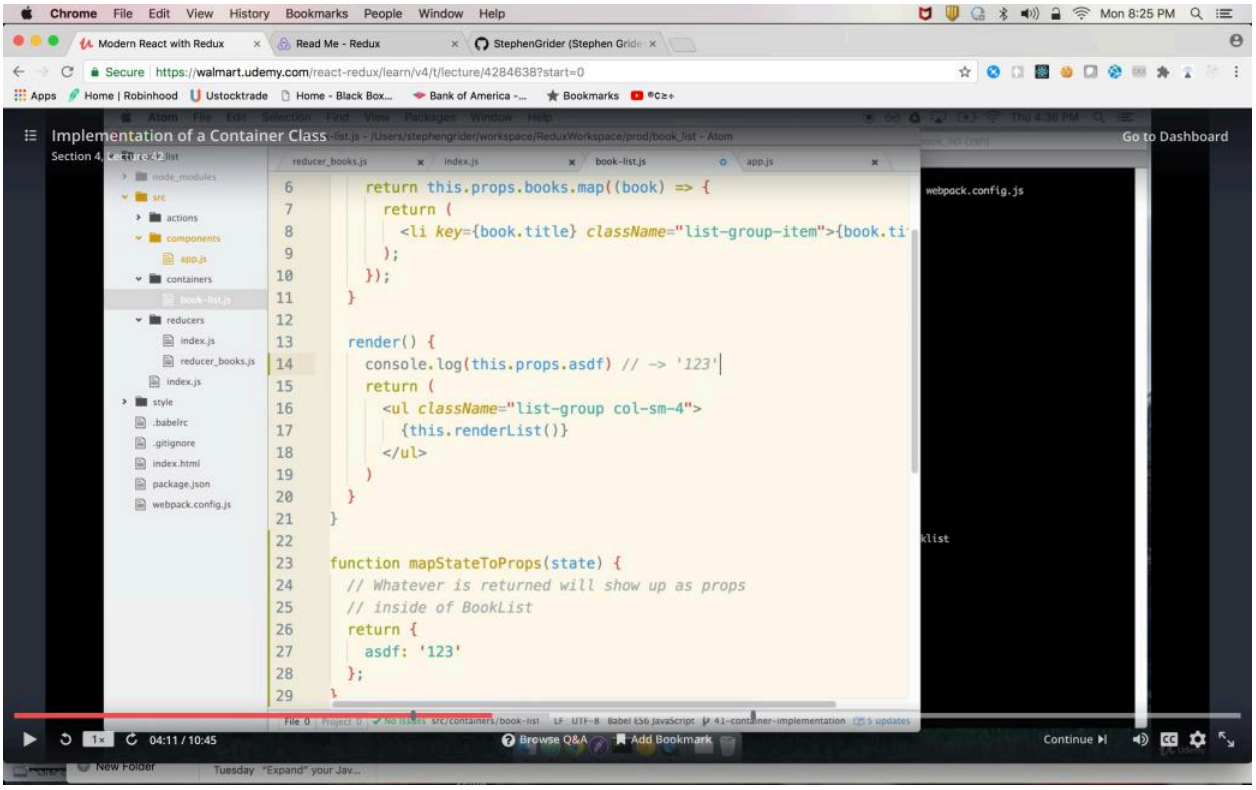


remove export default from class



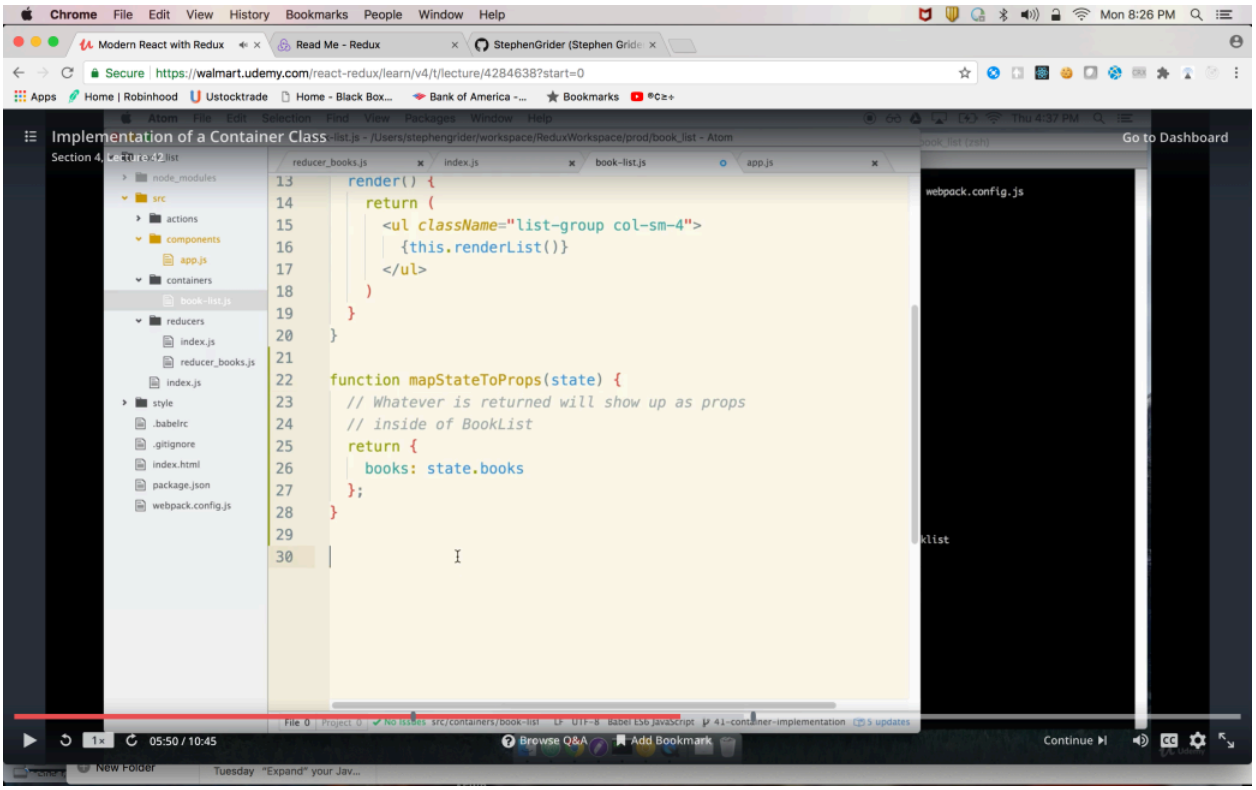
outside of class we will **define** a function **mapStateToProps**
it will have one argument that is state
and it returns a object
whatever is return as object from **mapStateToProps** it
will show up in props inside BookList class





since we are using this.props.books

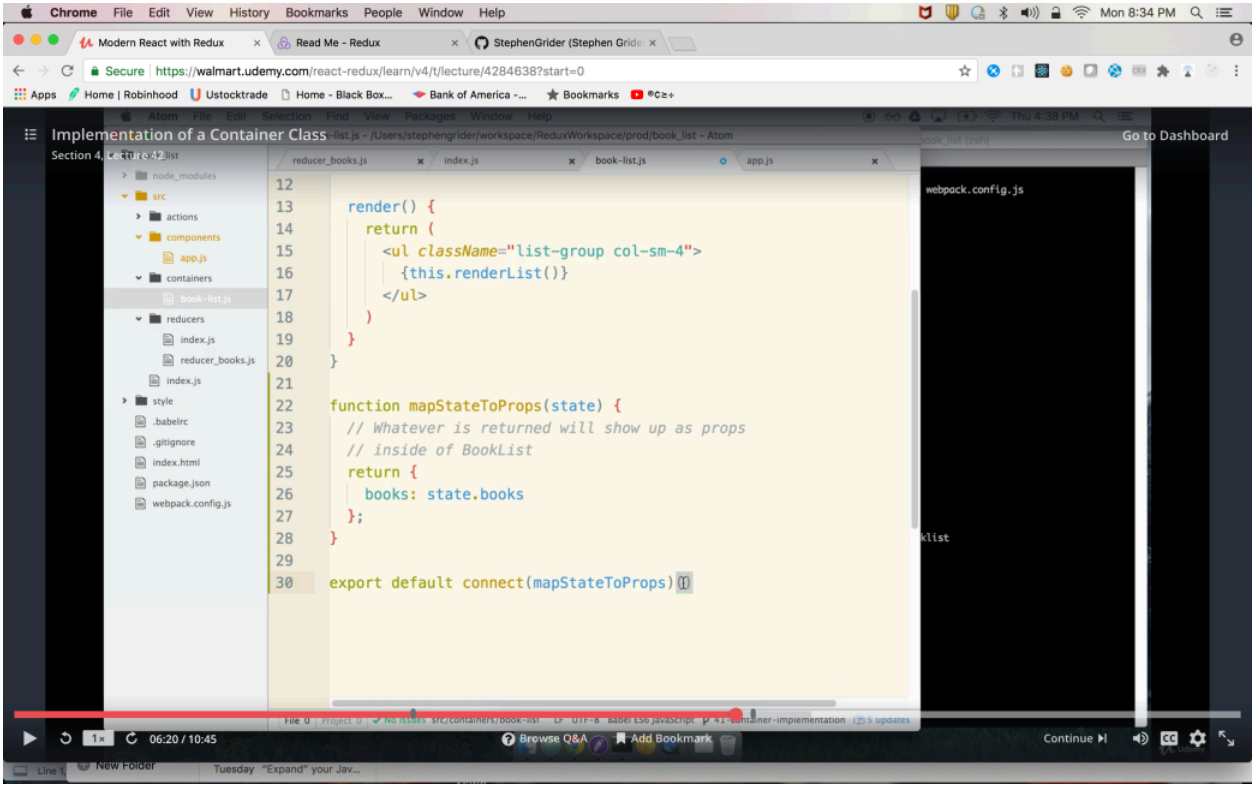
we will return { **books** : books from state }

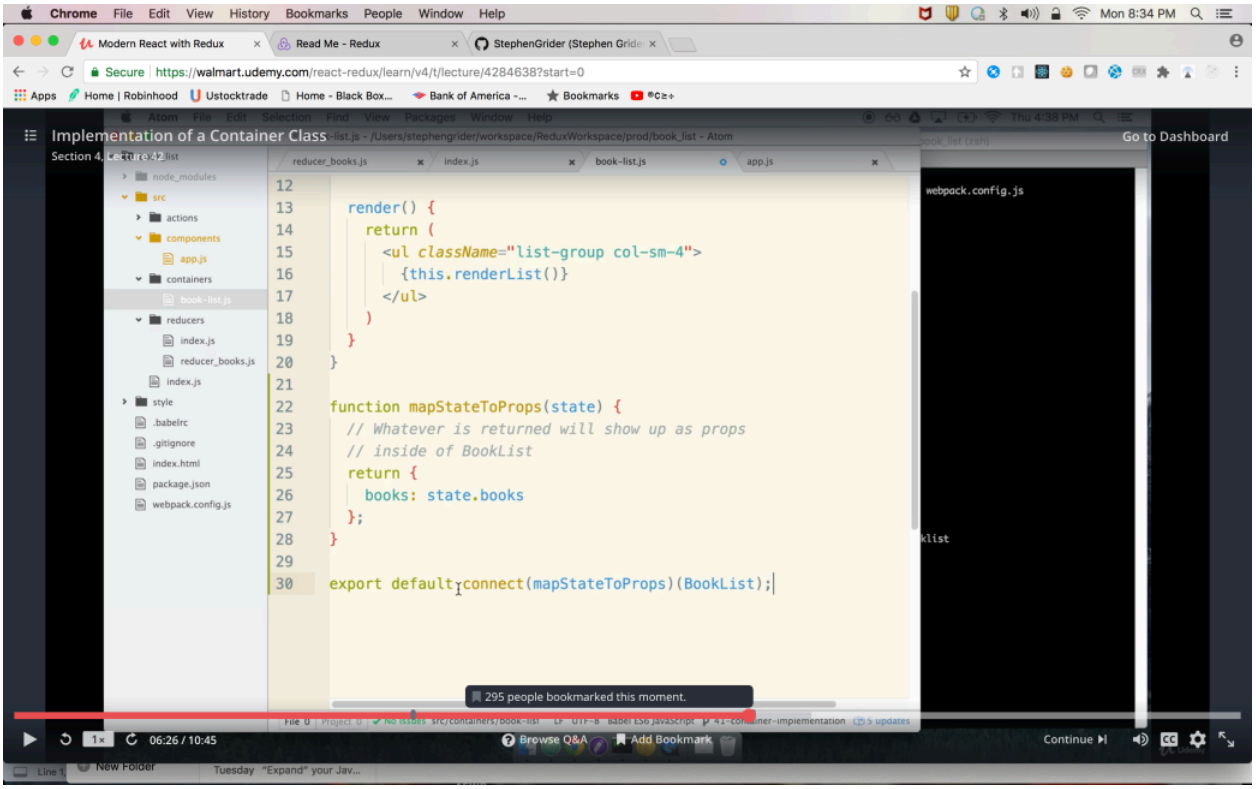


we don’t return export default BookList

instead of this we export default connect which expects a function and component

export default connect(mapStateToProps) it **returns a function that which is called with BookList class**
export default connect(mapStateToProps)(BookList)

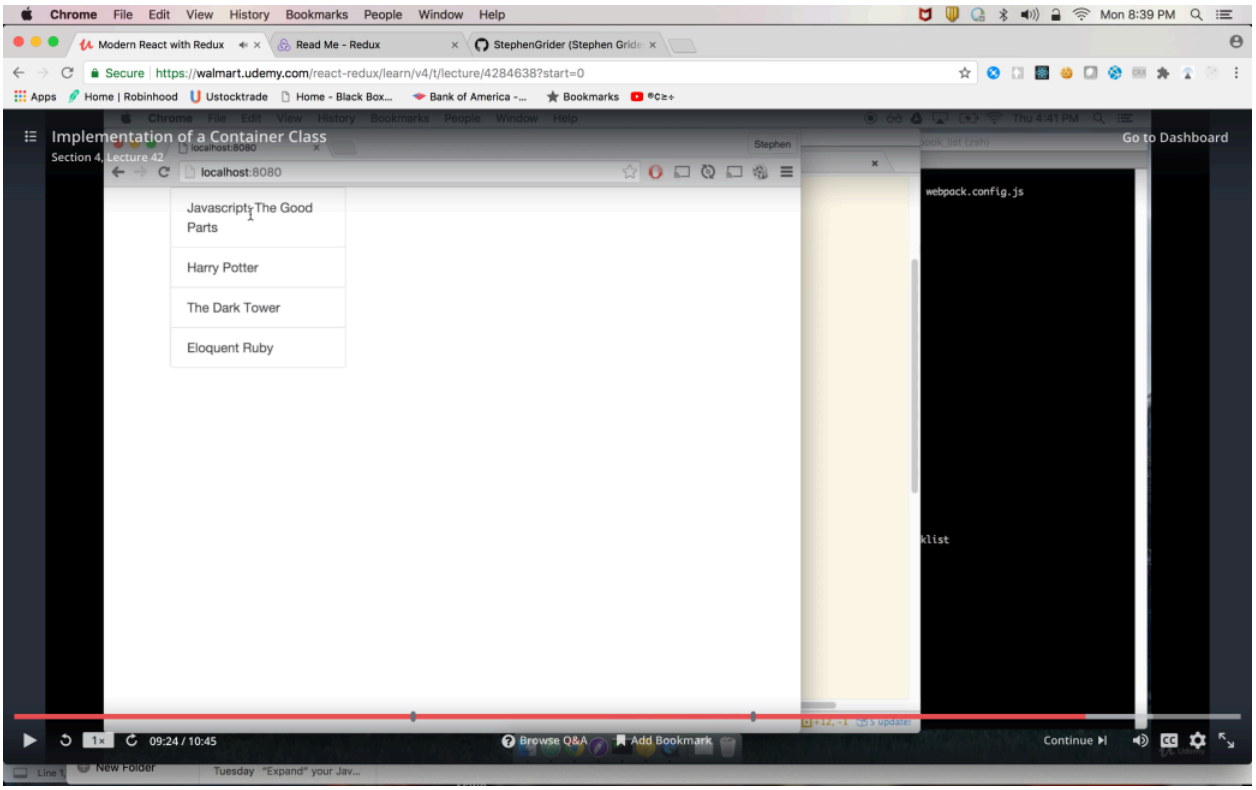

Implementation of a Container Class
Go to Dashboard
Modern React with Redux
Read Me - Redux
Secure | https://walmart.udemy.com/react-redux/learn/v4/t/lecture/4284638?start=0
reducer_books.js
index.js
book-list.js
app.js
webpack.config.js
12
13 render() {
14 return (
15 <ul className="list-group col-sm-4">
16 {this.renderList()}
17
18)
19 }
20 }
21
22 function mapStateToProps(state) {
23 // Whatever is returned will show up as props
24 // inside of BookList
25 return {
26 books: state.books
27 };
28 }
29
30 export default connect(mapStateToProps)
06:20 / 10:45
Browse Q&A
Add Bookmark
Continue


Implementation of a Container Class
Go to Dashboard
Modern React with Redux
Read Me - Redux
Secure | https://walmart.udemy.com/react-redux/learn/v4/t/lecture/4284638?start=0
reducer_books.js
index.js
book-list.js
app.js
webpack.config.js
12
13 render() {
14 return (
15 <ul className="list-group col-sm-4">
16 {this.renderList()}
17
18)
19 }
20 }
21
22 function mapStateToProps(state) {
23 // Whatever is returned will show up as props
24 // inside of BookList
25 return {
26 books: state.books
27 };
28 }
29
30 export default connect(mapStateToProps)(BookList);
295 people bookmarked this moment.
06:26 / 10:45
Browse Q&A
Add Bookmark
Continue

2 Imp notes

whenever application **states changes** (like list of books changes)
this container will re render with new list of books

and object in **mapStateToProps** will assign as props to component



User click on Book 2

we call an Action Creator .
Action Creator is a function which **return a object** which we called **Action { ACTION.TYPE: actionname, DATA: {data object}, }**

Action : {
type: BOOK_SELECTED // object has a **TYPE of ACTION** is triggered
book : {title: 'Book 2'} // object also has **DATA** that further **describes the action**.

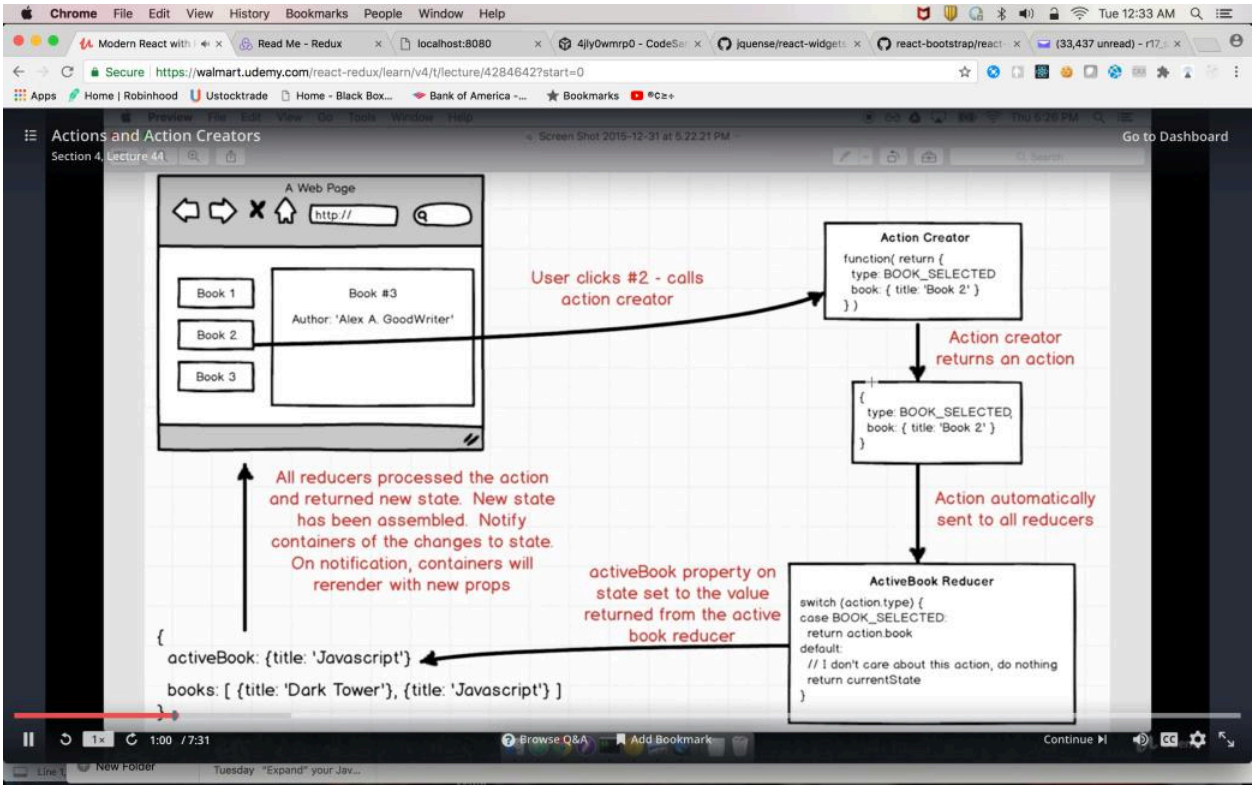
}

Action sent To **all reducers** in the application

Reducer has **switch case** based on action.type
it returns the value which is set to state

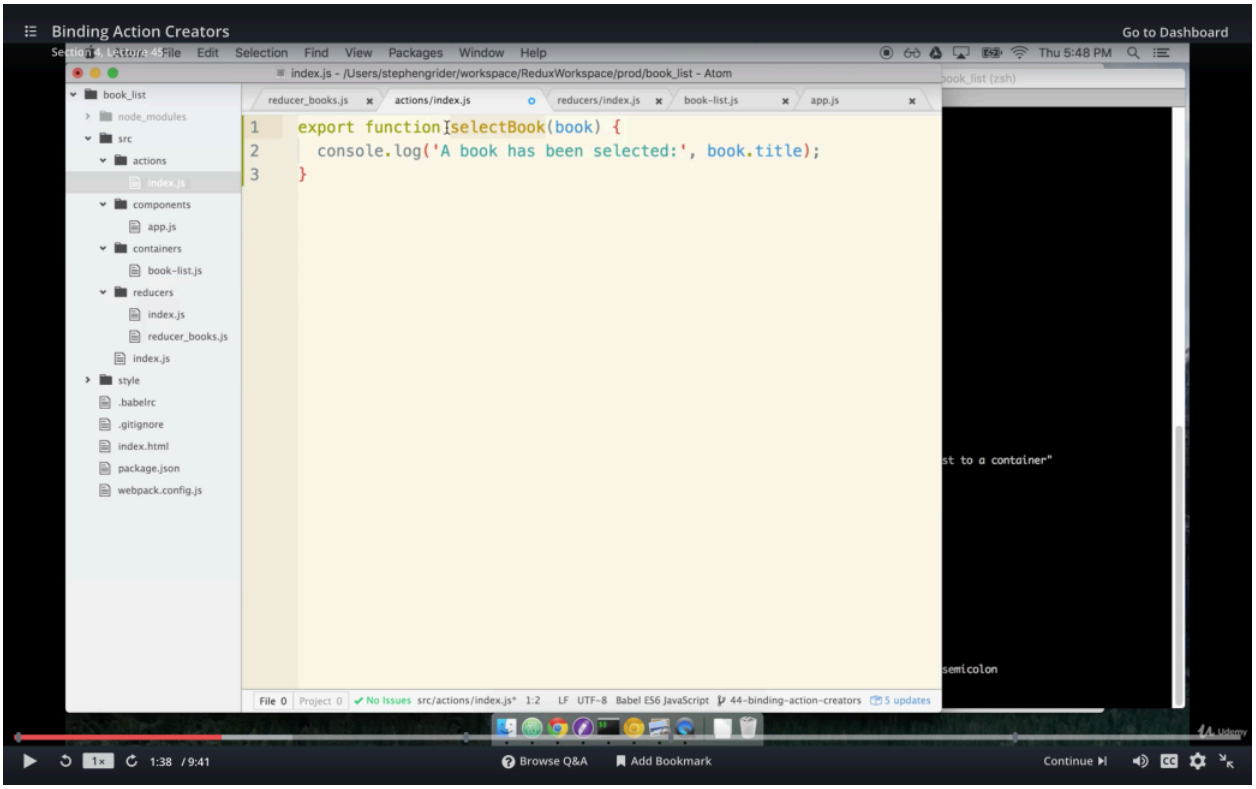
```
switch (action.type) {
        case " ":  {
        }
        default " ": {
        }
}
```

once **reducer return** the **state to mapStateToProps**
New **state flow to all** the **containers**
which it will **cause re render** of page.



go to **actions/index.js**
create single **action creator selectBook**

action Creator is just function
it will only take one book object with title property
we will export the function



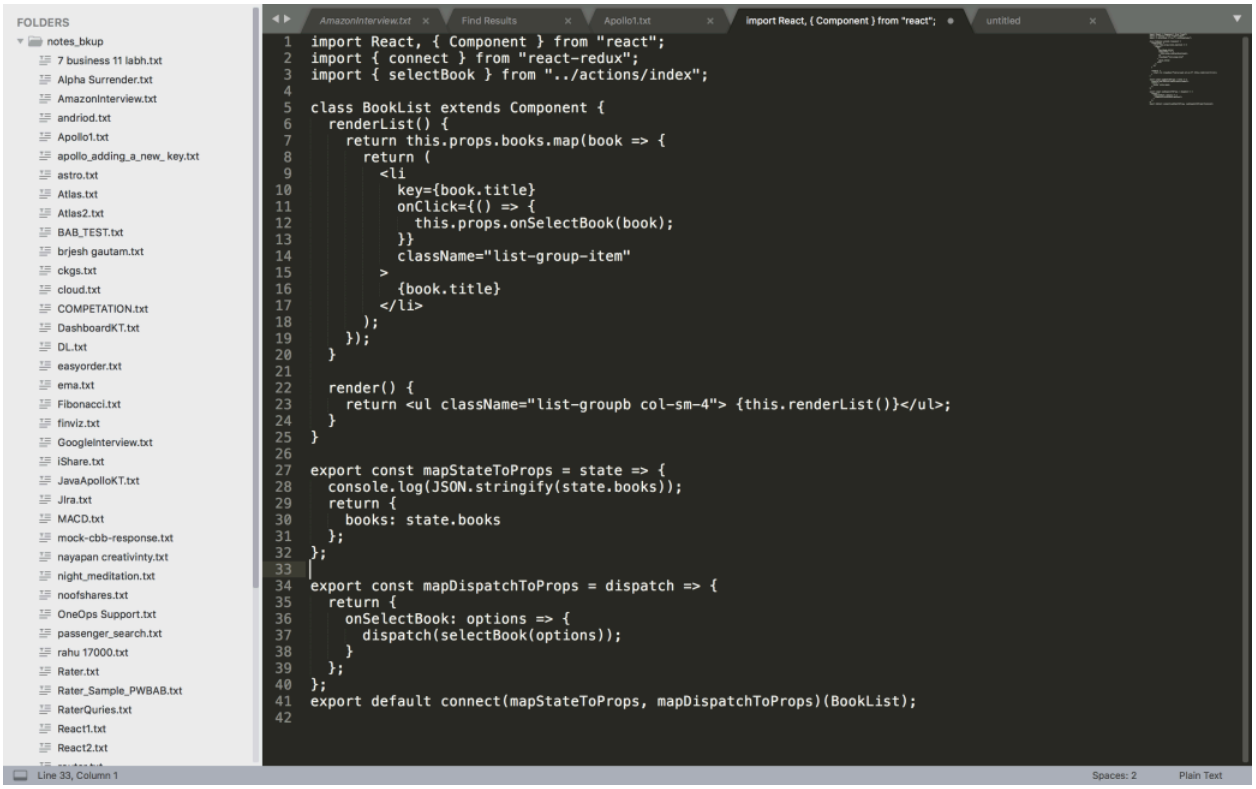
import {selectBook} action to this BookList component

1 // we will add **mapDispatchToProps** outside of class BookList (similar to mapStateToProps)

2 //

add function **mapDispatchToProps = (dispatch) =>** {
return {
onSelectBook : (**options) => dispatch (selectBook(options));**
}
}

3 in connect method we will add another argument with mapStateToProps, **mapDispatchToProps**



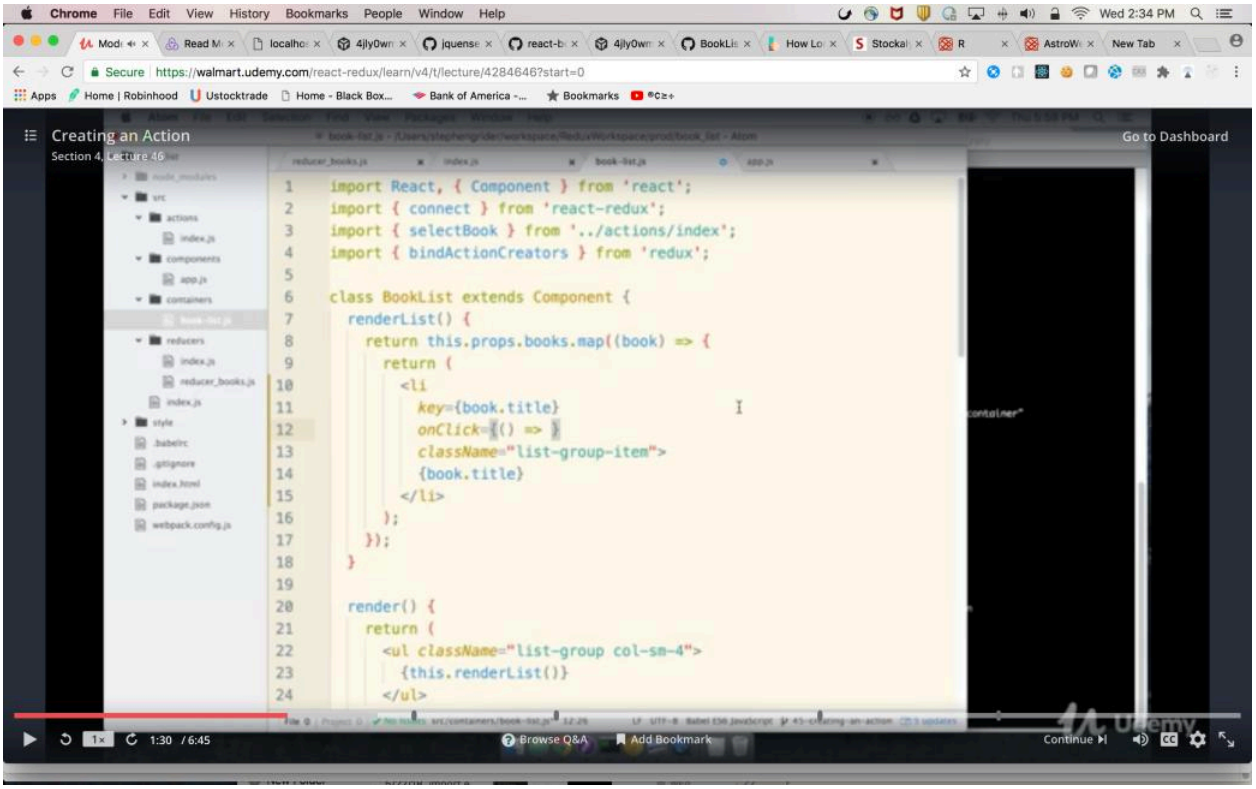
whenever selectBook is called, the result should be passed
to all the our reducers using dispatch function

anything return from this function will ends up in props on BookList container.

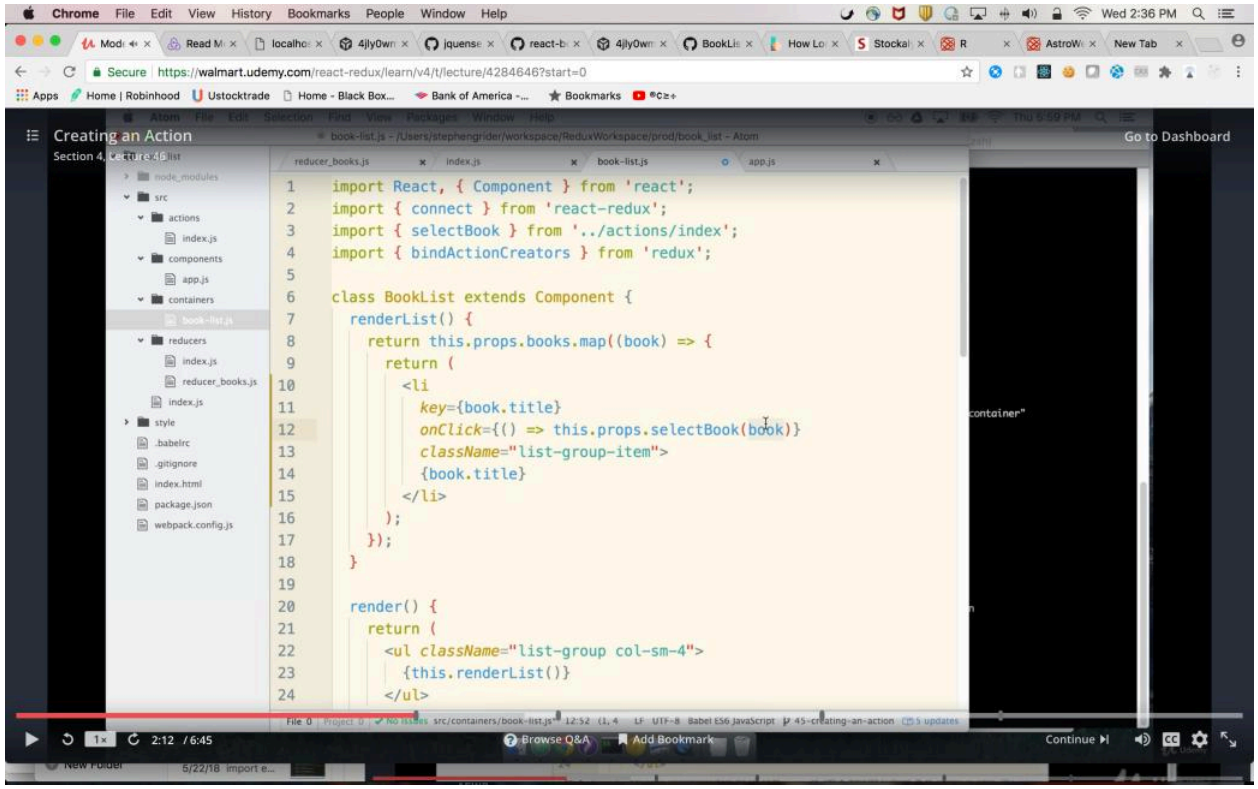
now we called this.props.**onSelectBook()** in our container
that will call our action creator

we will add onClick on Li
onClick = {}
we will add arrow function

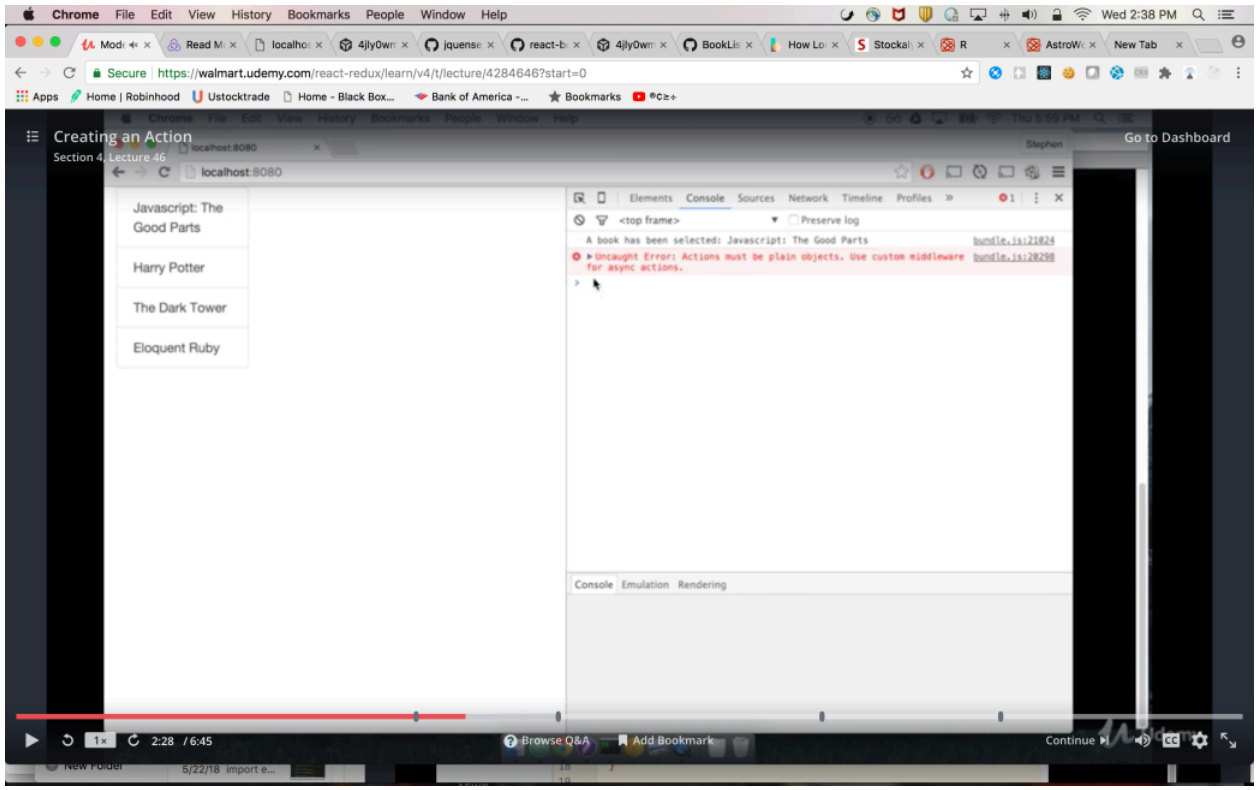
onClick = { () => {} }



we will call this.props.selectBook() action with book argument
props.selectBook(book)

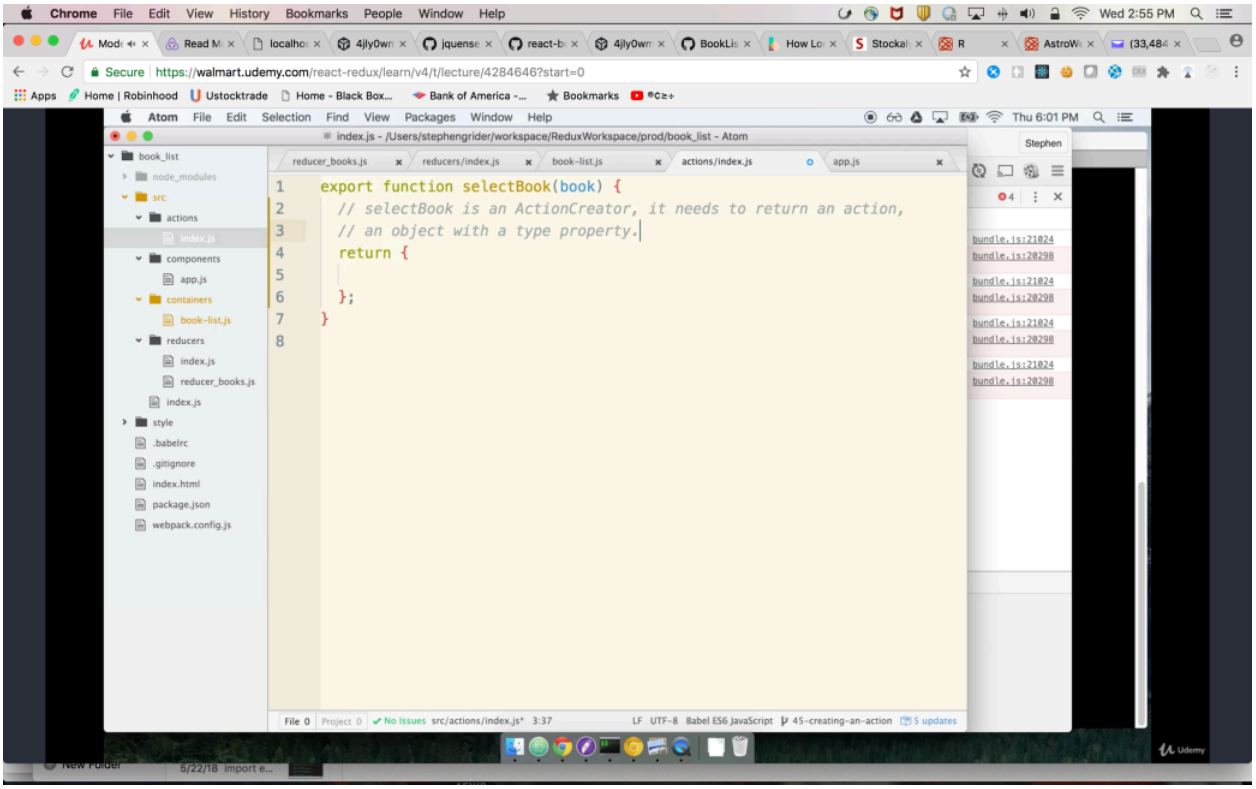


now click on container's li will show on action's console.log

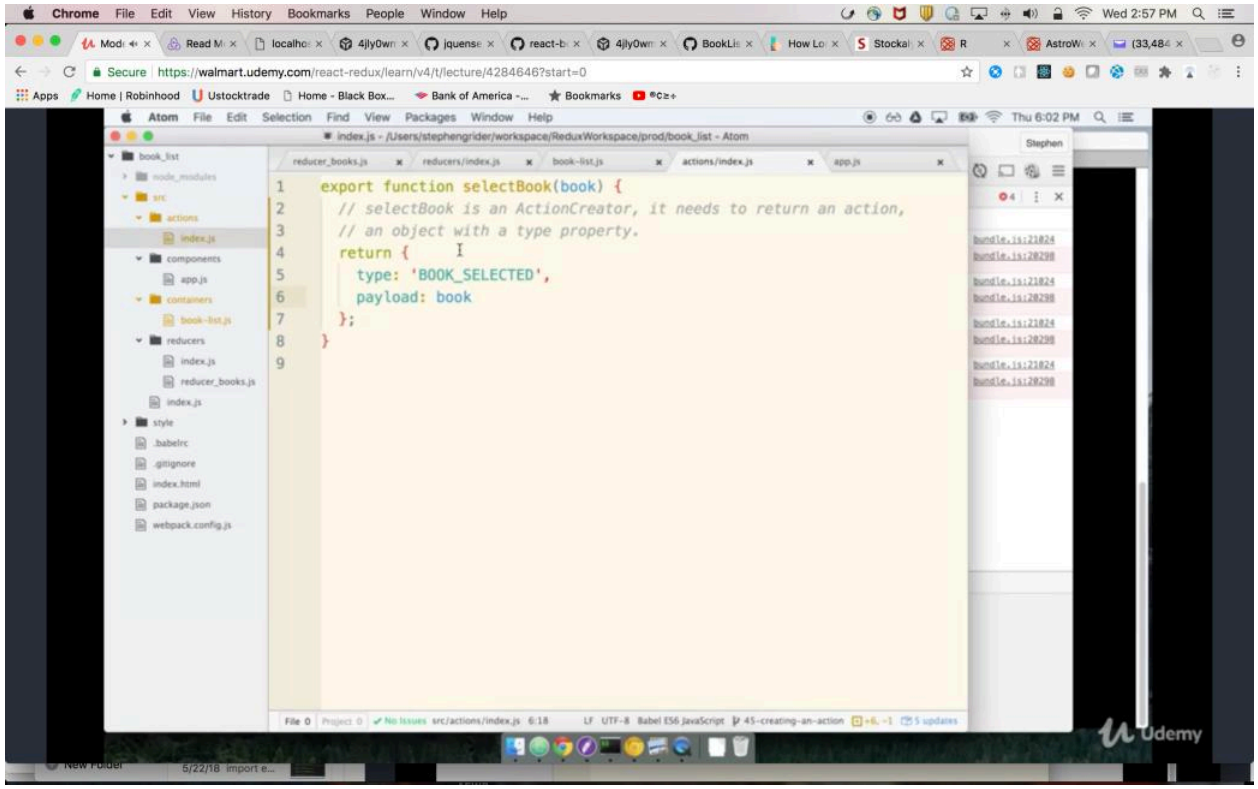


now selectBook we will change selectBook

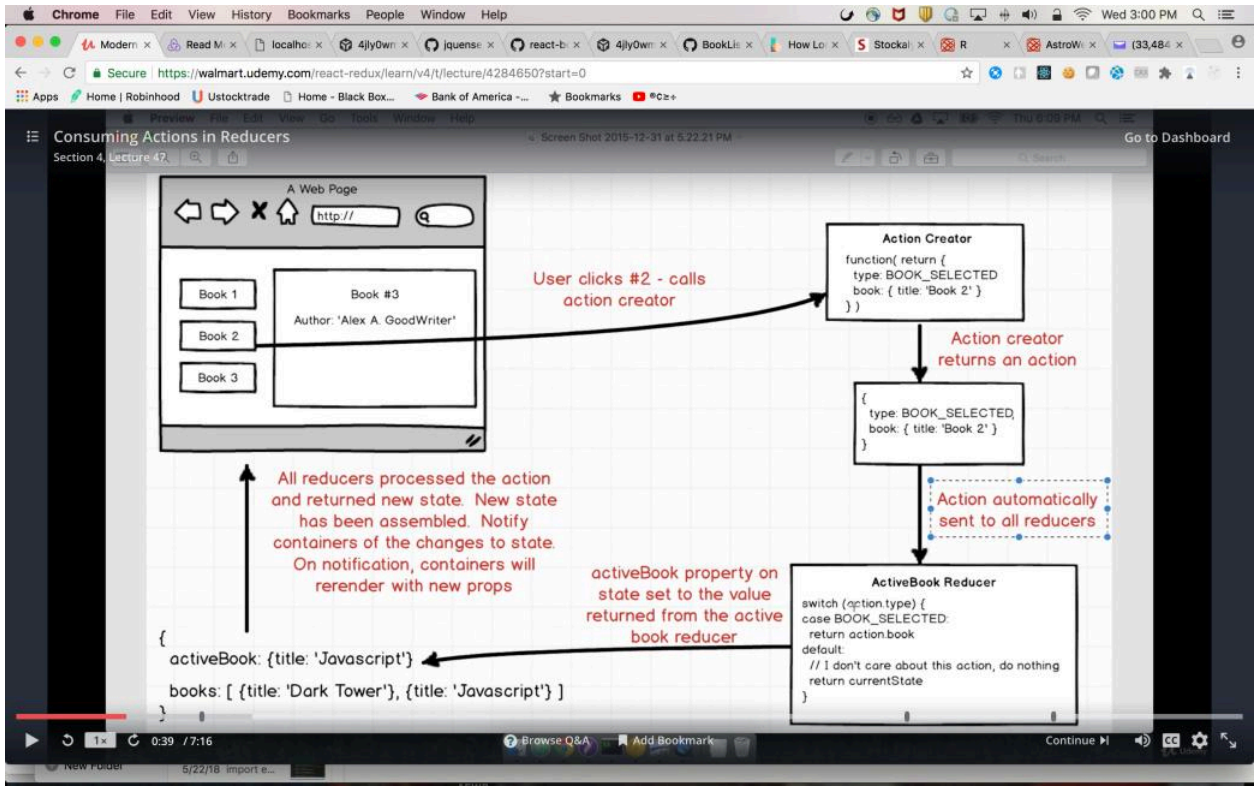
//selectBook is a ActionCreator , it needs to return an action,
// an object with type property and payload



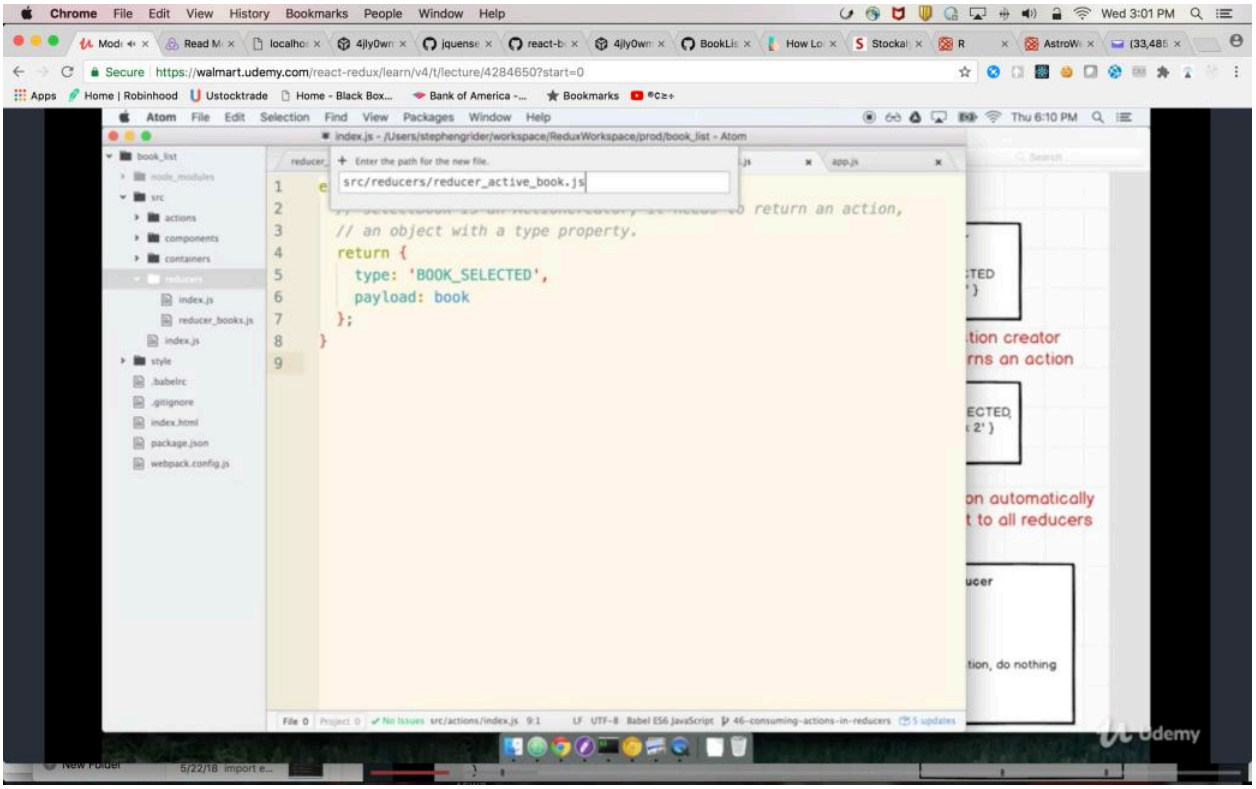
type: upper case string
payload: data



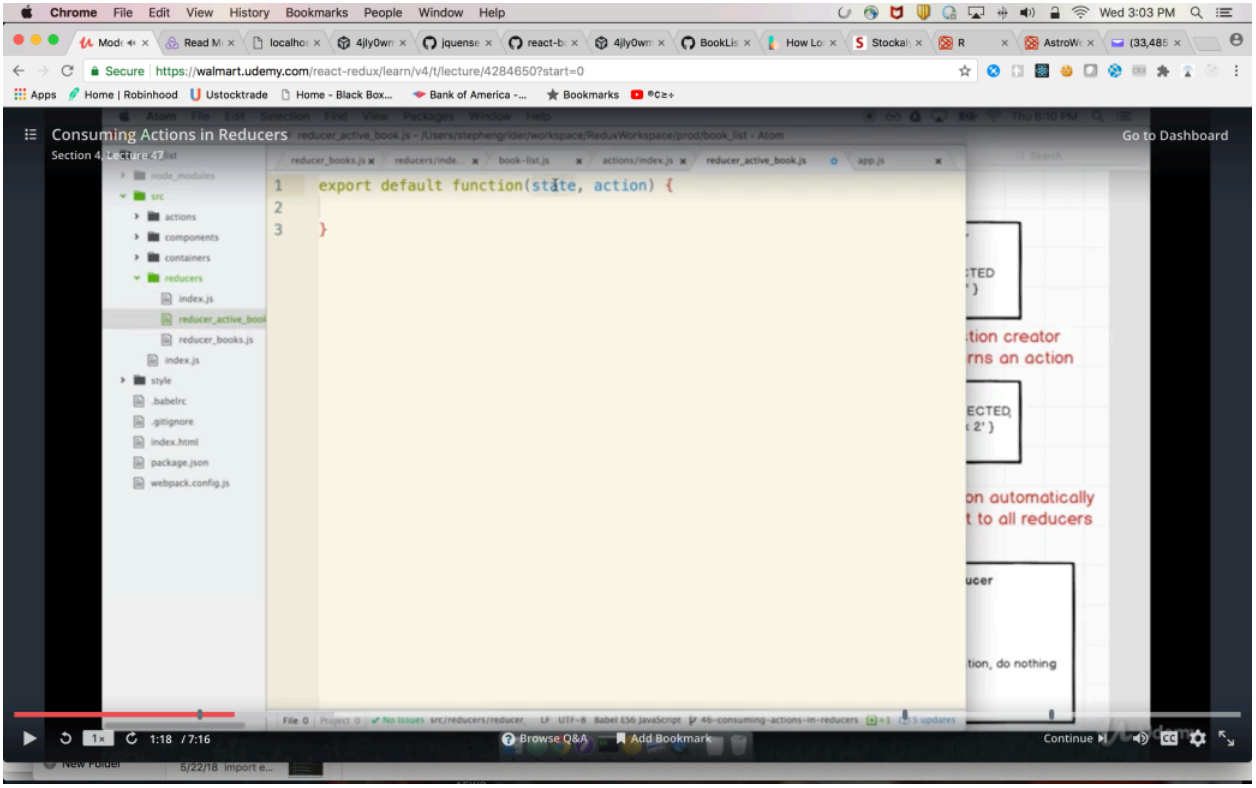
after this we will call ActiveBook Reducer



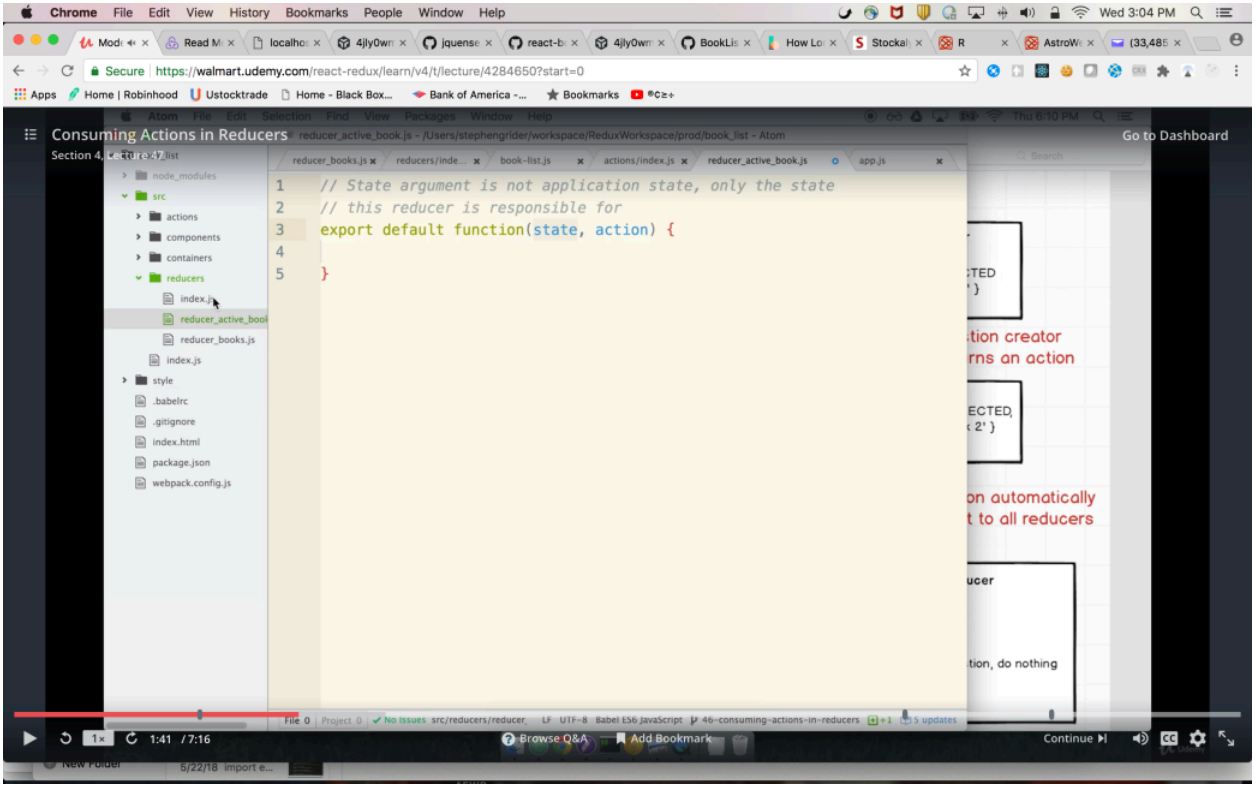
we will create new file in reducers dir
reducer_active_book.js



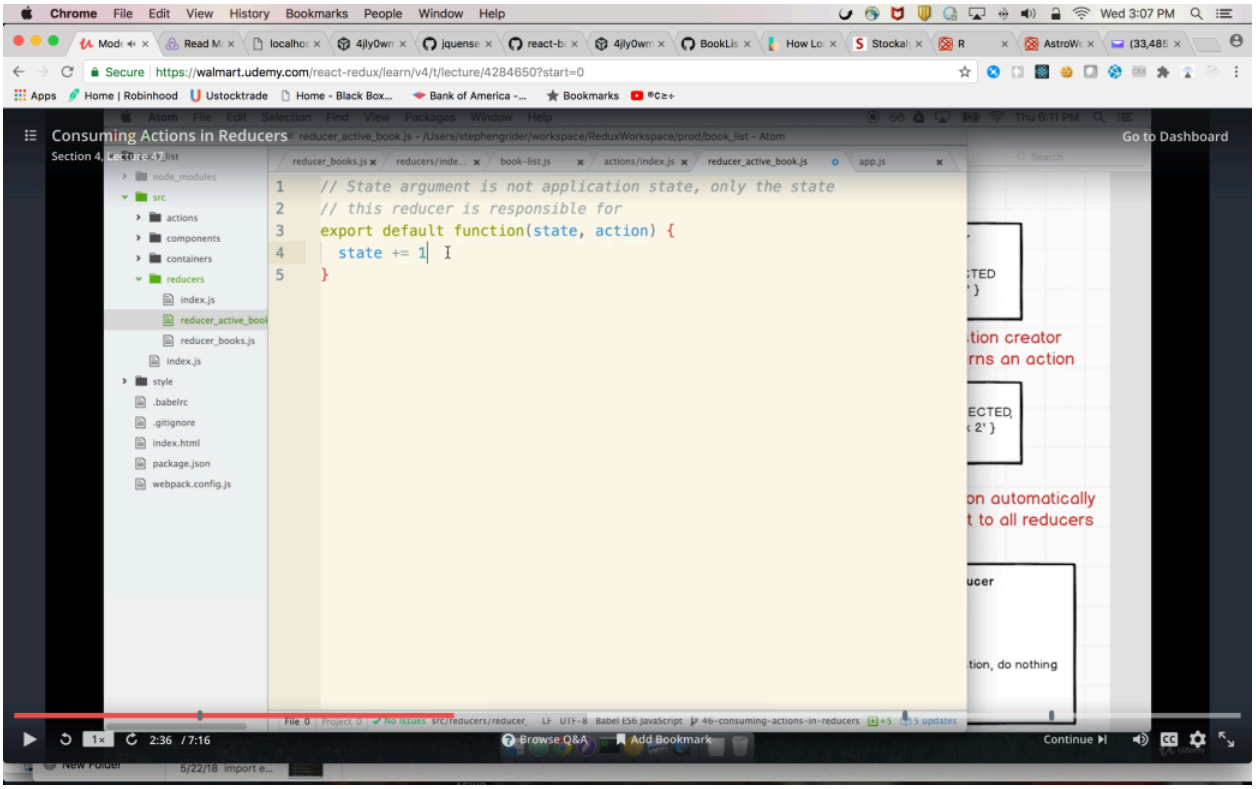
every reducer has 2 arguments
state, action



state is the current state of reducer not the application state

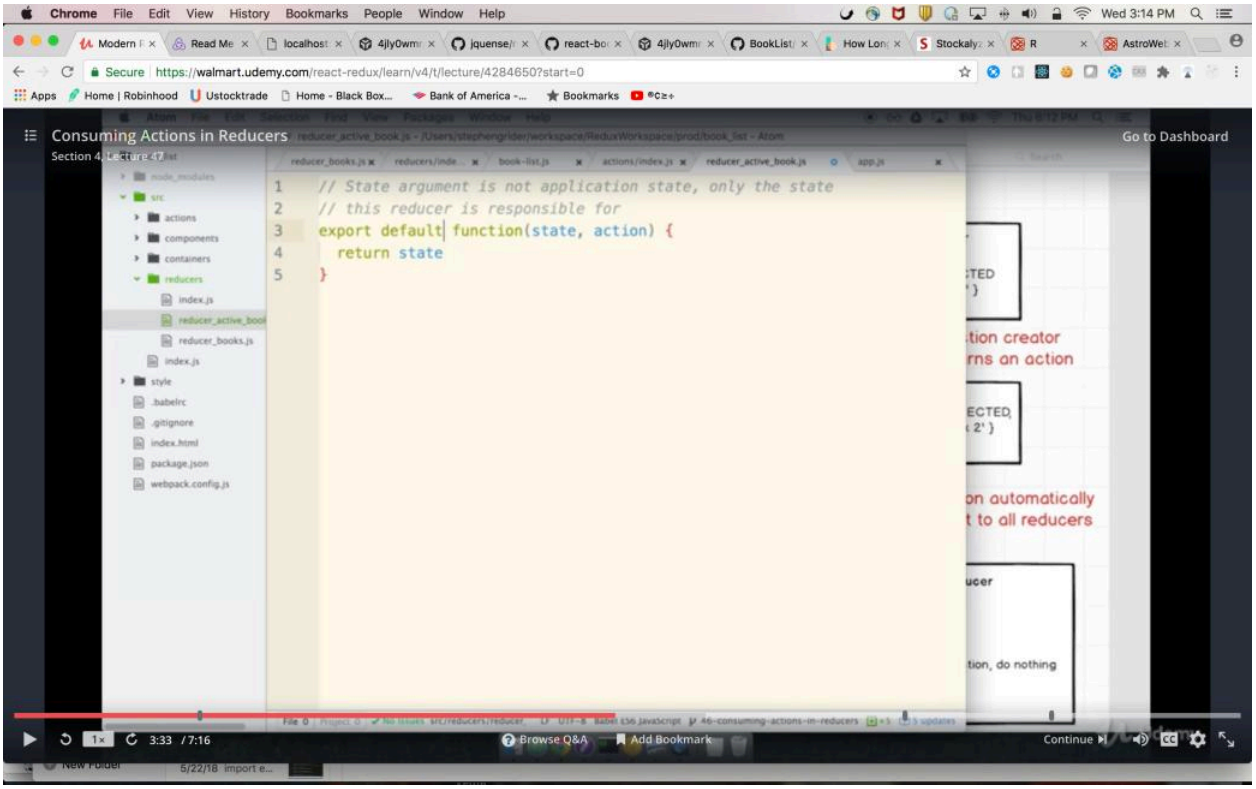


for example if state is number
first time action is triggered it will update to 1
next time action will triggered tit will update to 2and so on
same state is flowing into reducer over an over again when action is called

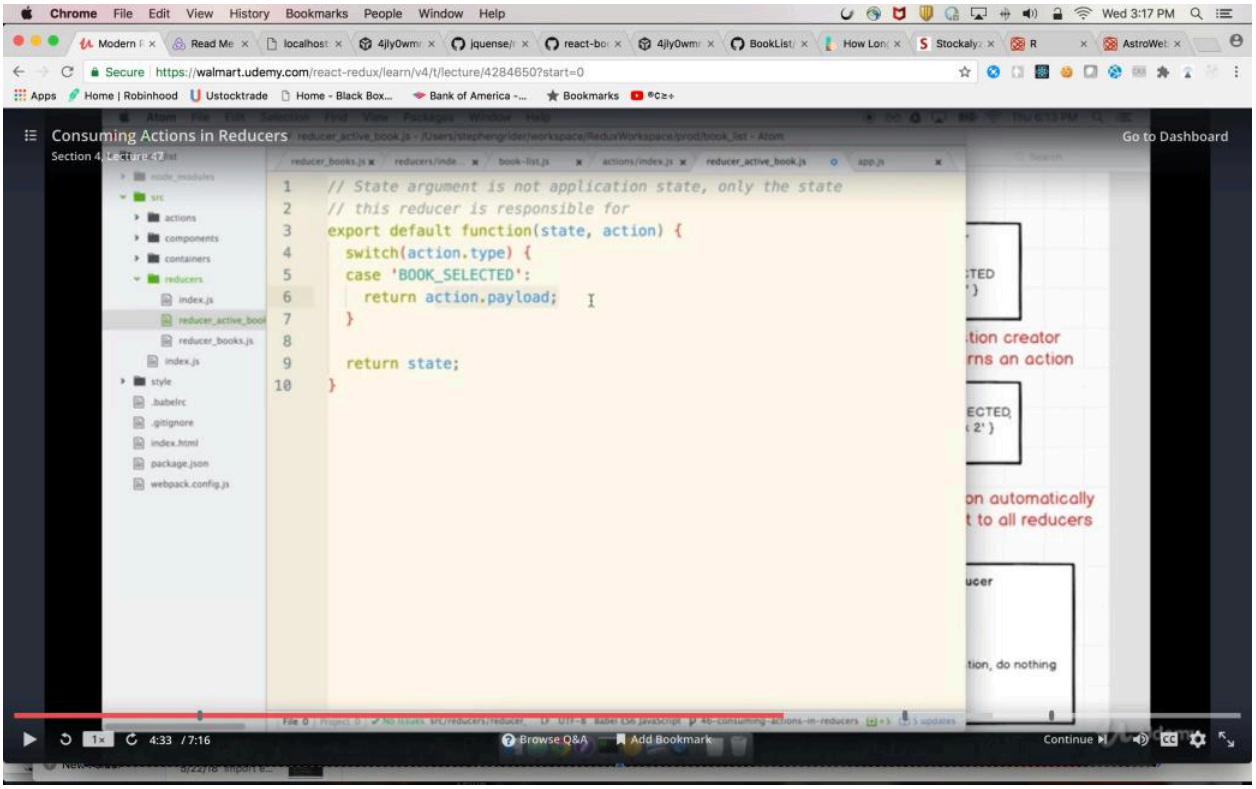


if we don't care about current action pass the current state back through

return state

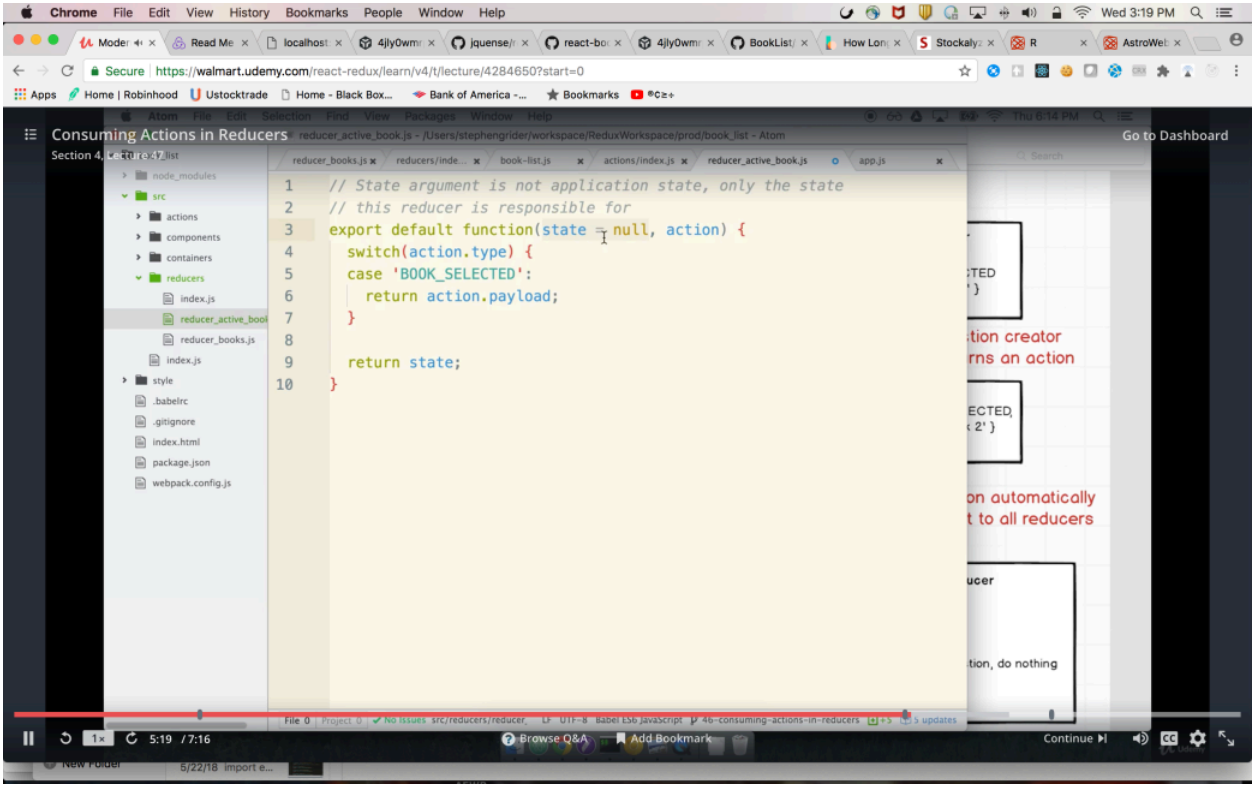


if action type is BOOK_SELECTED
we will return action.payload (which is selected book)



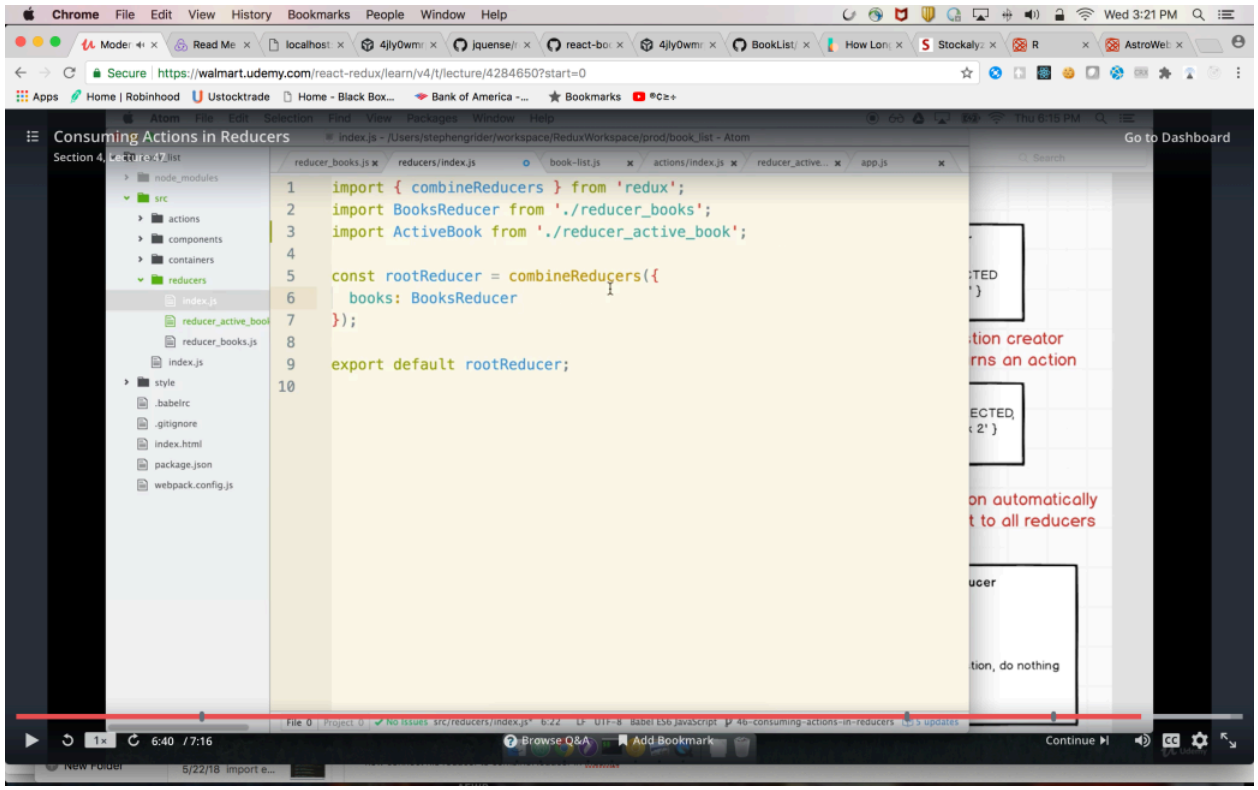
at beginning if user has not clicked on any thing
state will undefined
redux does not allow us to return undefined
to avoid that we will default state argument to null at top
this is es6 syntax if argument is undefined it will set it null

other wise if action is any thing else we just return current state

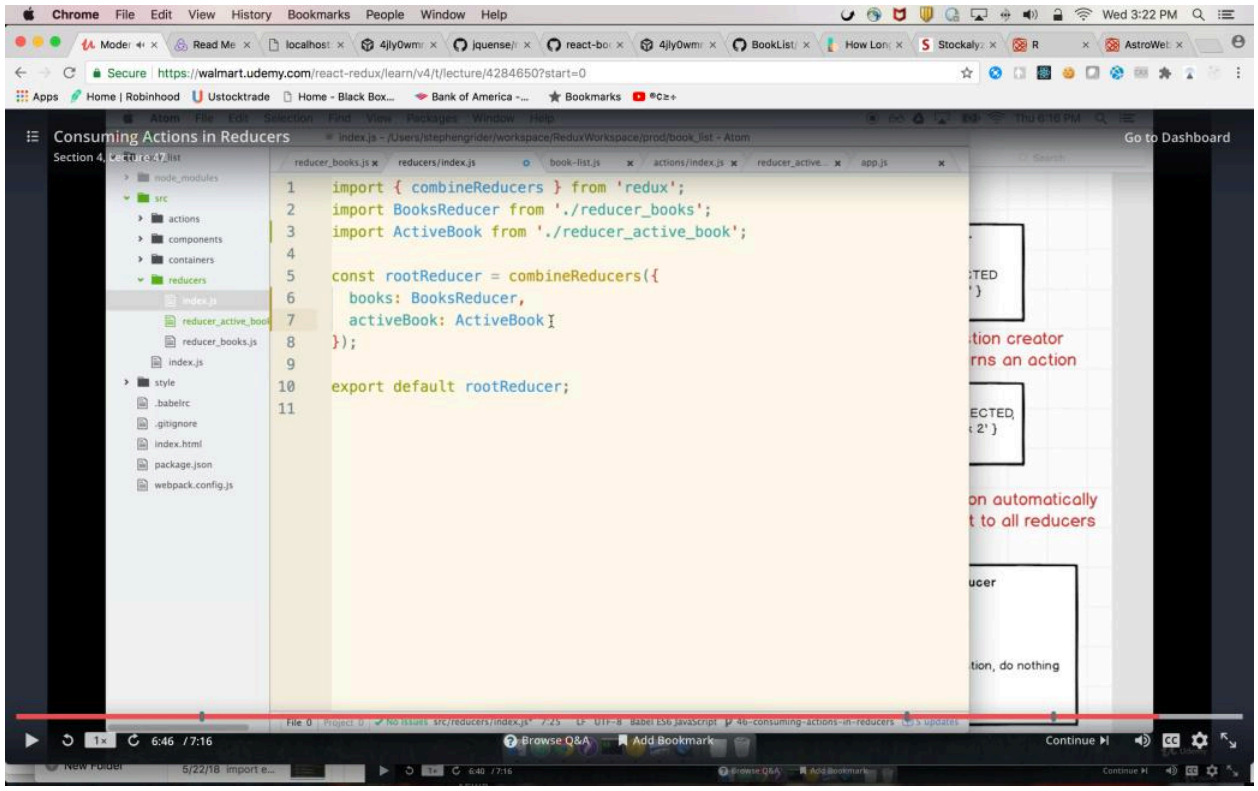


now connect his reducer to combineReducer in index.js

import ActiveBook Reducer

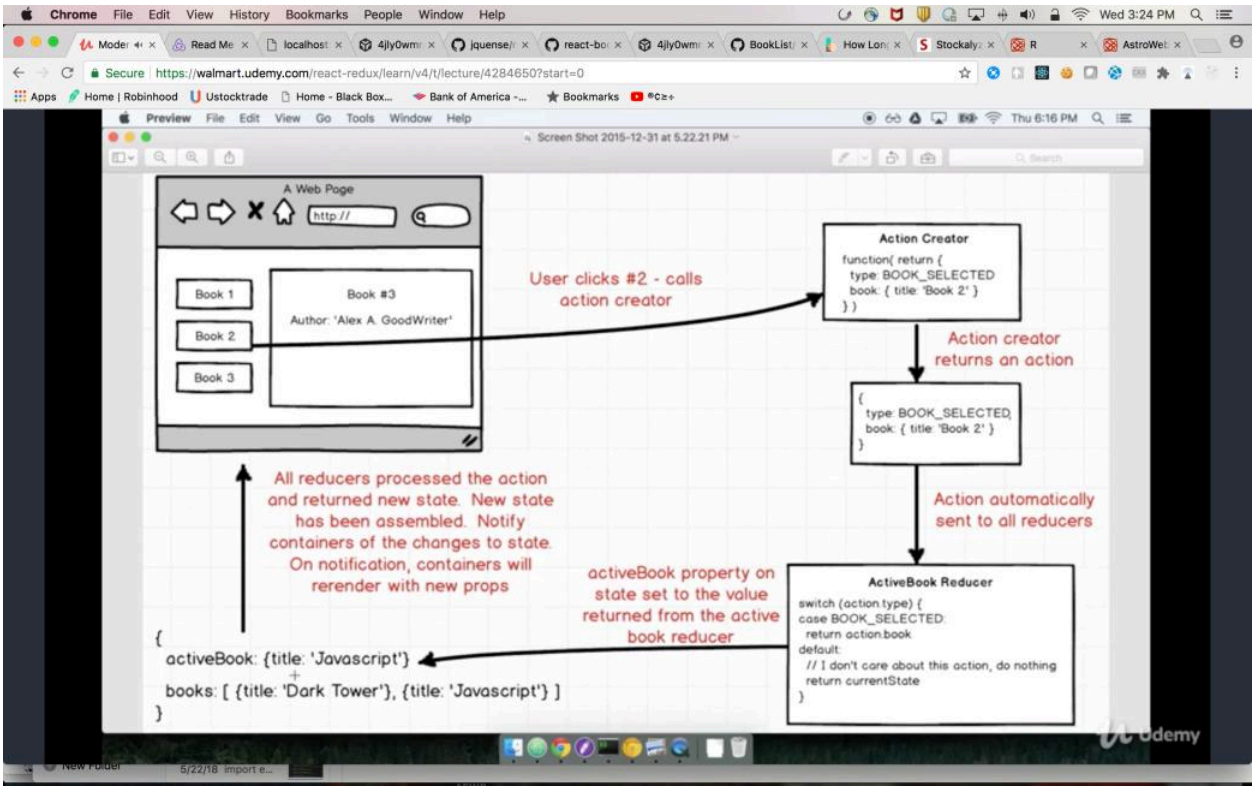


and combine it as another piece of state activeBook activeBook: ActiveBook

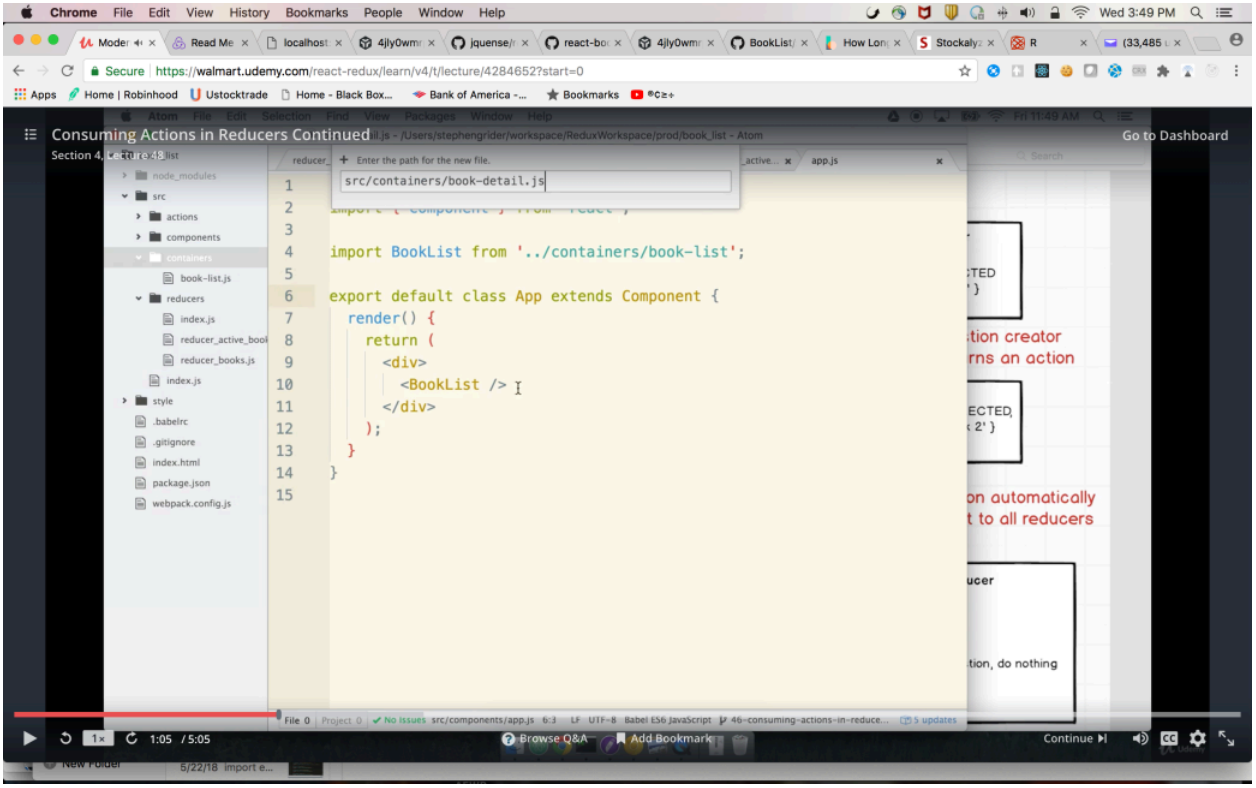


now any key to combineReducers object will ends up in global state

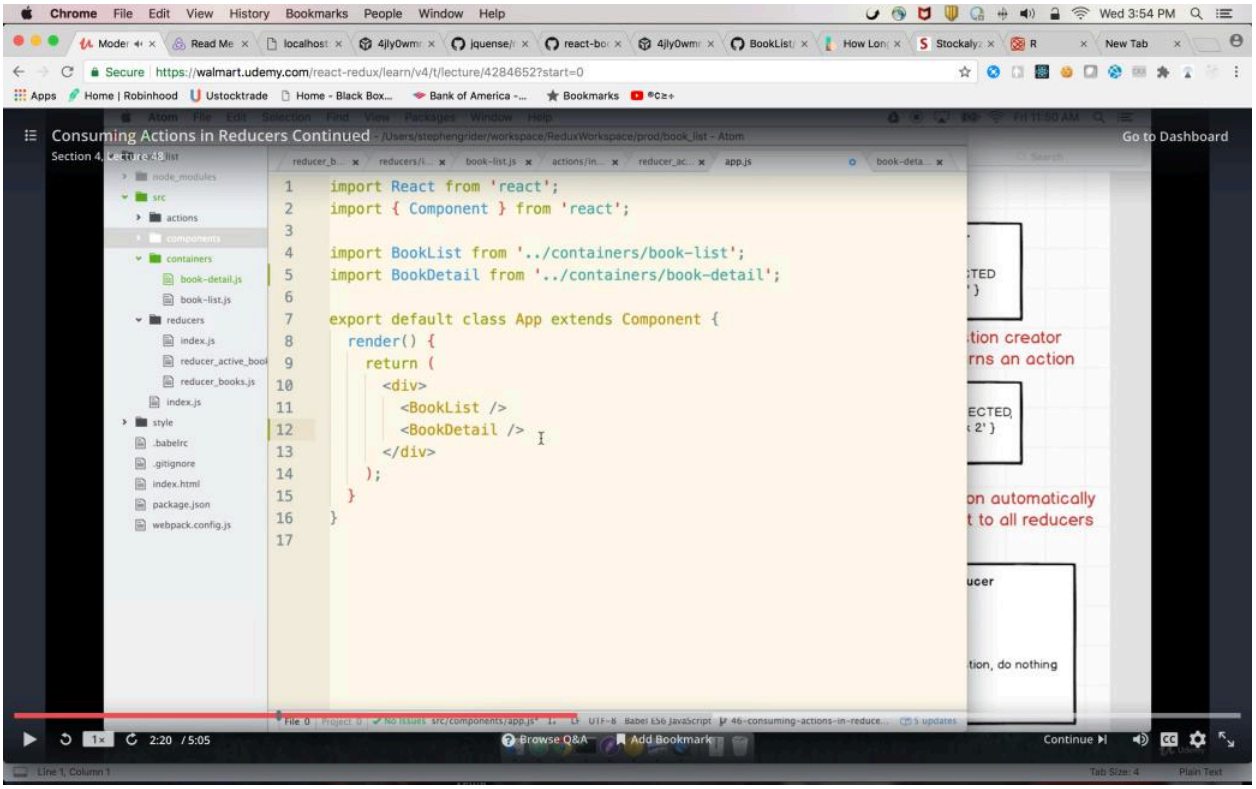
now **books , activeBook** will appear in global state



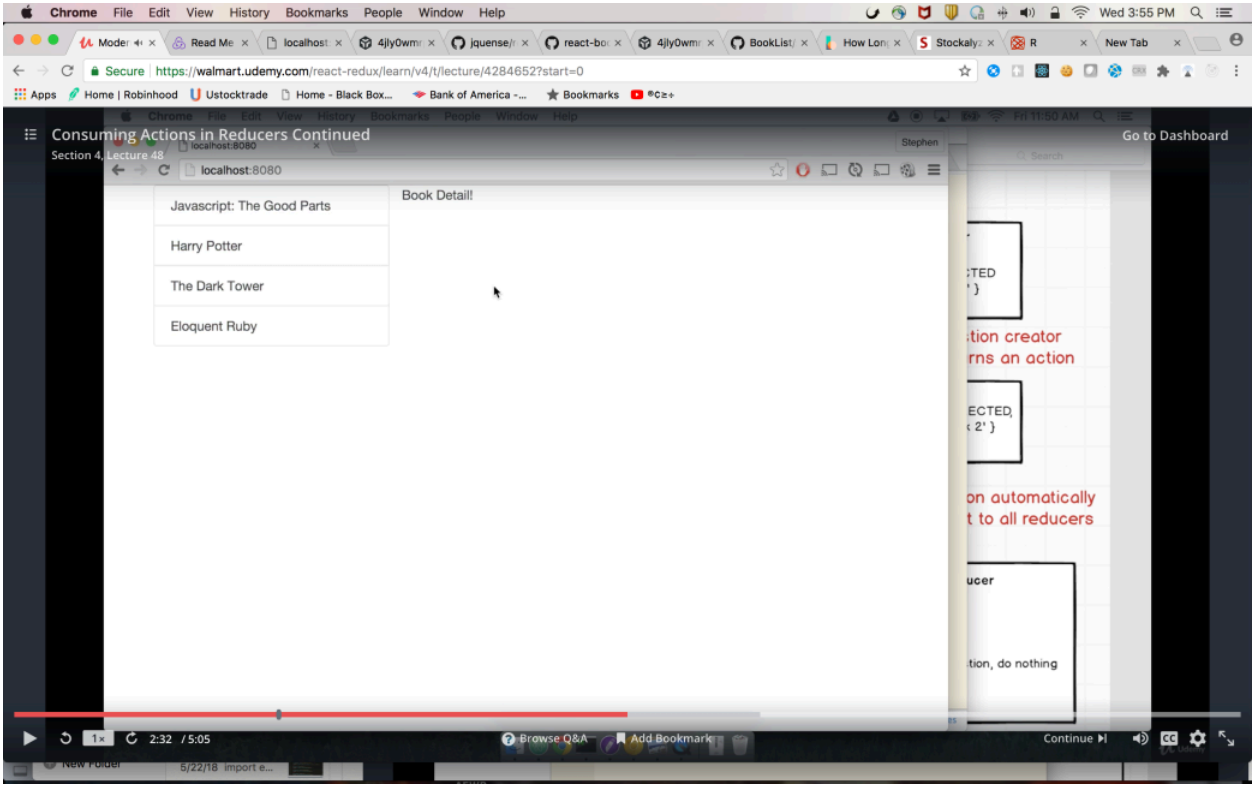
now we will create a new component in container folder book_detail.js (it is in container as it will connect to redux store)



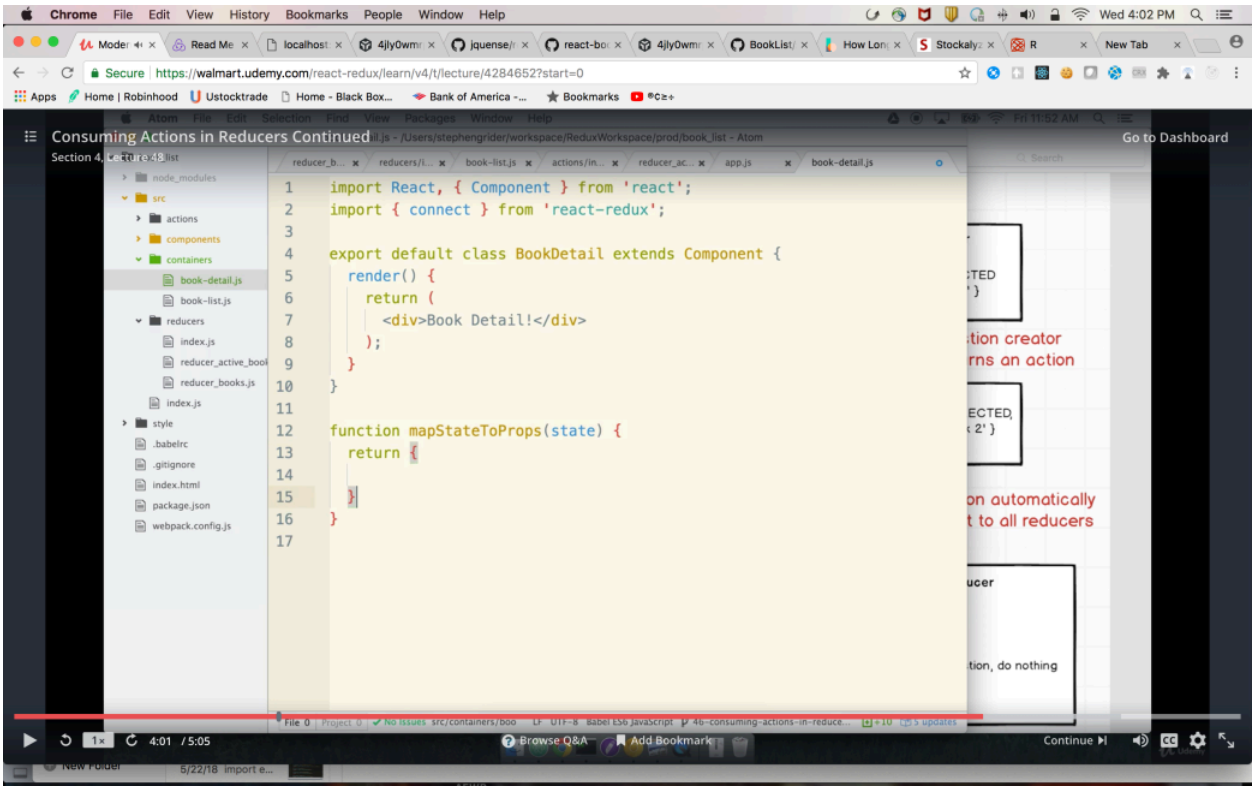
import BookDetail
and call it in redner() method.



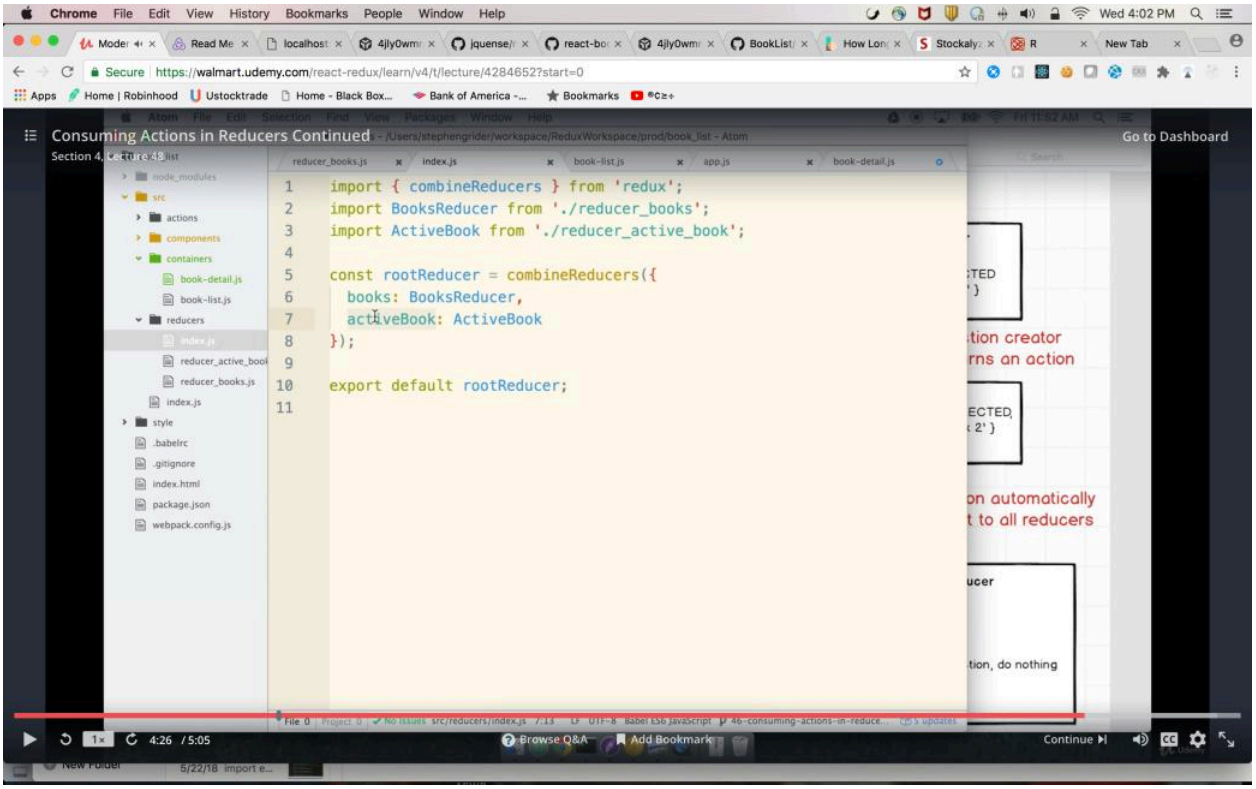
now refresh the browser now components are showing up here



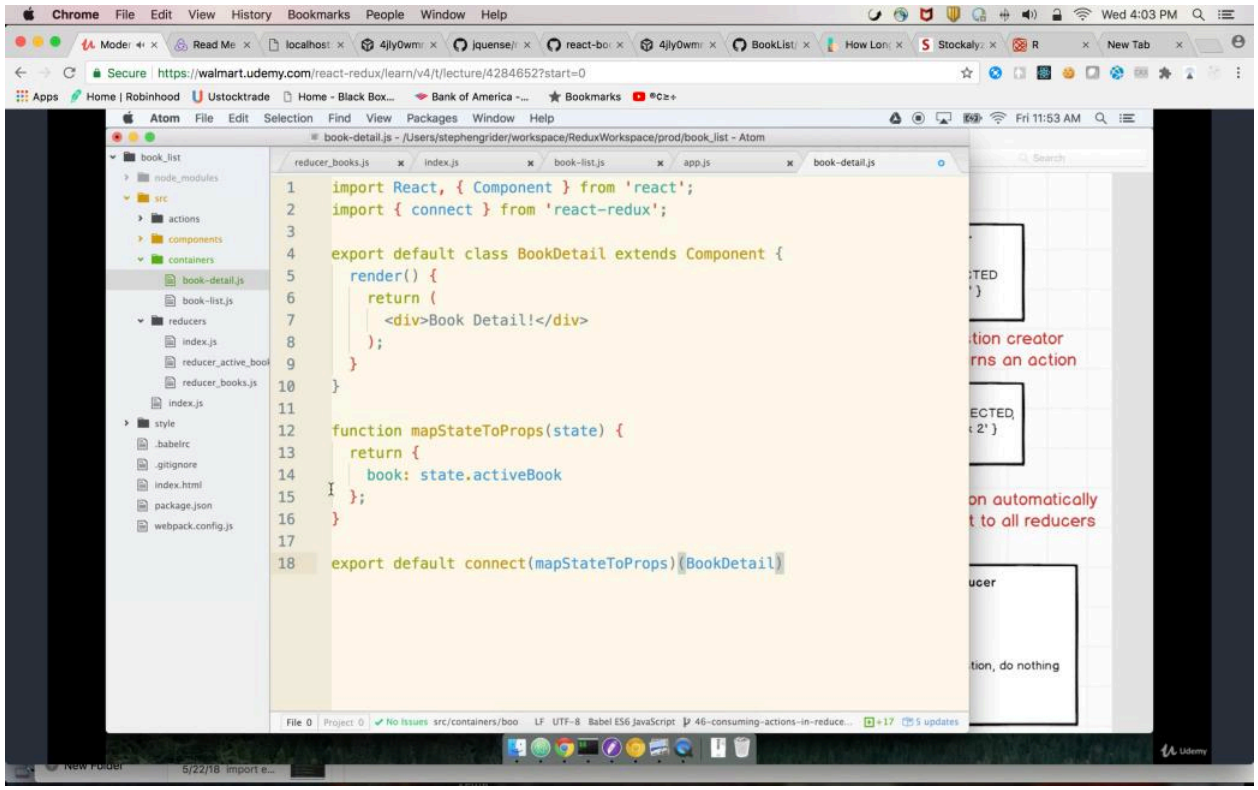
now we will import {connect} from ‘react-redux’
and define mapStateToProps



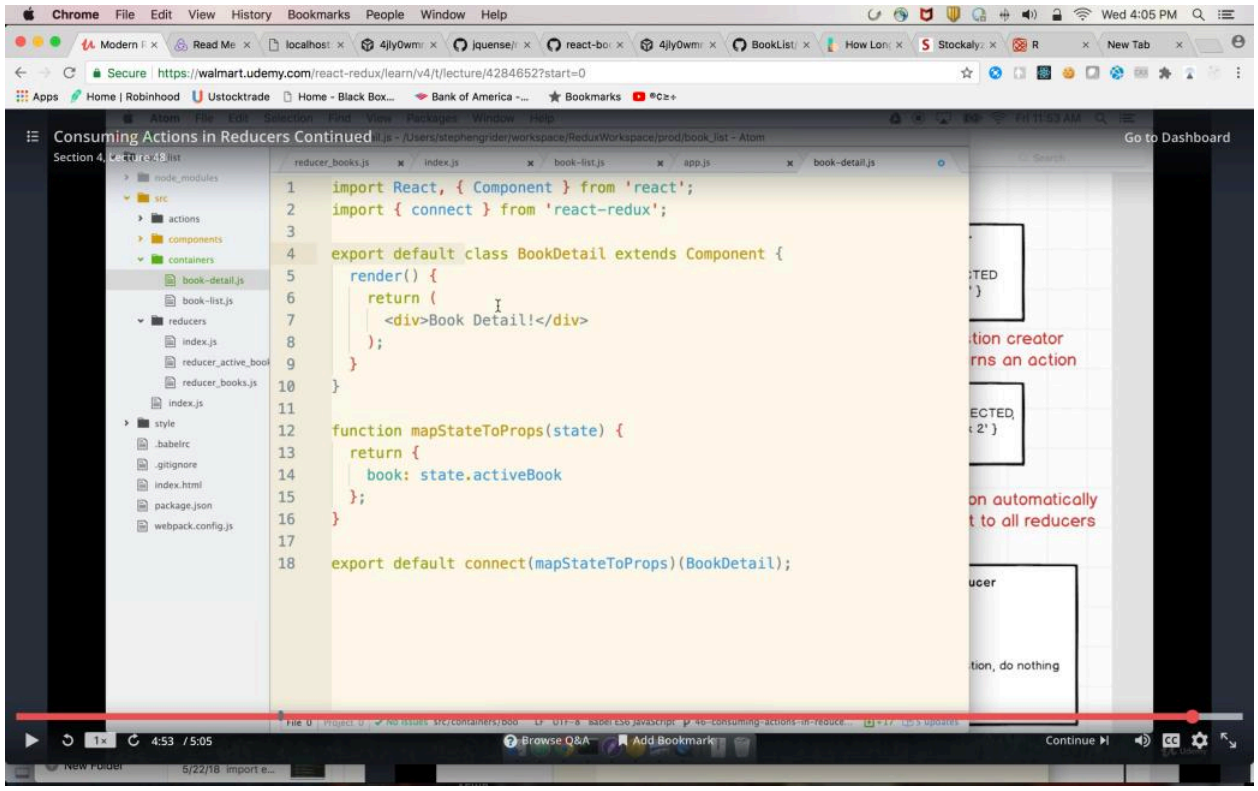
now we need to reference activeBook



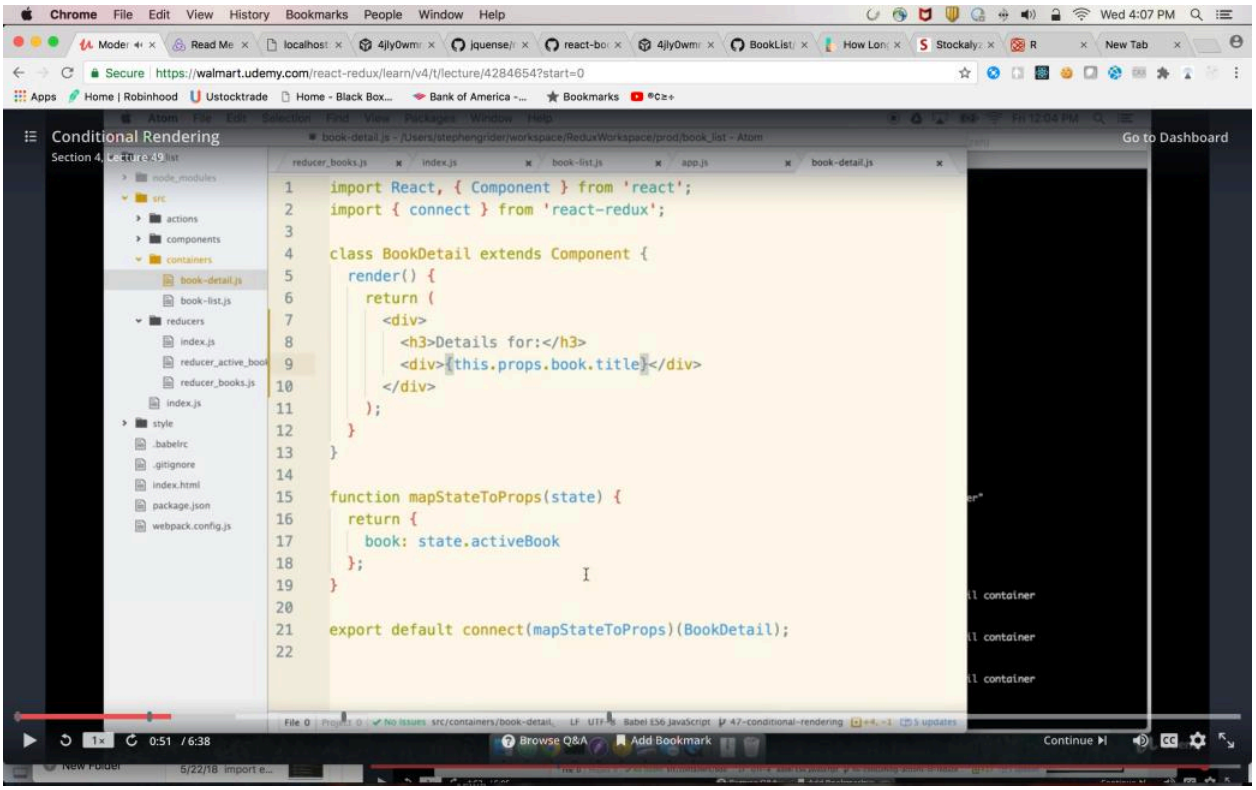
we will return book : state.activeBook (avtiveBook set in state coming from combineReducers({}))
now we export default connect(mapStateToPrps)(BookDetail)



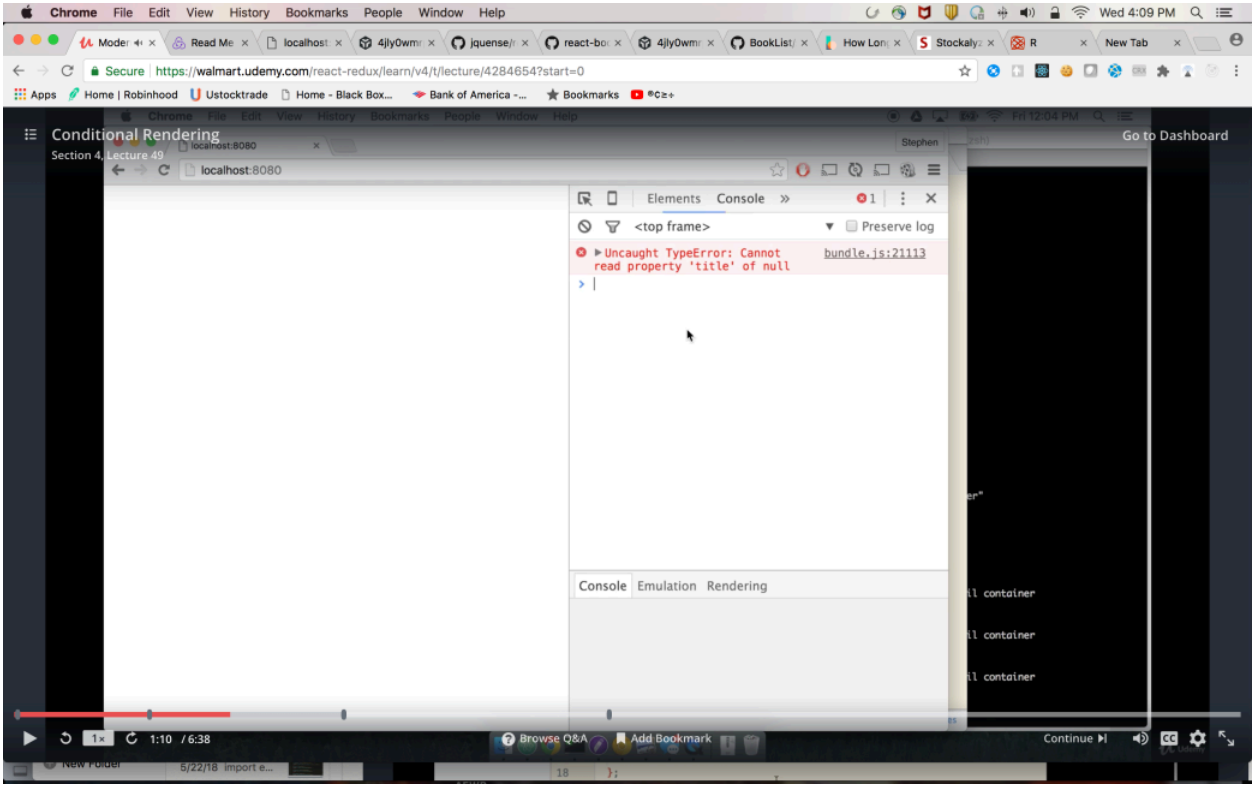
remove export default from class name



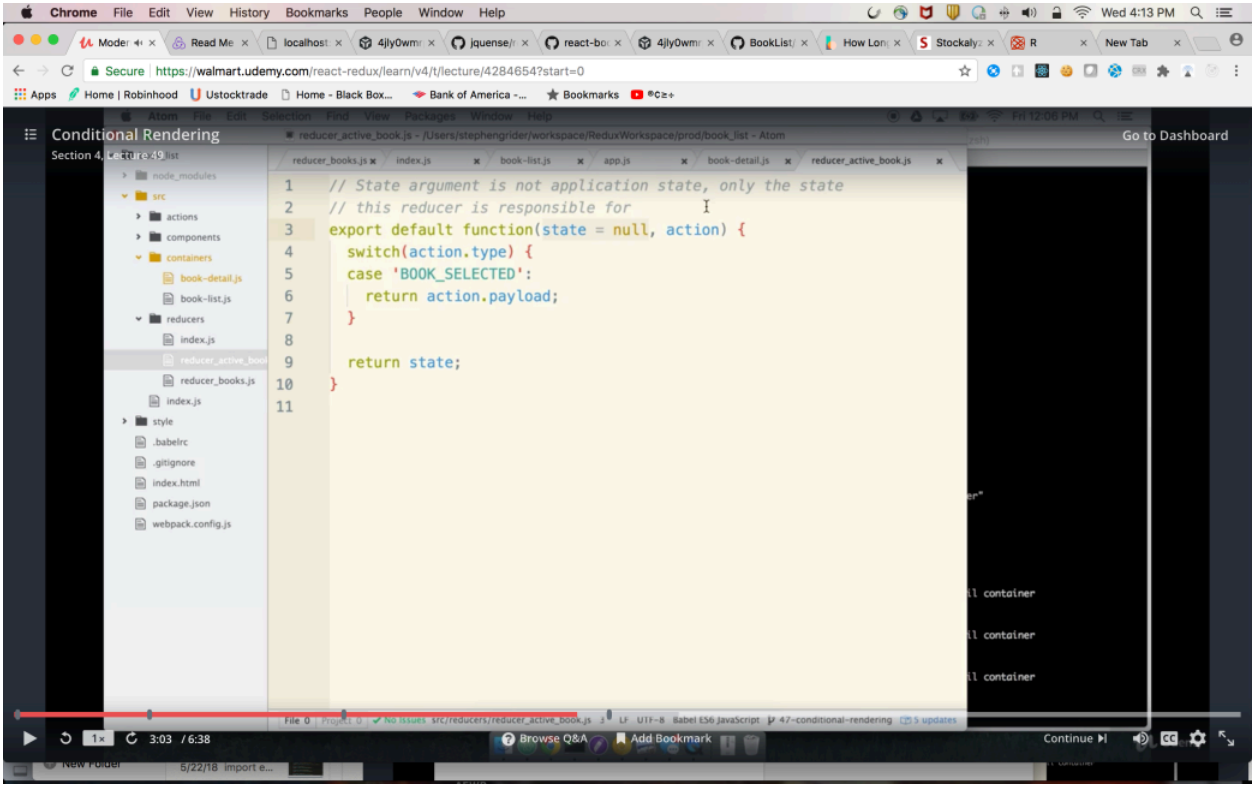
now we can reference this.props.book return from mapStateToProps
we will render the title



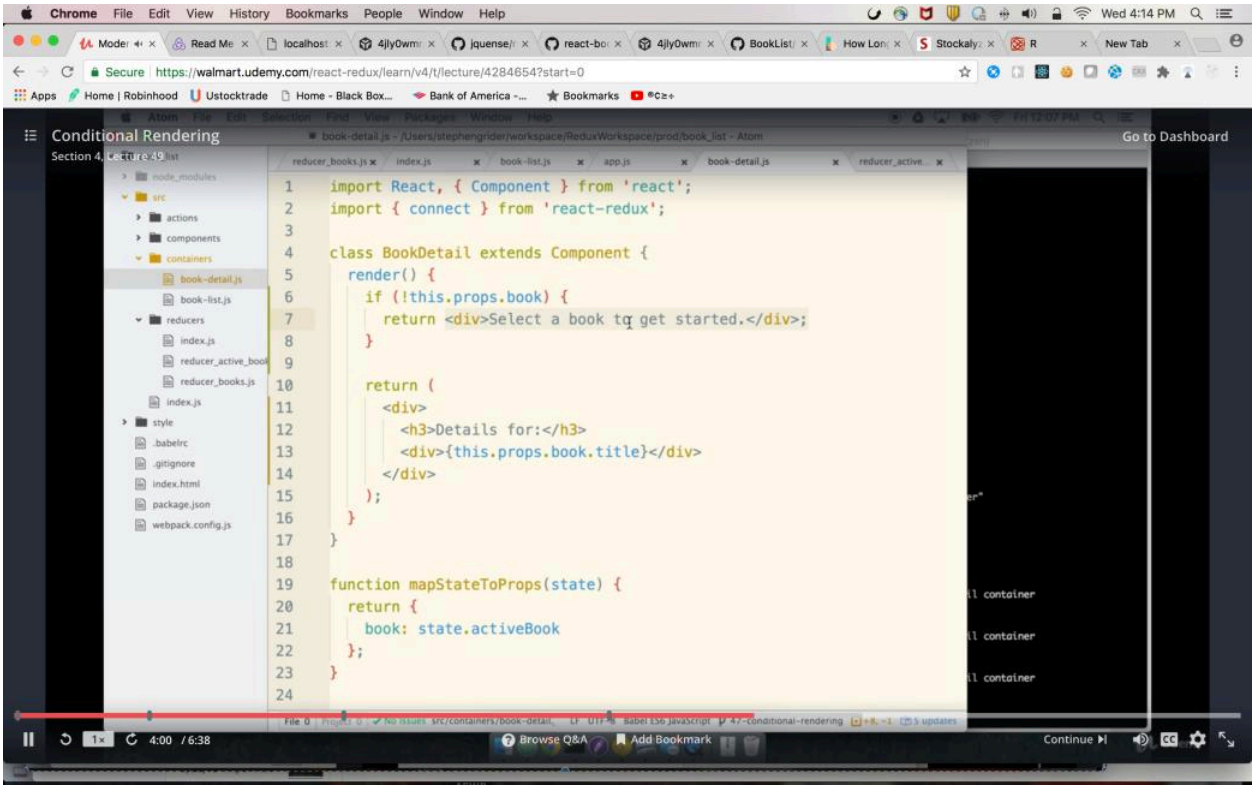
now we refresh the browser
and we get the error cannot read property ‘title’ of null

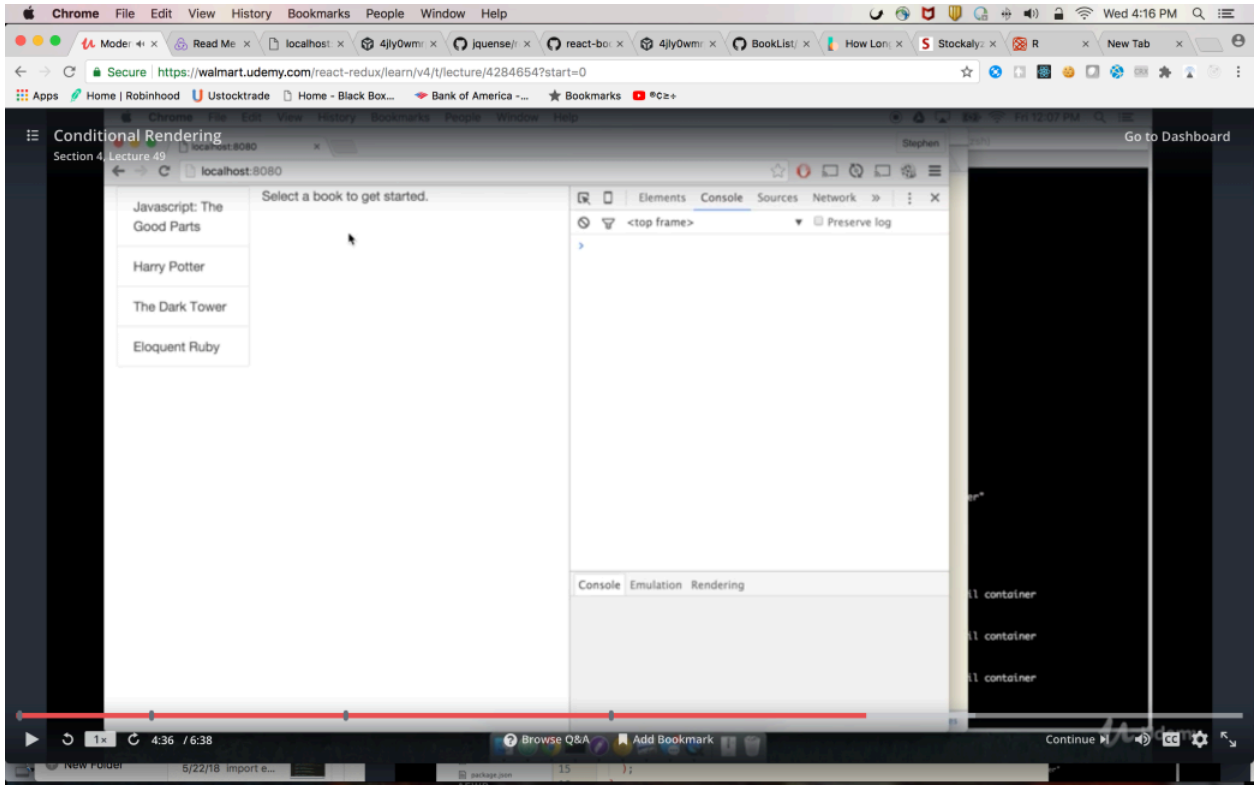


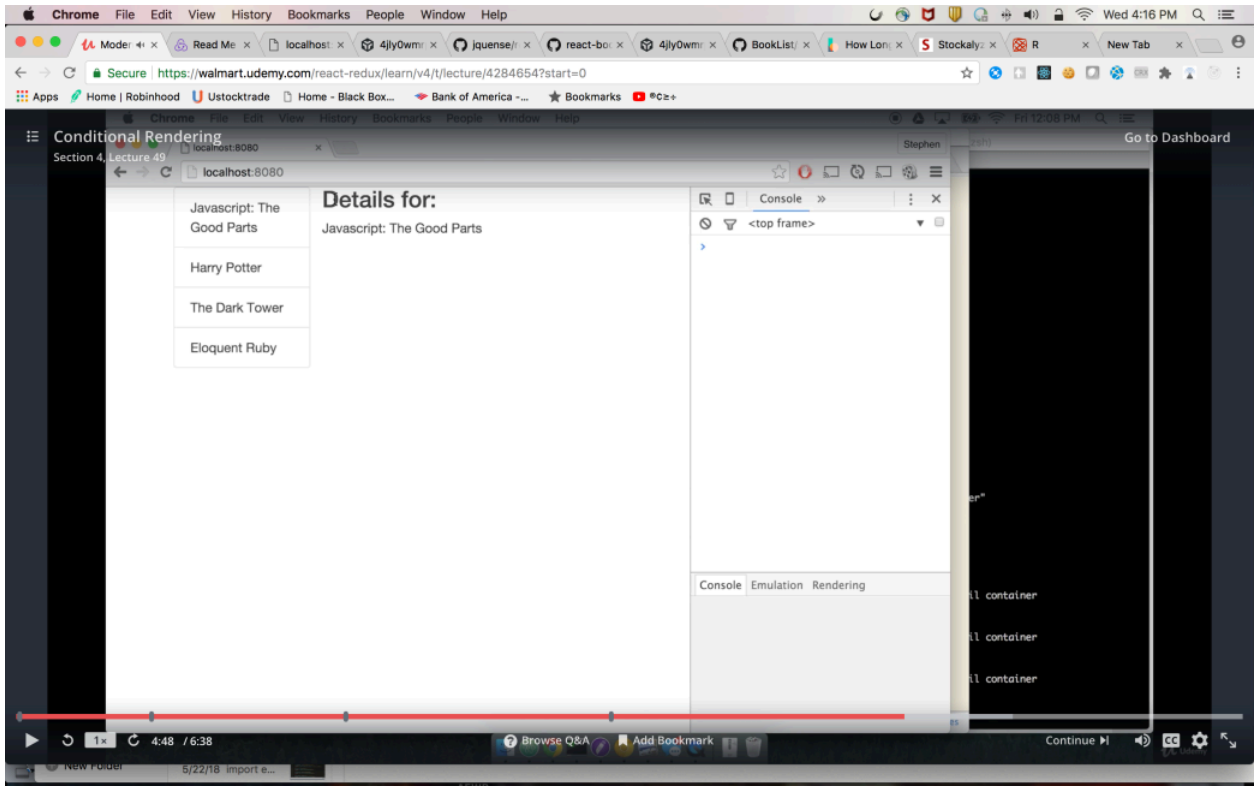
because at beginning when app is loaded there is action and state is default to null that is by this.props.book is null



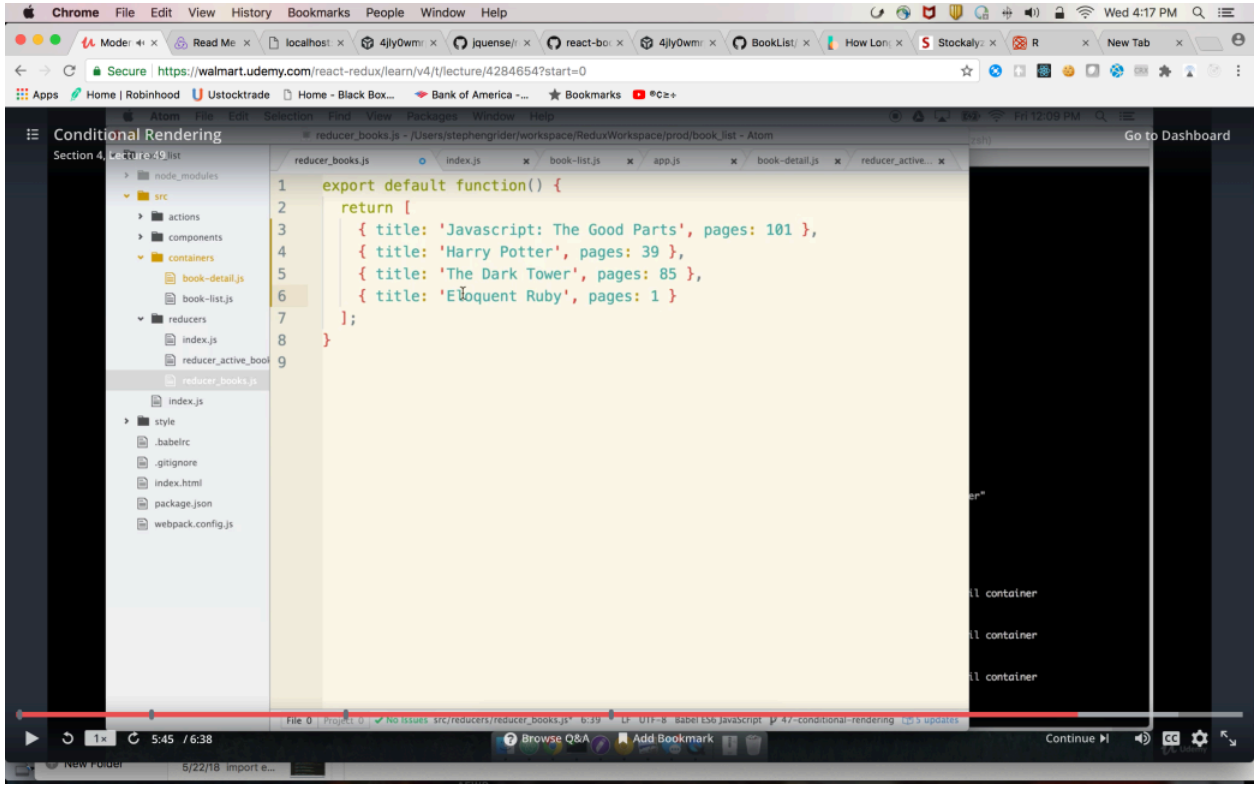
we will put a condition when this.props.book is null
and return message earlier



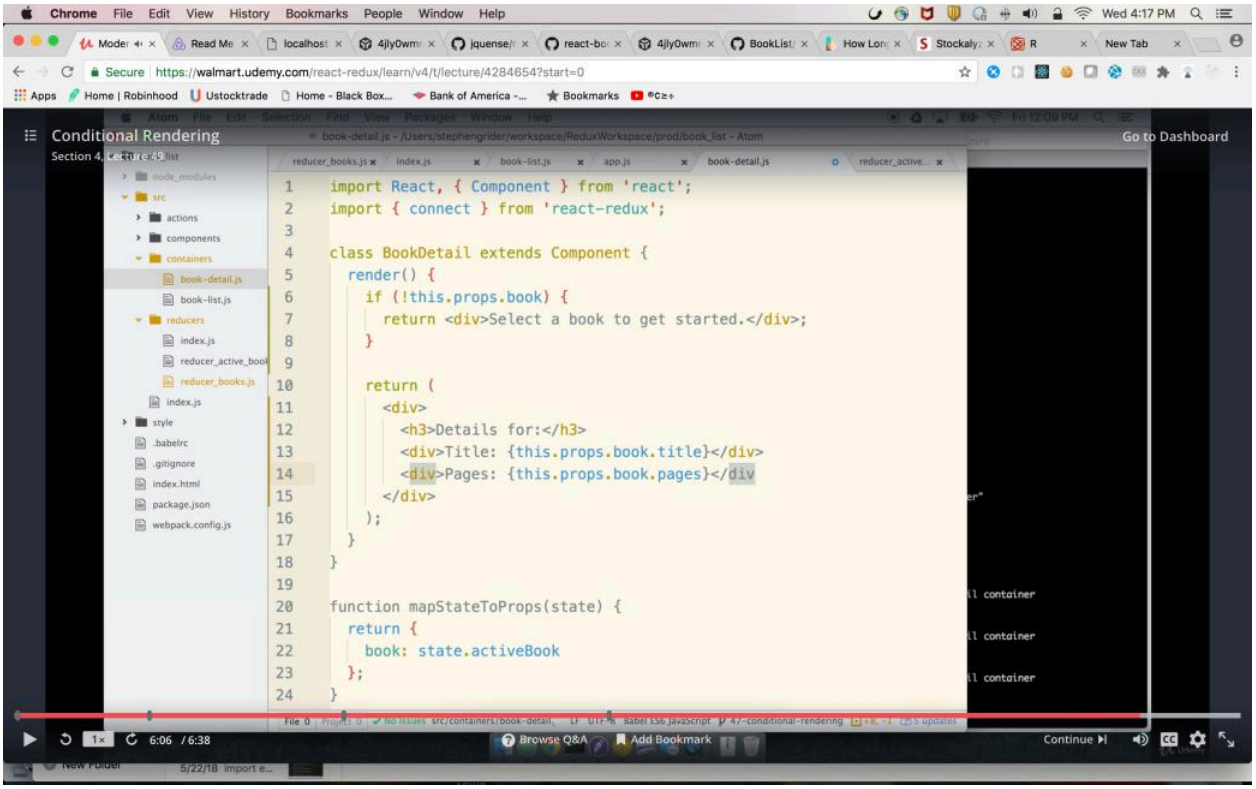

Chrome File Edit View History Bookmarks People Window Help
Wed 4:16 PM
Secure | https://walmart.udemy.com/react-redux/learn/v4/t/lecture/4284654?start=0
Apps Home | Robinhood Ustocktrade Home - Black Box... Bank of America -... Bookmarks
Conditional Rendering
Section 4, Lecture 49
Go to Dashboard
localhost:8080
Javascript: The Good Parts
Harry Potter
The Dark Tower
Eloquent Ruby
Select a book to get started.
Elements Console Sources Network
<top frame>
Preserve log
Console Emulation Rendering
1×
4:36 / 6:38
Browse Q&A
Add Bookmark
Continue

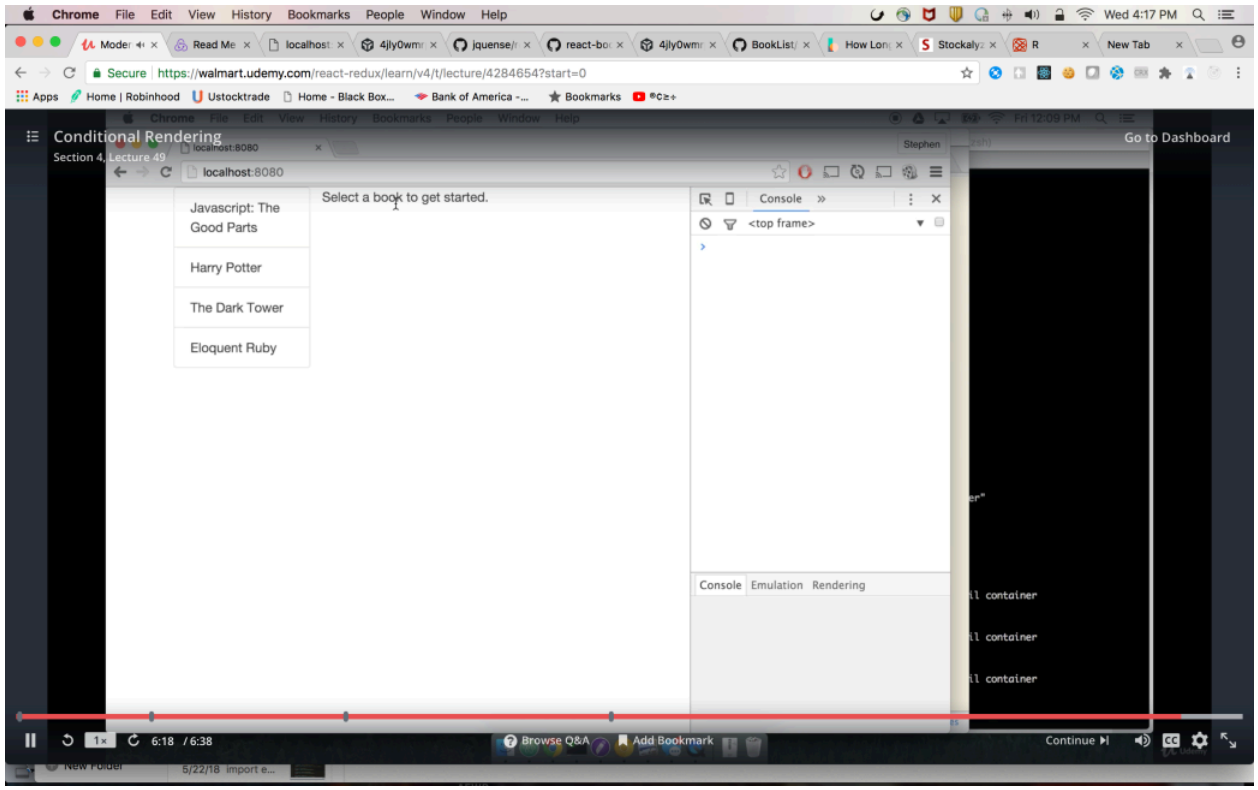


now add number of pages in reducer



number of pages in bookDetail component





click in li will result in