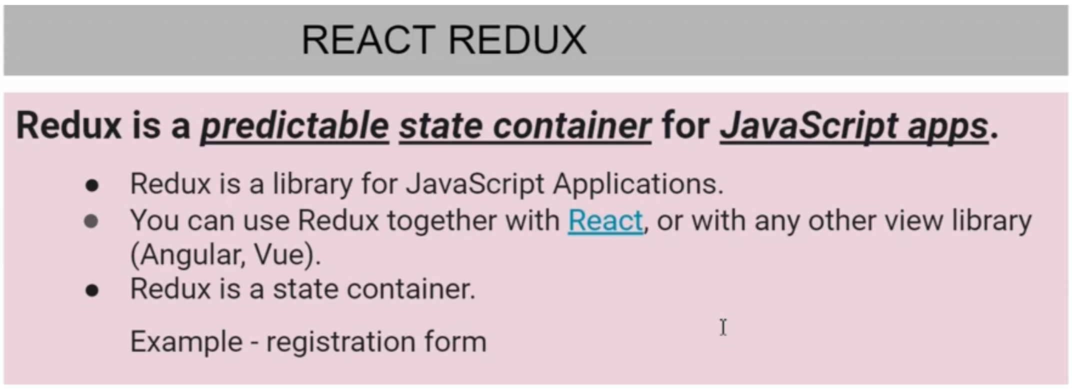
React Redux



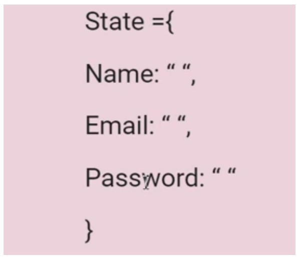
REACT REDUX

Redux is a *predictable* *state container* for *JavaScript apps*.

- Redux is a library for JavaScript Applications.
- You can use Redux together with React, or with any other view library (Angular, Vue).
- Redux is a state container.

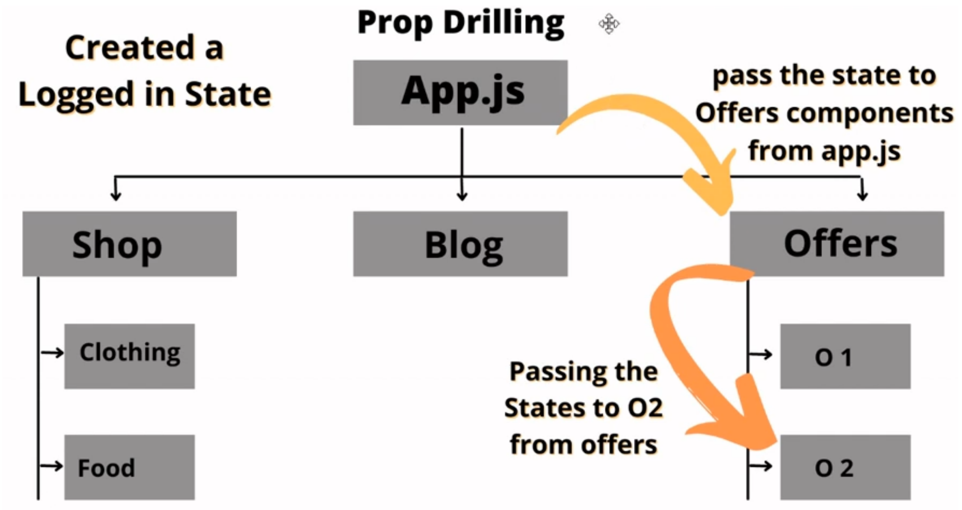
Example - registration form

```
=> state container ?
-- it is like store which contains all the states
```



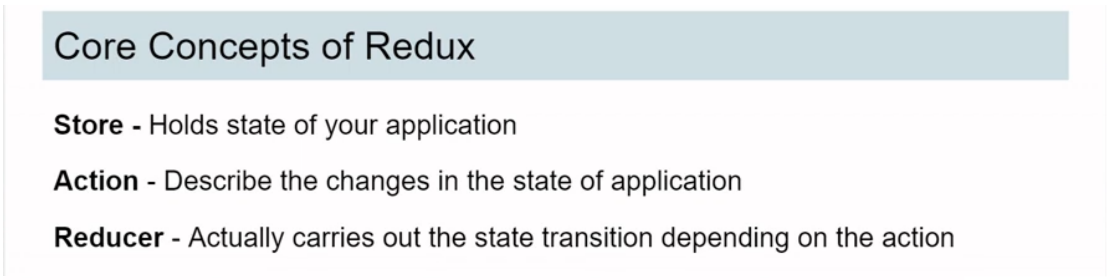
```
State ={
Name: " ",
Email: " ",
Password: " "
}
```

```
=> -- predictable ?
-- react contains track of the state management / or
contains track of the states

=> why we use ??
-- in big applications it is difficult to manage the states , like problems
to handle :
- prop drilling
- lifting state up
```


Prop Drilling
Created a Logged in State
App.js
pass the state to Offers components from app.js
Shop
Blog
Offers
Clothing
Food
Passing the States to O2 from offers
O 1
O 2

```
-- we were able to handle these problems with the help of context api and
useContext hook
=> then why redux ??
-- as redux was introduced first then these hooks were introduced and in bigger
applications it is required as we can't do everything with hooks
```

=> concepts of the redux

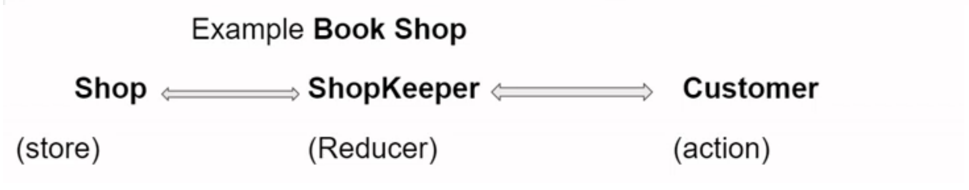


Core Concepts of Redux

Store - Holds state of your application

Action - Describe the changes in the state of application

Reducer - Actually carries out the state transition depending on the action

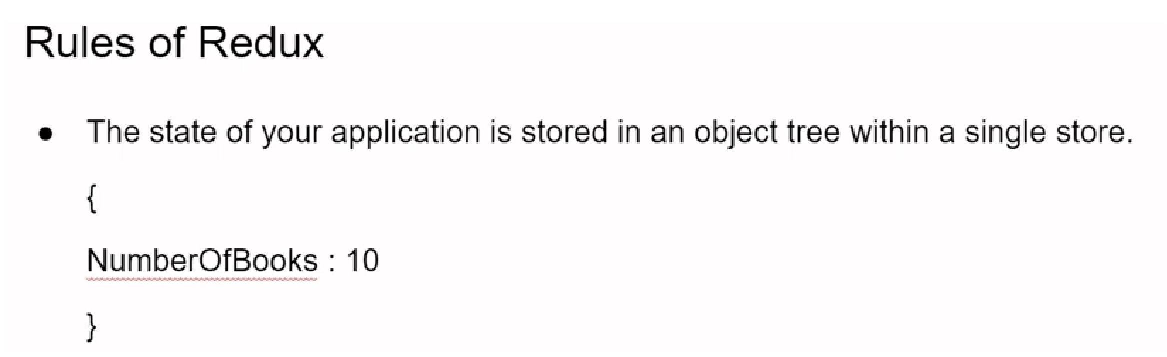


Example **Book Shop**

Shop ⟺ **ShopKeeper** ⟺ **Customer**

(store) (Reducer) (action)

-- whatever action you give to the reducer on the basis of that it updates the state of the store
-- reducer acts as a bridge between the action and the store
< reducer is a function which accepts the action as a parameter and on the basis of that updates the state >

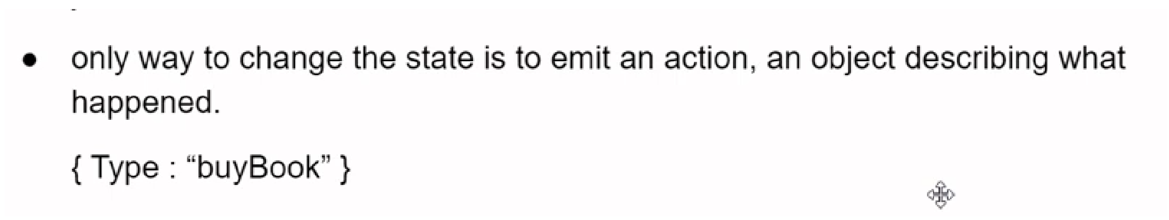


Rules of Redux

- The state of your application is stored in an object tree within a single store.

```
{
NumberOfBooks : 10
}
```

1> there will be a single object in the store and all the states will be defined in that object

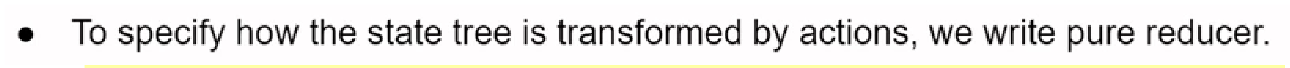


- only way to change the state is to emit an action, an object describing what happened.

  { Type : “buyBook” }

2> and state of the store can be only updated if we have some action < as we can’t directly update the states of the store .

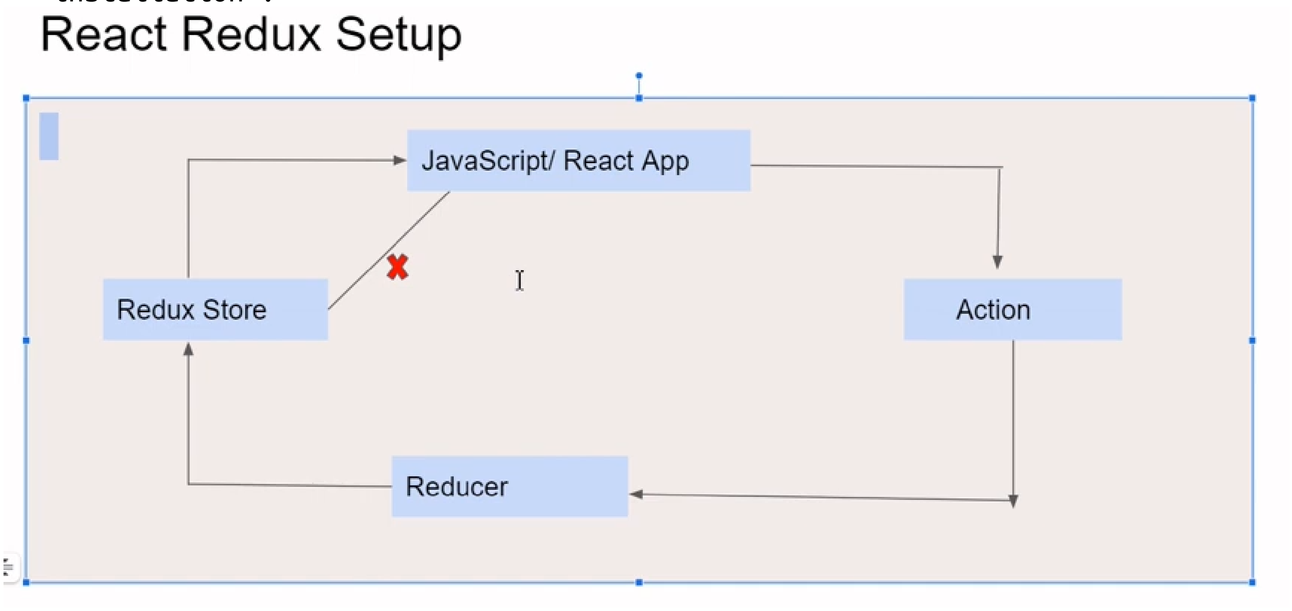
** action is an object which have a property called Type :
-- and type decide what type of action we have to perform with the states .



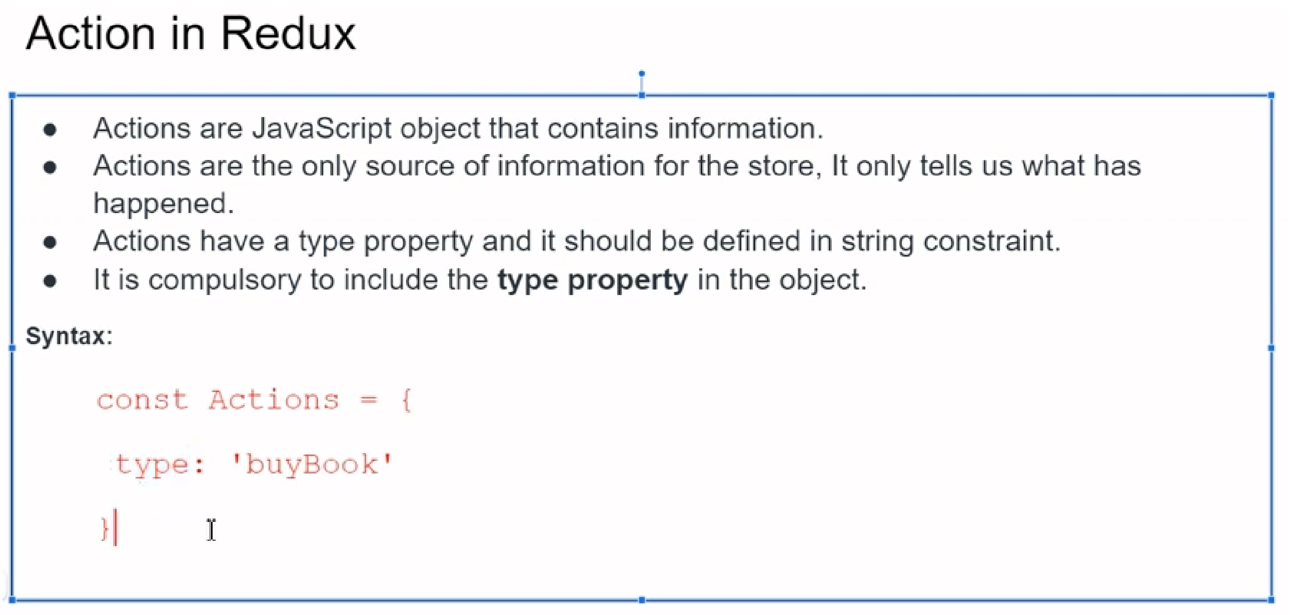
- To specify how the state tree is transformed by actions, we write pure reducer.

3> < reducer is a function which accepts the action as a parameter and on the basis of that updates the state > .

=> installation :


React Redux Setup

=> Actions :



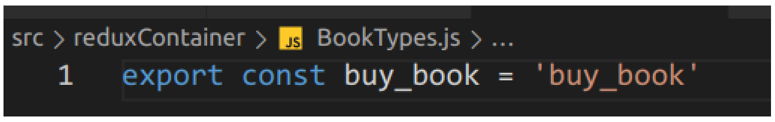
Action in Redux

- Actions are JavaScript object that contains information.
- Actions are the only source of information for the store, It only tells us what has happened.
- Actions have a type property and it should be defined in string constraint.
- It is compulsory to include the **type property** in the object.

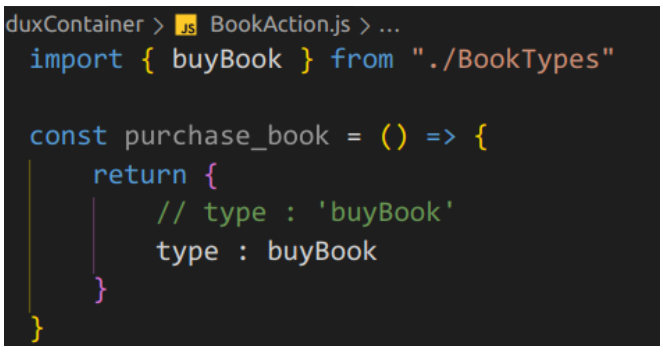
Syntax:

```
const Actions = {
 type: 'buyBook'
}
```

=> how we define actions in react/js app ?
-- BookAction is action creator and these action creators are the functions which returns the action and
-- and it's good to use const value for the type so instead of just giving the value/name directly first define it as const and export and import


src > reduxContainer > BookTypes.js > ...

```
export const buy_book = 'buy_book'
```

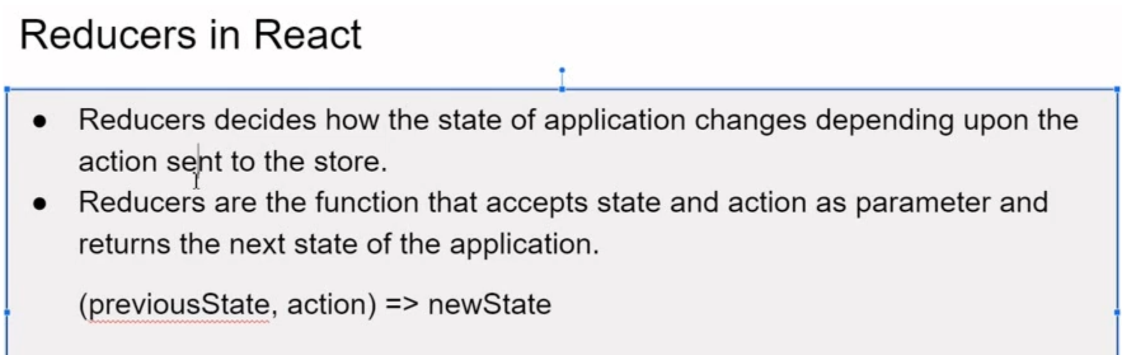

duxContainer > BookAction.js > ...

```
import { buyBook } from "./BookTypes"

const purchase_book = () => {
    return {
        // type : 'buyBook'
        type : buyBook
    }
}
```

-- it is imported in action creator
< action creators are the function which returns the action >

-- like here purchase_book is a function

=> Reducer :



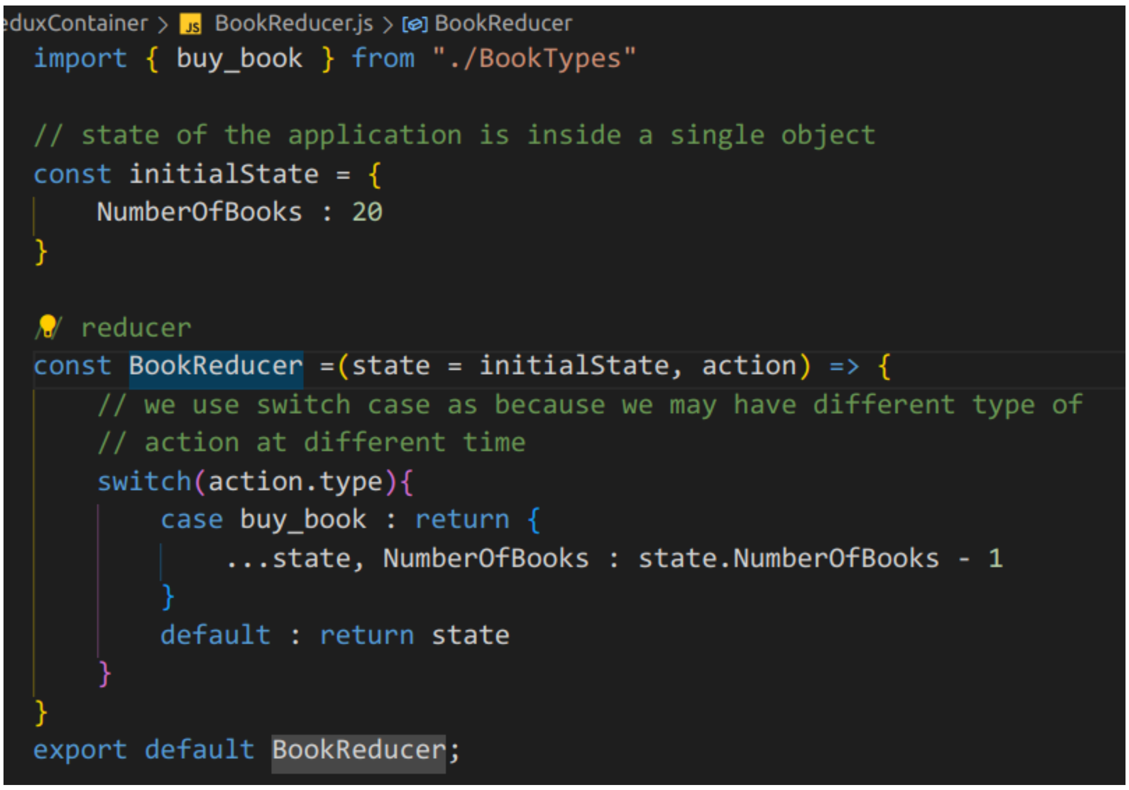
Reducers in React

- Reducers decides how the state of application changes depending upon the action sent to the store.
- Reducers are the function that accepts state and action as parameter and returns the next state of the application.

  (previousState, action) => newState

** action tells what change we have to do and reducer implements and tells how ? this can be change

-- reduceres are functions

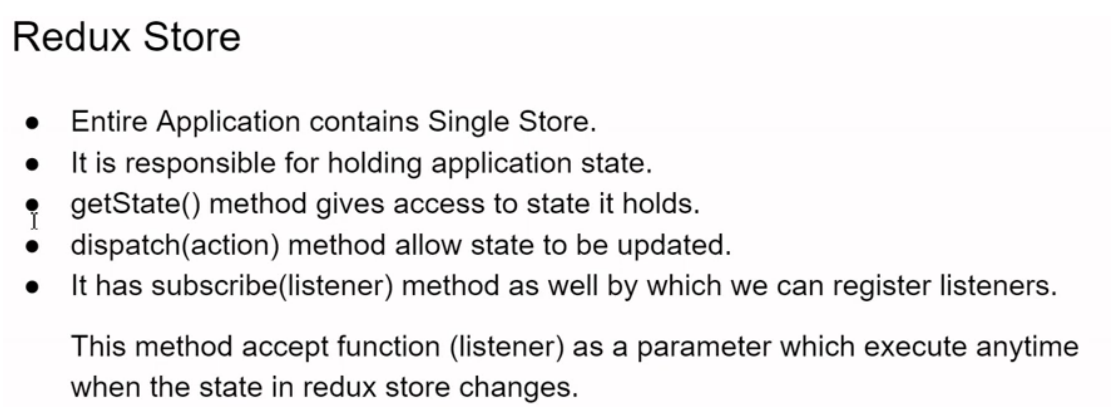


duxContainer > BookReducer.js > BookReducer

```
import { buy_book } from "./BookTypes"

// state of the application is inside a single object
const initialState = {
    NumberOfBooks : 20
}

// reducer
const BookReducer =(state = initialState, action) => {
    // we use switch case as because we may have different type of
    // action at different time
    switch(action.type){
        case buy_book : return {
            ...state, NumberOfBooks : state.NumberOfBooks - 1
        }
        default : return state
    }
}
export default BookReducer;
```

-- -- ------

=> Store in redux :
? how to connect our store with application

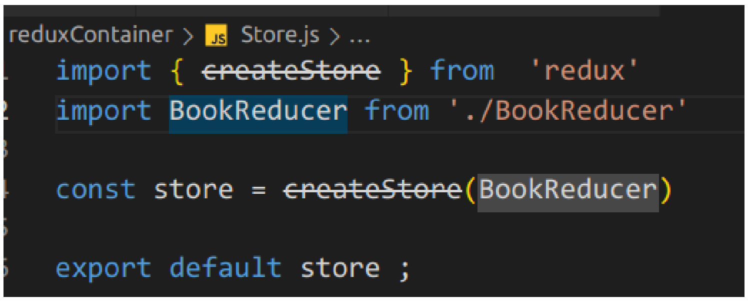


Redux Store

- Entire Application contains Single Store.
- It is responsible for holding application state.
- getState() method gives access to state it holds.
- dispatch(action) method allow state to be updated.
- It has subscribe(listener) method as well by which we can register listeners.

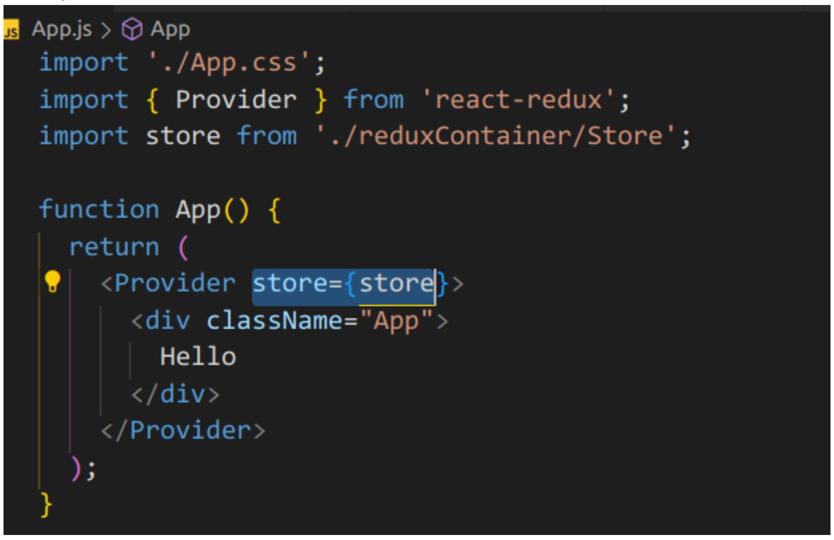
  This method accept function (listener) as a parameter which execute anytime when the state in redux store changes.

```
-- we can create the store using the createStore function
```



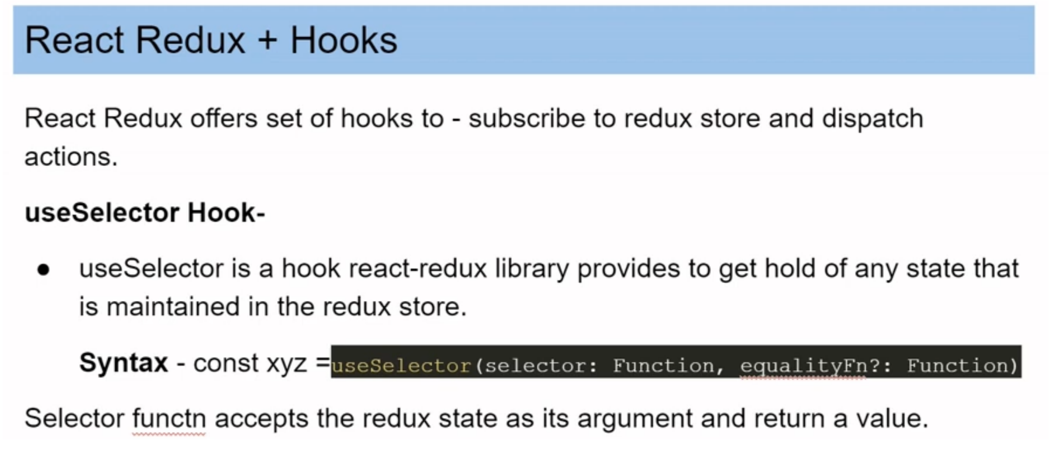
```
import { ~~createStore~~ } from  'redux'
import BookReducer from './BookReducer'

const store = ~~createStore~~(BookReducer)

export default store ;
```

```
** as we know that reducer contains the state of the app and when we pass the
reuducer in the createStore function then indirectly state is stored inside the
store of the app

=> how to connect this store with the react app : so here react-redux comes to
the picture .
```



```
import './App.css';
import { Provider } from 'react-redux';
import store from './reduxContainer/Store';

function App() {
  return (
    <Provider store={store}>
      <div className="App">
        Hello
      </div>
    </Provider>
  );
}
```

```
-- sometimes we also use configureStore instead of the createStore

-- ------------------------------------------------------------------------ --

REACT REDUX + HOOKS
-------------------
```



React Redux + Hooks

React Redux offers set of hooks to - subscribe to redux store and dispatch actions.

useSelector Hook-

- useSelector is a hook react-redux library provides to get hold of any state that is maintained in the redux store.

  Syntax - const xyz =`useSelector(selector: Function, equalityFn?: Function)`

Selector functn accepts the redux state as its argument and return a value.

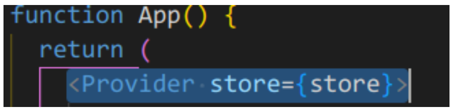
```
=> when hooks were not available then we were using some methods like connect()
=> but now react-redux library gives some hooks as well
```

1> useSelector hook :

- we can take a hold / access the state of the store of the application with the help of the useSelector

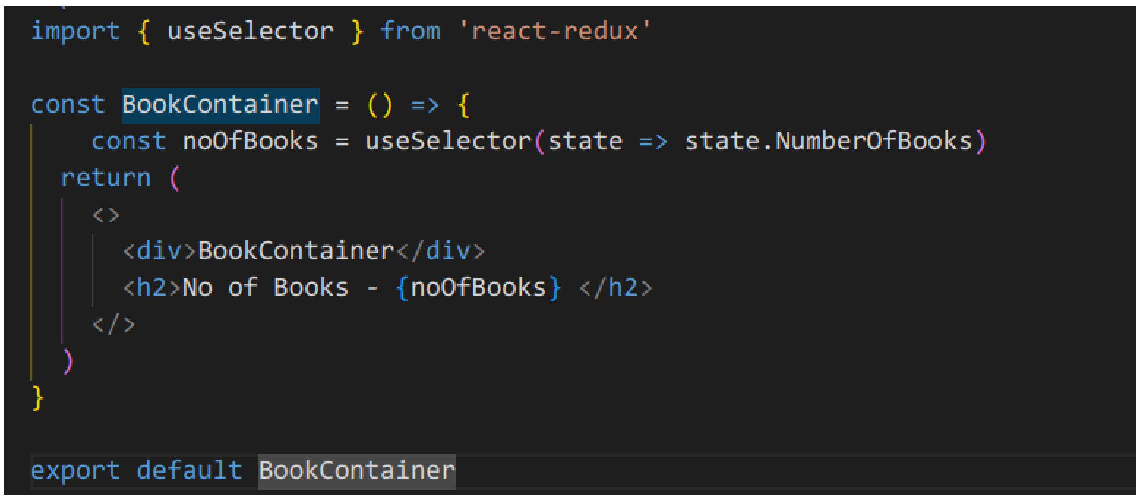
- it accepts a function <called selector function> as a paraemeter and returns a value and this selector function
  -> takes state of the redux and return the value

-- till now we have actions, store and reducer and our entire app is wrapped with provider so all the components in app will be able to access the store



```
function App() {
  return (
    <Provider store={store}>
```

- for using useSelector we make one file BookContainer :

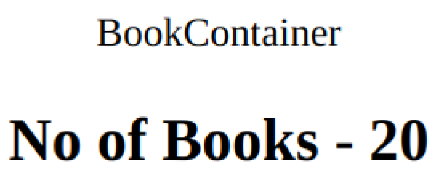


```
import { useSelector } from 'react-redux'

const BookContainer = () => {
    const noOfBooks = useSelector(state => state.NumberOfBooks)
  return (
    <>
      <div>BookContainer</div>
      <h2>No of Books - {noOfBooks} </h2>
    </>
  )
}

export default BookContainer
```

-- useSelector function accepts a selector funtion and here
state => is a selector function and it is returning noOfBooks using the state

** and the value returned by the selector function is returned by the useSelector hook as well

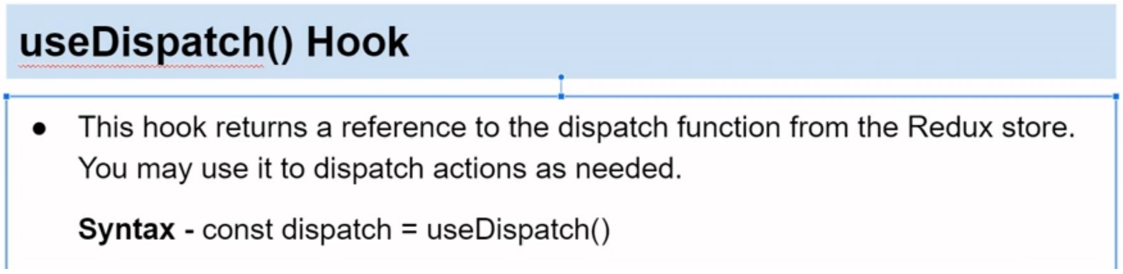
-- and since BookContainer is inside app so it can access the state
-- and as you can see that we are able to access the state with useSelector



BookContainer

No of Books - 20

=> useDispatch hook :



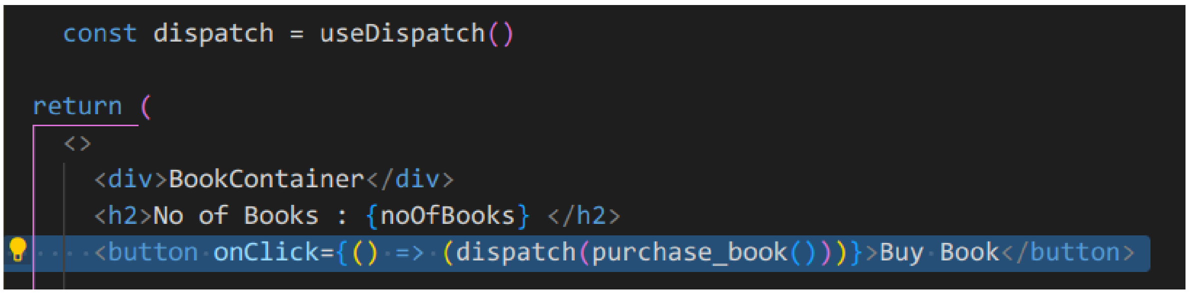
useDispatch() Hook

- This hook returns a reference to the dispatch function from the Redux store. You may use it to dispatch actions as needed.

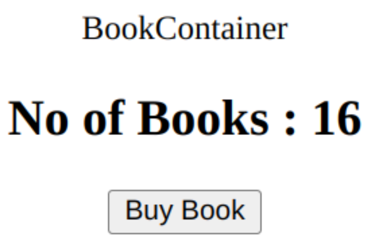
  Syntax - const dispatch = useDispatch()

-- we were able to access the state of the store with the help of useSelector and now if we have to update that state then here useDispatch comes into the pic

** [redux store have a dispatch function , useDispatch hook returns the reference of this dispatch function and whenever we want to dispatch the actions in any component we can use this reference]

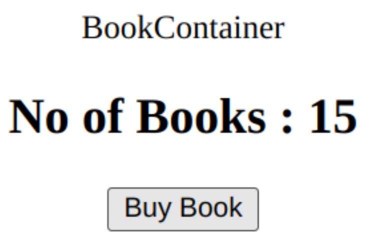


```
  const dispatch = useDispatch()

return (
  <>
    <div>BookContainer</div>
    <h2>No of Books : {noOfBooks} </h2>
    <button onClick={() => (dispatch(purchase_book()))}>Buy Book</button>
```



BookContainer

No of Books : 16



BookContainer

No of Books : 15

-- so now we are able to update the state of the store of