# REACT

## What is react?

* A javascript library for creating user-interface.
* Popular library used for creating Single Page Applications.

## REACT Component

* The reusable html structures (user-interface) in React are called components.
* Think of a component as a user-interface that we see on the screen, like a search widget,

form, etc.

## Types of Components:

|  |  |
| --- | --- |
| Functional Components | Class based components |
| * Created as a JS function. | * Created as ES6 class that extends from React Component |
| * Used when the only job of the component is to display some static data. | * Used when we want more capabilities from the component, use life cycle methods, add addition methods. |
| * Functional components do not have state | * We need to use class based components if we want to use states. |
| * Props are passed as arguments. | * Props can be used as ‘this.props’ |

## Named Exports:

* Exporting an object without ‘default’ keyword means that we are doing named export.

For example:

export comp1;

export comp2;

Now we have to import them using their exact names as:

**Import { comp1, comp2 } from ‘./somefile’**

Alternatively if you export as the default export like this,

**export default class Template {}**

Then in another file you import the default export without using the {}, like this,

**import Template from './components/templates'**

**There can only be one default export per file**. In React it's a convention to export one component from a file, and to export it is as the default export.

You're free to rename the default export as you import it,

import TheTemplate from './components/templates'

And you can import default and named exports at the same time,

import Template,{AnotherTemplate} from './components/templates'

## REACT State

* A state is a plan JavaScript object used to record and react to user events.
* It usually contains some data about the component.
* Each class based component in React has its own state
* Functional components do not have state.
* Whenever a component state is changed then the component (along with all its children component will be re-rendered)
* Each instance of the class based component has its own copy of the state
* We have controlled fields in our application where the value of the element is set by the react state.

### Initializing State:

* A state can be initialized inside a class Constructor:



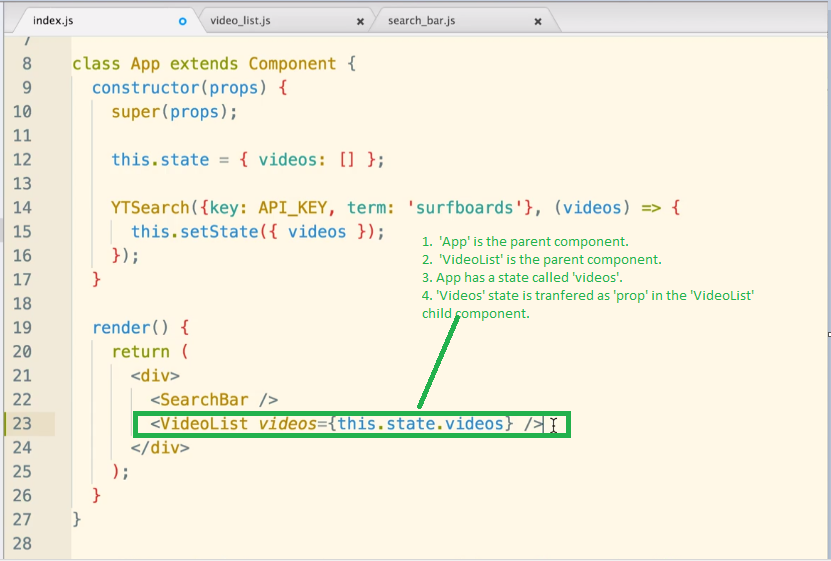
* A class constructor function is the first function that is called automatically when ever the new instance of the class is created.

## Updating State:

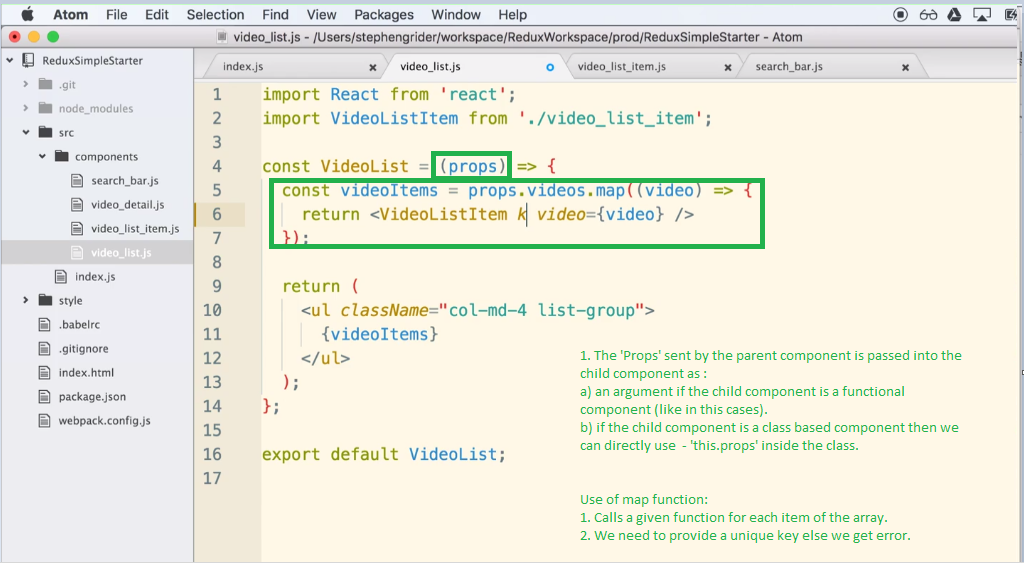
* We can update the state using ‘**this.setstate ( {term : ‘<new\_value>’})**’. This is the only way we should ever update our class’s state.

## PROPS:

* Props is way of passing data between from parent to child component in React JS.



* In a functional component, the ‘props’ can be passed in an arguement when starting the function.



* In class based component, the ‘props’ can be used by just refering to ‘this.props’.

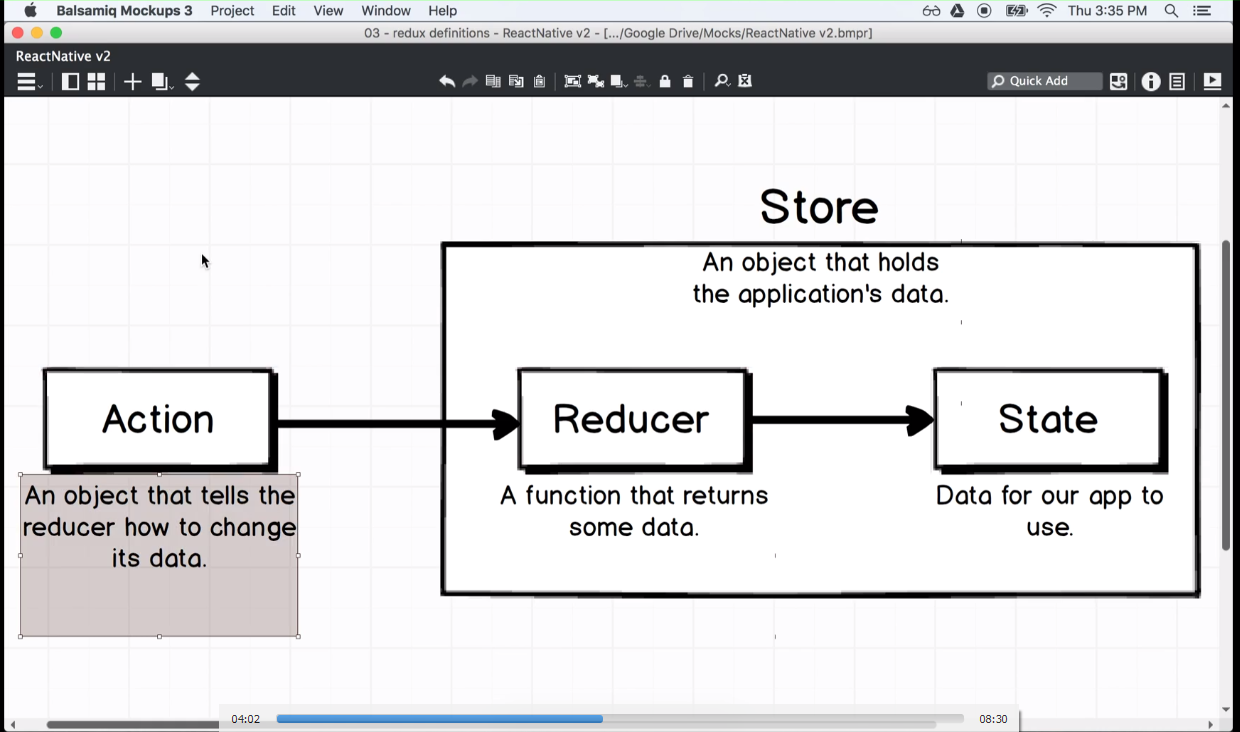
## Passing Callbacks between Parent to Child for communication

# REDUX:

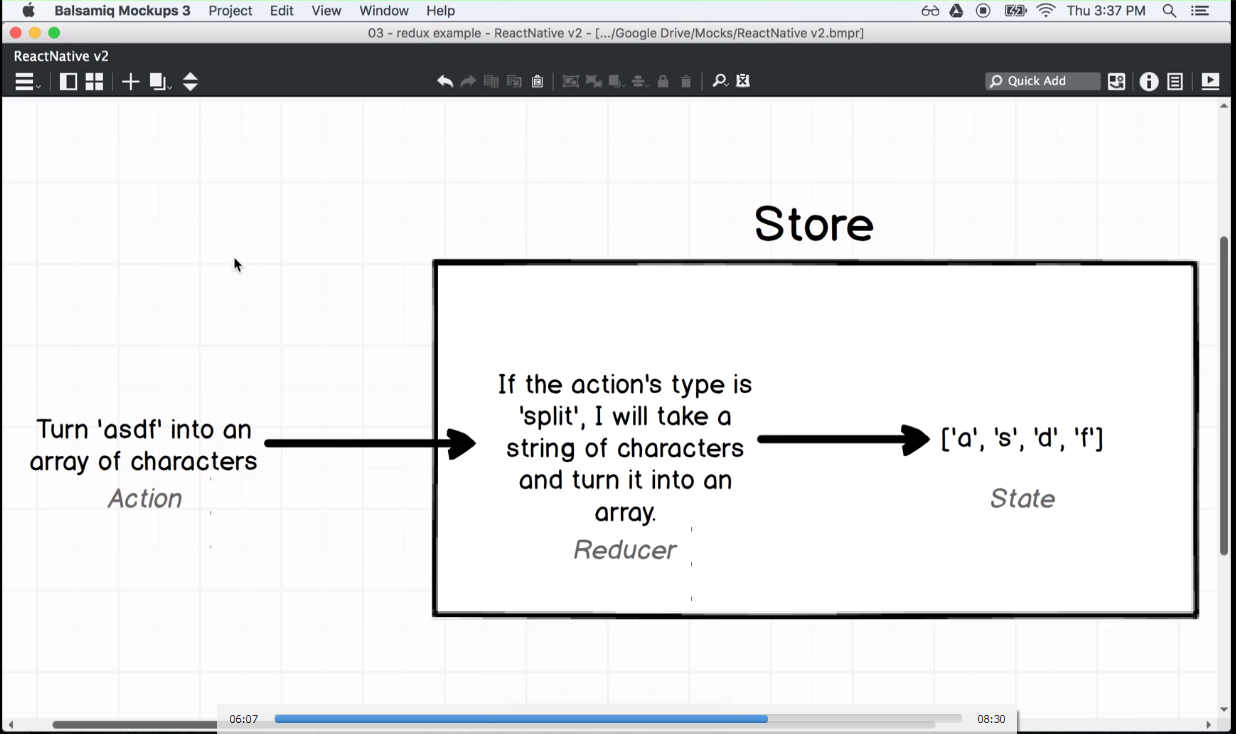
* Redux is the data container of our application.

## How Reducer works:

Different components of Redux



Example of this architecture:

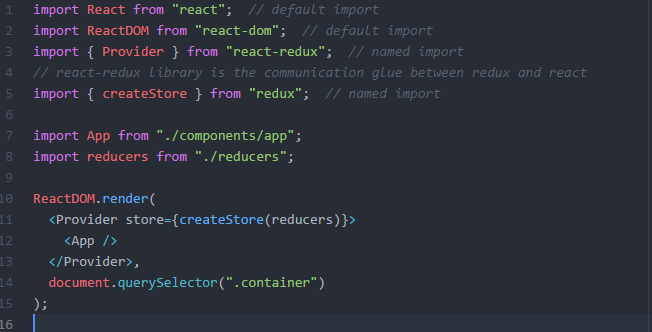


How to connect React with Redux:

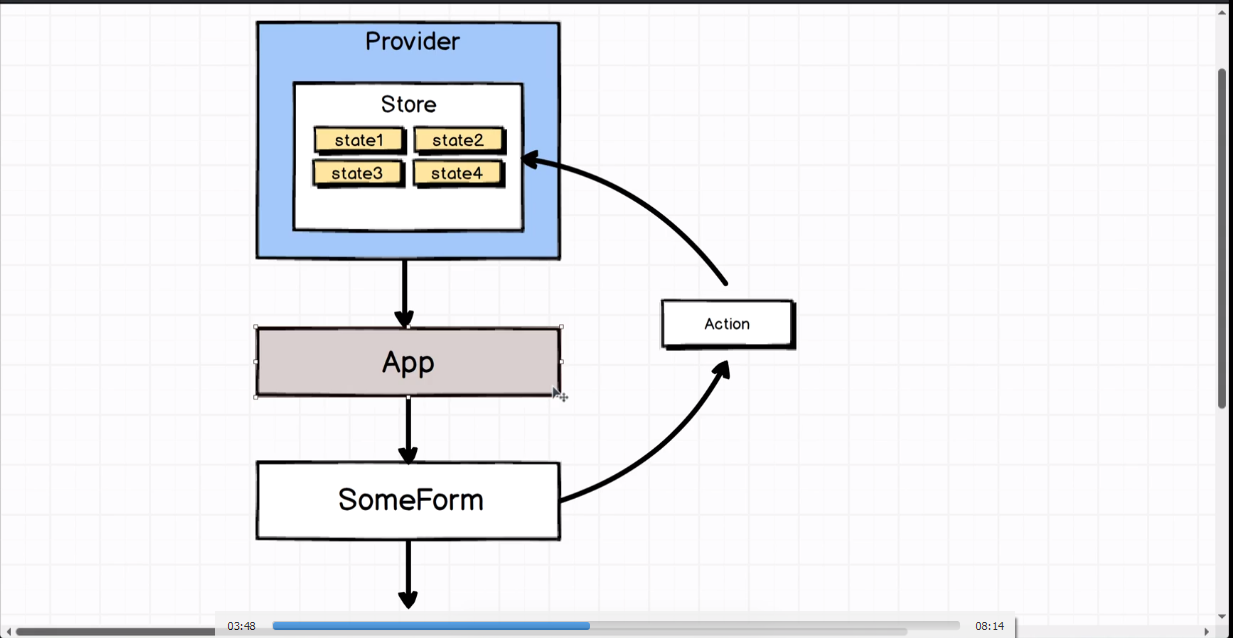
We can do this in our root app.js file by using following things:

* Provider from ‘react-redux’ : Helps to connect our redux store with our react app. This is the main communication glue with React.
* CreateStore method from ‘redux’ library takes all the reducers that we have.

Code Example:

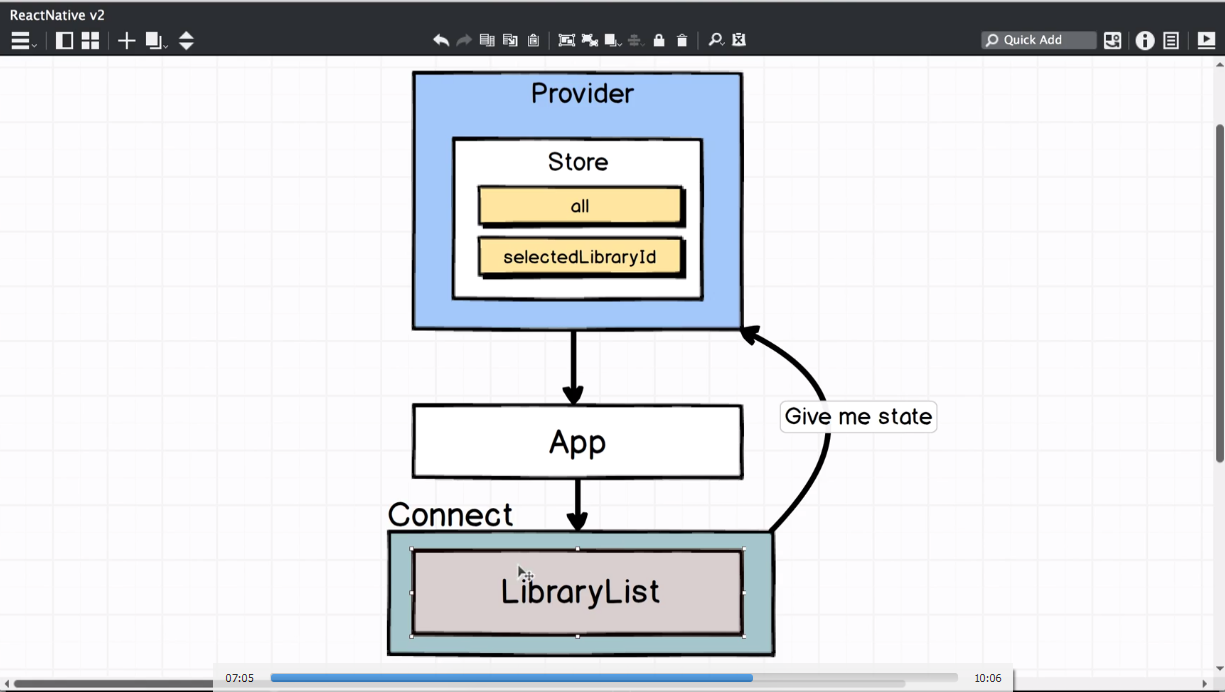


Flow example:



Connect Helper method:

* The state produced by the reducers is present inside the redux store.
* We need some way to help our components to use this global state data.
* Connect methods helps in this. Its a tool to connect our component to our redux store.



* Syntax for Connect:

import {connect} from ‘react-redux’

....

...

..

.

export default connect()(<our component to be exported>)

// runs the connect method that returns another method, which is executed using the arguments in the second brackets

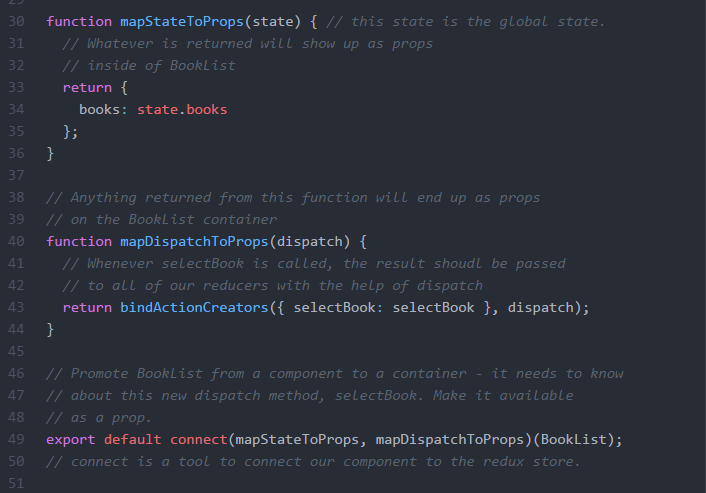
## mapStateToProps:

* This method takes in the global state and returns a js object that can used in our component as props.
* This method helps to use the date in the redux store state in our component.

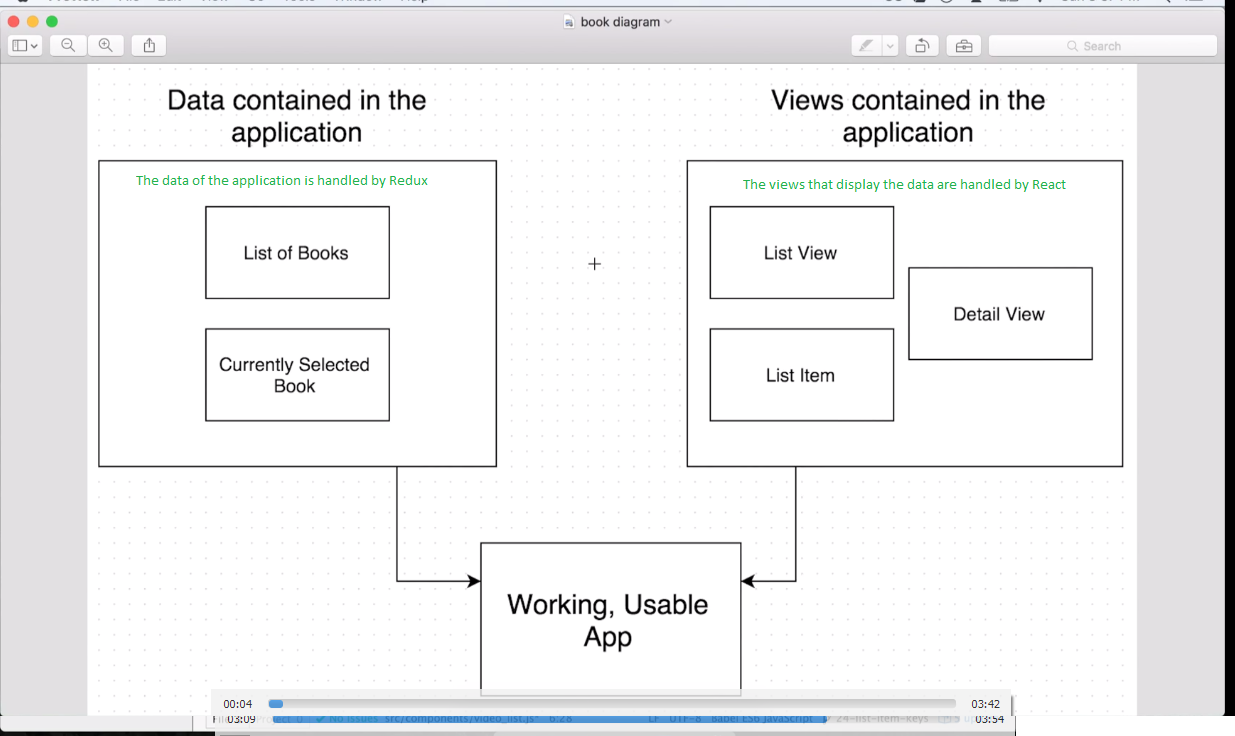
## mapdispatchToProps:

It performs two functions:

* dispatches our action creator to all the reducers.
* Helps to use our action creator inside the component as a prop.

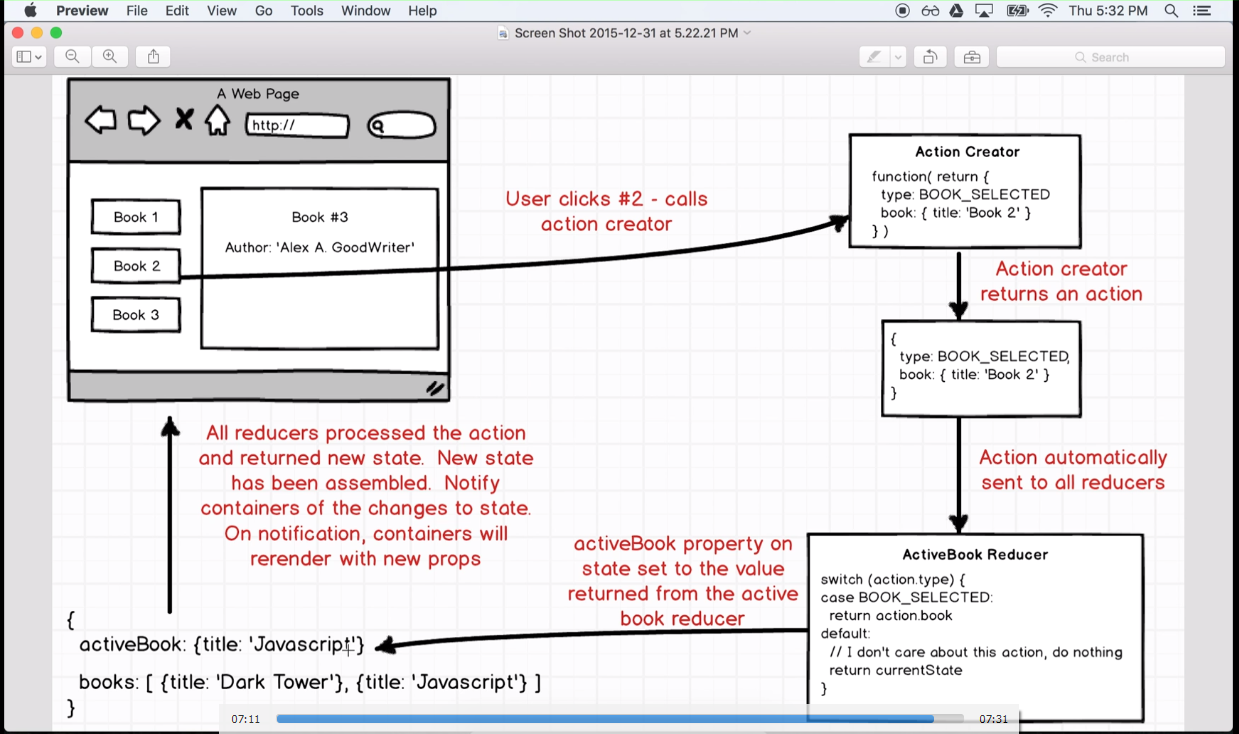


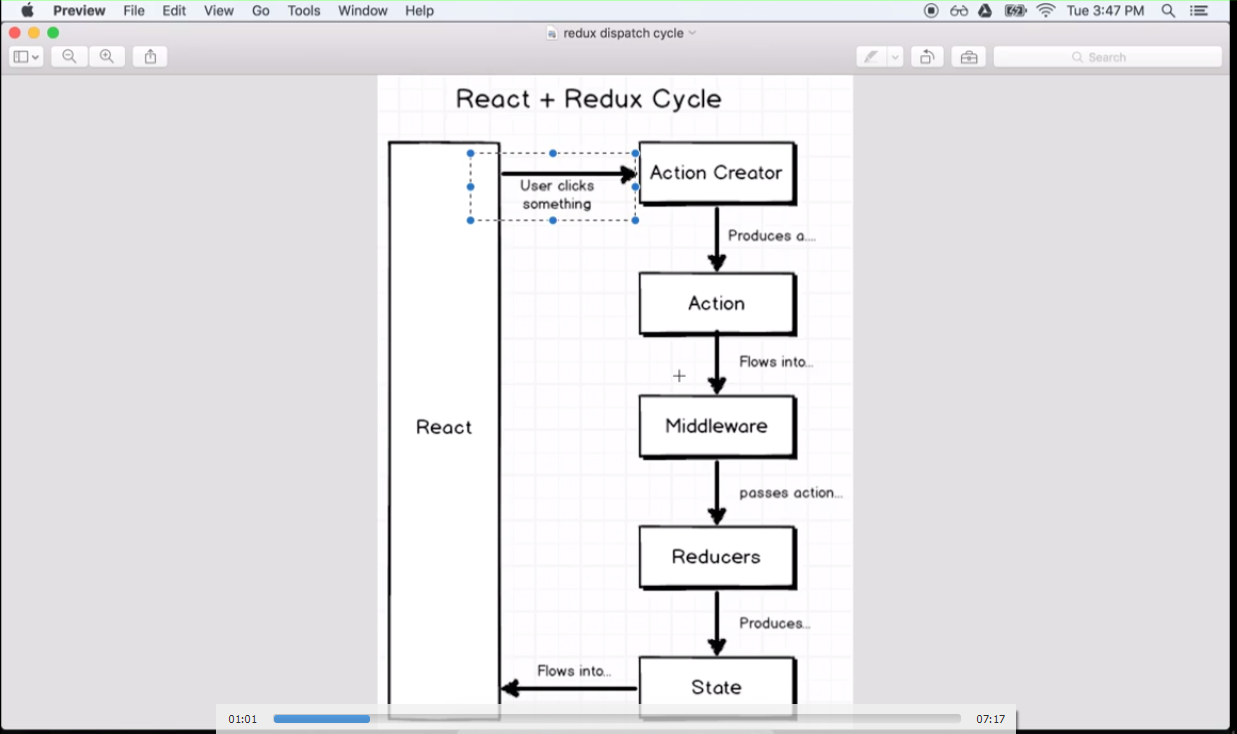
* Consider the example of a website showing list of books in the left and details of the currently selected book on the right. React and Redux can be used together to build this application using following:



* So redux is the collection of all the data the describes the app.
* It not only contains the hard data (like list of books) but also shows stores meta data like the currently Selected book.

Redux Architecture:





1. User performs some action on the application.
2. Action Creator will be called. An action creator returns a JavaScript object with :
3. Type of action performed with the key ‘type’. This is mandatory.
4. Any additional data for that action. This is optional.
5. Optionally, this action can also be passed to some middlewear for futher mainpulation.
6. This javascript object returned by the ‘Action Creator’ is automatically sent to all the ‘Reducers’.
7. Reducers are functions return some application State.

We use switch block to check what kind of action is performed and return state based on the action.

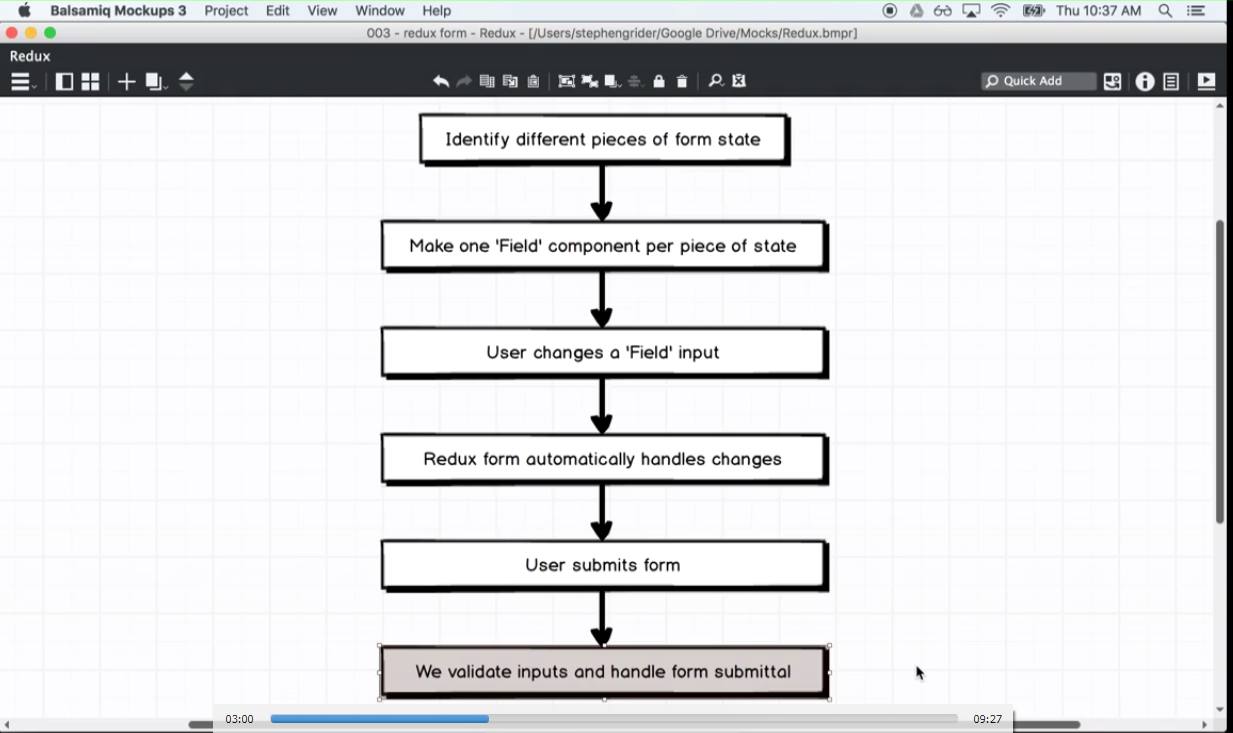
1. Now we want to be able to use this updated state in our ‘react’ application.
2. The components that use our redux state are known as ‘containers’.
3. We create a new directory of containers and add the component to it.
4. React and Redux are completely independent of each other. So we need some tool to connect them with each other.
5. ‘react-redux’ library is used to connect react with redux.
6. We import the ‘connect’ method from ‘react-redux’ library and then wire up the things.

REDUX FORM:

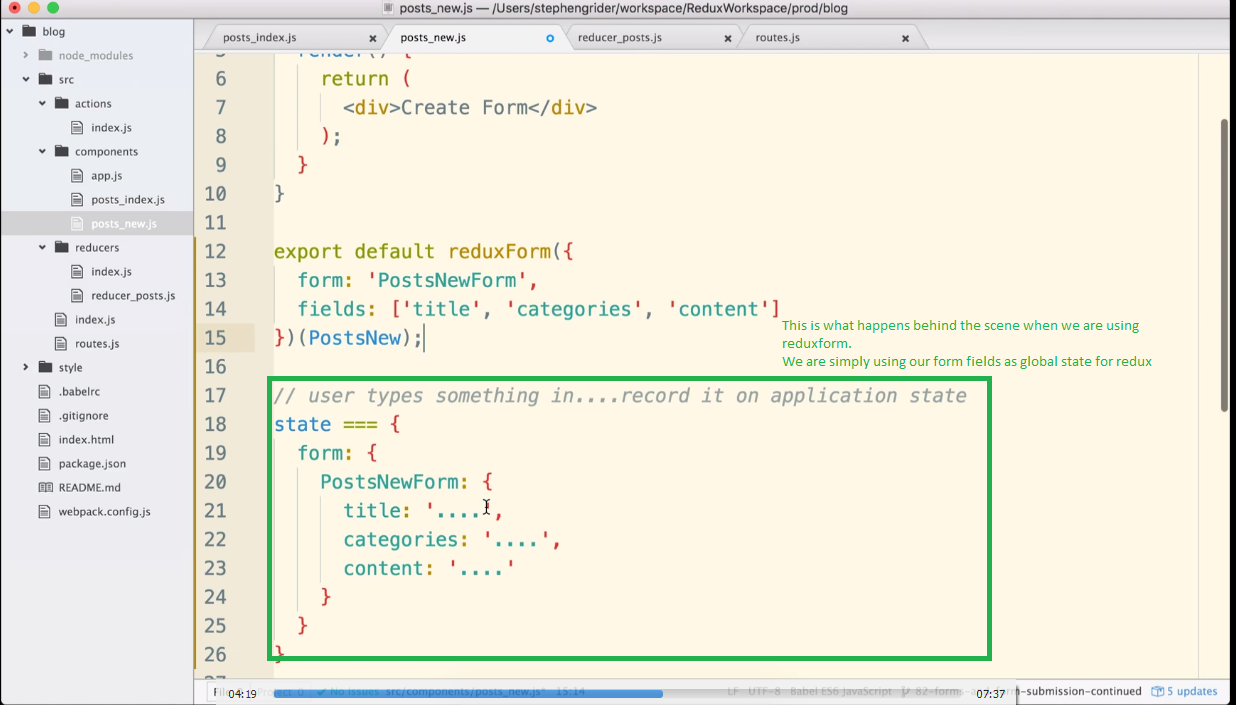
* Library for working with Forms in Redux.
* Provides form validation.
* Redux forms enables our form fields to be used an application level state.

1. Redux form behind the scenes:





#### reduxForm helper



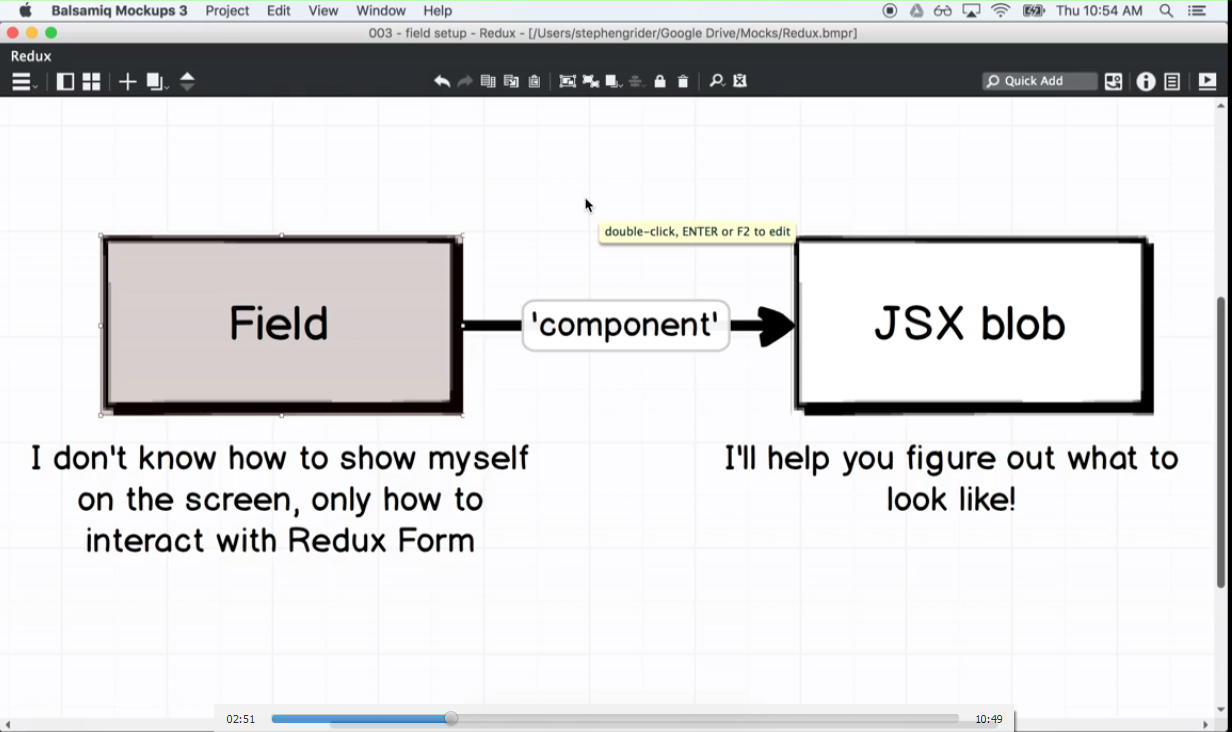
* Similar to ‘connect’ method, ‘reduxForm’ helps to connect our Fields to the Form and provides some properties on our props:

1. this.props.handlesubmit:

* is called when the user tries to submit button by clicking on submit or pressing enter key.
* It first validates the form, then calls the method passed to it (usually an action creator) to post the data.

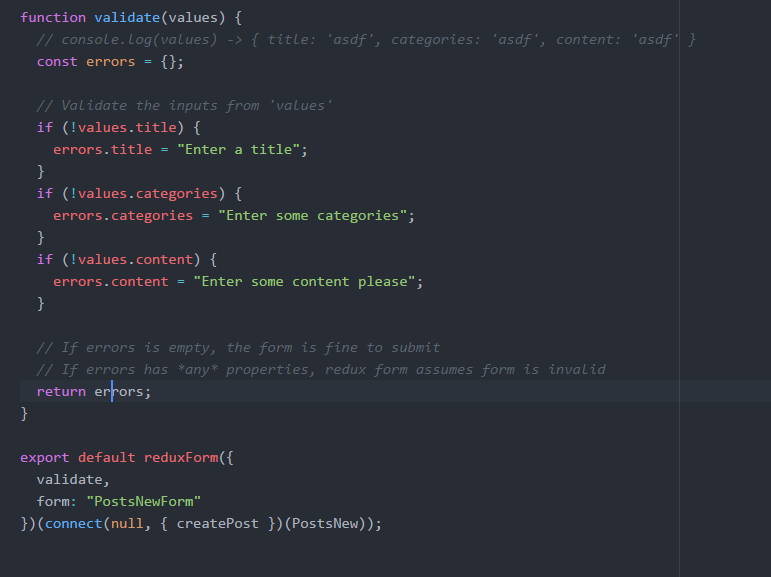
#### Field Component:

* Field Component from redux-form is for communication with the redux form.
* The ‘component’ prop added to the ‘Field’ component helps to create the actual form input that the user can interact with.



Form Validation:

1. Create a method that takes a ‘values’ attribute and returns an ‘errors’ js object.
2. This method name should be added in the form config in ‘reduxForm’ helper
3. If ‘errors’ object has no properties when returned then there are not errors in form validation and form is submitted succcessfully.
4. If there are properties in ‘errors’ object then it means that there are errors and form validation is stopped.



1. this.props.fields : All the feilds that we mentioned in the reduxform help at the bottom.

Using mapstatetoprops and mapdispatchtoprops with reduxForm helper:

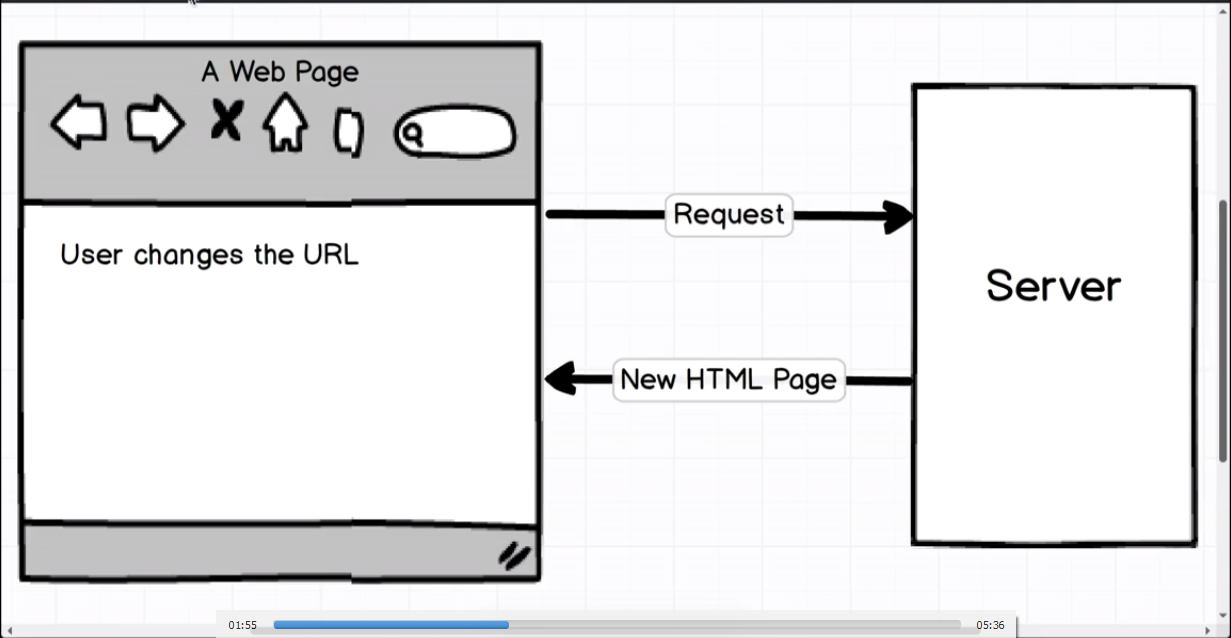


# React ROUTER:

* Installation - npm install --save [react-router-dom@4.0.0](mailto:react-router-dom@4.0.0)

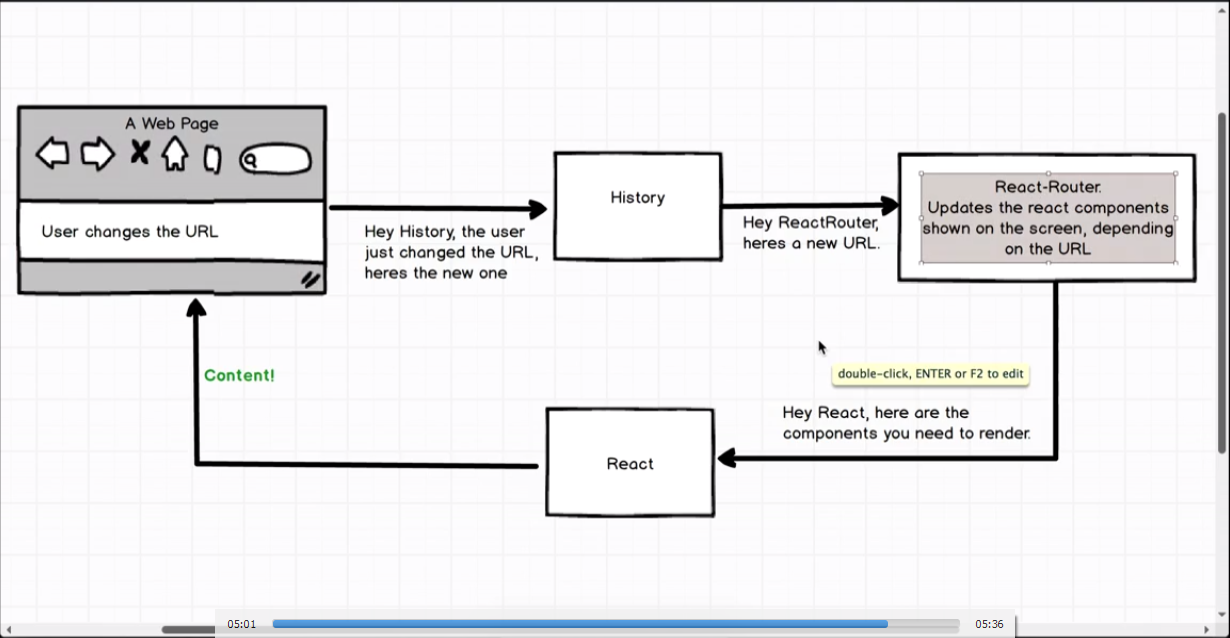
How routing works?

* Traditional it used to work like this:



1. User would navigate to a URL (by clicking on the link)
2. This would generate a request to the server.
3. Server will respond back with an HTML page.

How REACT routing changed this traditional approach?



1. Whenever the user changes the url, the history library (a part of React Router), checks the change in the URL>
2. History then sends the URL to ‘React Router’.
3. The react router checks which components we want to show on the screen.
4. The react rounter informs the React about these components.
5. React then shows those components.
6. This is also the idea behind Single Page Application

Browser Router:

* Interacts with the history library to decide what to do with the changed URL.

### Route:

* A react component that shows component based on the URL.
* The Route Component takes ‘path’ and ‘component’ props to decide component to be displayed for a particular path.
* Its important to wrap the <Route>s inside a <div> as we can have only single element inside a Route.

### Switch:

* The switch component takes in a collection of different routes, that is, routes are nested inside a switch component.
* Switch component looks into each route inside and only routes the first component that matches the current URL.
* So we need to put our most specific routes on the top of the list of routes.



* To work with a dynamic path, we can add wildcards in the path.

<Route path=”/somepath/:id” component={<ComponentName>} />

* We can programatically navigate to a url and send some state to that new URL as well using:

**this.props.history.push({pathname: ‘’ , state: { } });**

## Creating Hyperlinks using React Route:

* For this we need to use the ‘Link’ component from React-Router-dom and then enclose the

Hyperlink within the <Link> component.

import { Link } from "react-router-dom"

<Link to={`/posts/${post.id}`}>

{post.title}

</Link>

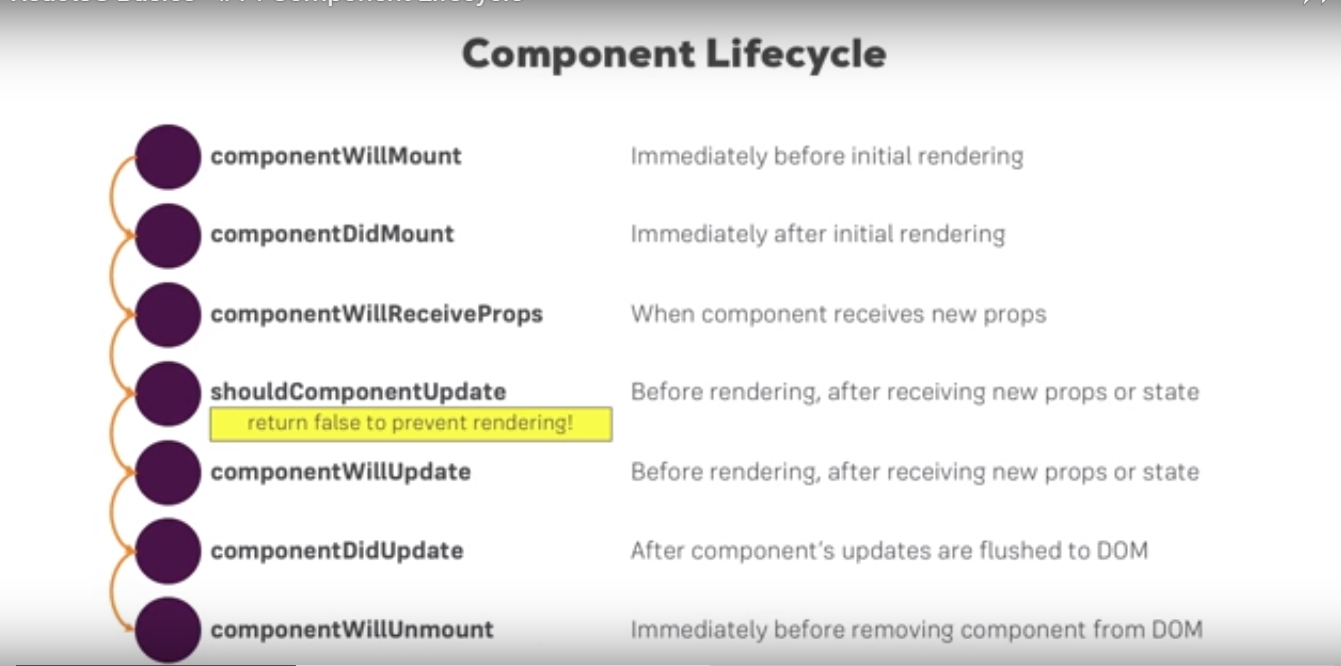
// Here we used Link to replace the button tag.

<Link className="btn btn-primary" to="/posts/new">

Add a Post

</Link>

# React LIFE CYCLE METHOD:



Idea behind React?

What is virtual dom?

How react interact with the DOM?

Aysnchronous tasks in React?

Call back functions in JS?

Promise in JS?

Redux Promise?

Redux Forms

React Routing.

What is callback binding?

What is callback function?

* In JavaScript, functions can be send as an argument to another function.

