【React】

Babelにコードを自動でコンパイルさせるコマンド

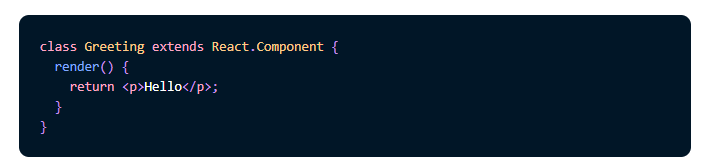
第一引数はコンパイル元、第二引数の--out-file=より後はコンパイル先。

babel src/app.js --out-file=public/scripts/app.js --presets=env,react –watch

live serverの起動。引数は起動させたいディレクトリ

live-server public

Reactについて、JSX上でクラスで定めたコンポーネントを記述する際、インスタンスを作成してからレンダリングを行う。例えば以下のようなコンポーネントの定義があったとしたとき、

以下のようにJSXにてコンポーネントの記述をしたとする。  


この時、Reactの内部では以下の処理がなされている。



<https://overreacted.io/ja/how-does-react-tell-a-class-from-a-function/>

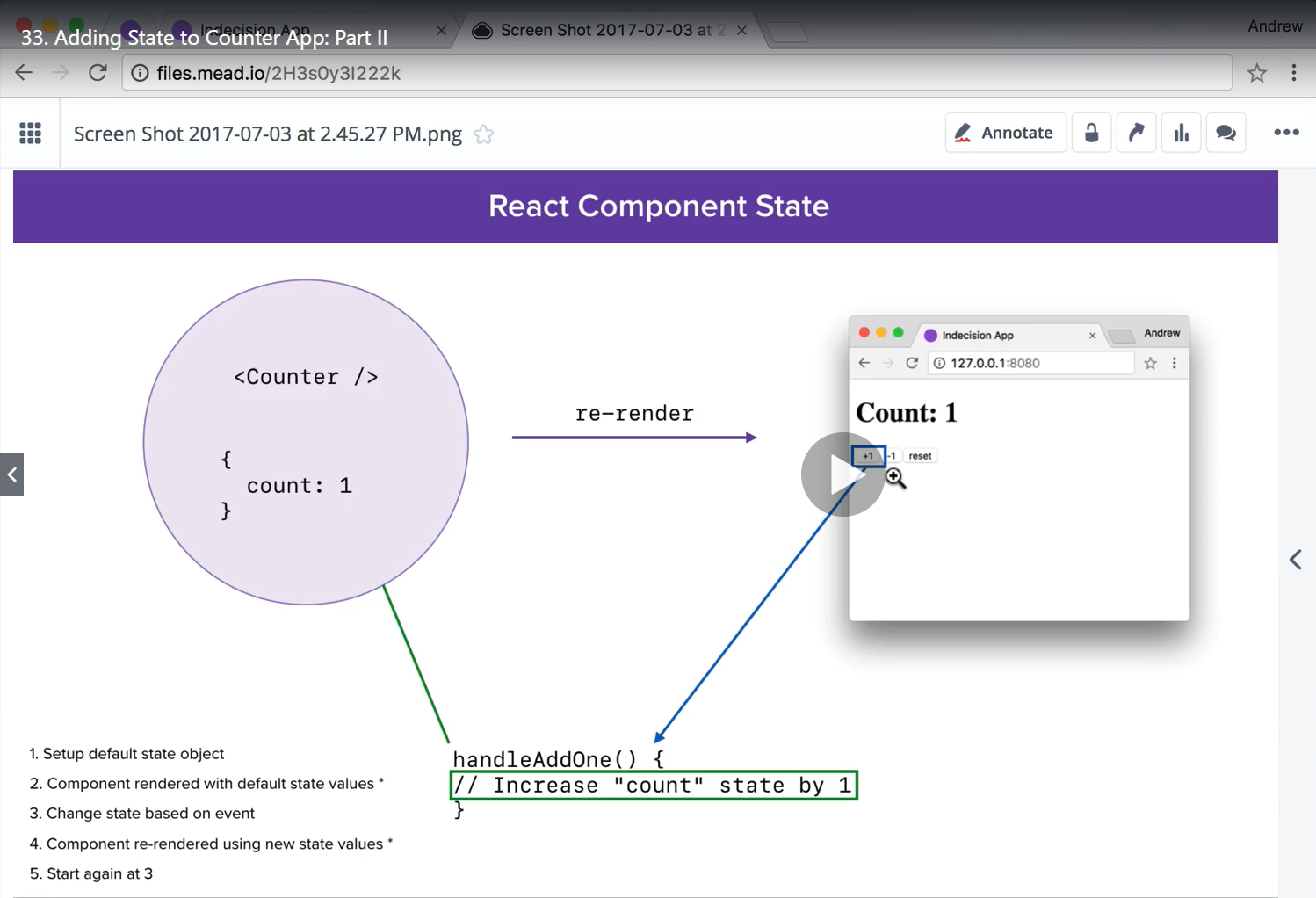
onSubmit attributeにfunctionを割り当てる場合について、e.target.elements.option.value～というように書くが、

e.targetはイベントを発生させたオブジェクト(要素自体)をさし、elementsによってその中の要素を指すよう。

<https://developer.mozilla.org/ja/docs/Web/API/Event/target>

()なしでreferenceされた関数はthisを受け継がない(thisの参照先がグローバルになる)ので、bind()によってthisの値を新たに割り当て(バインド)しなければならない。

React ComponentのStateの概念について



Stateの状態が変わるたびにレンダリングが行われるので、Stateにはアプリ上でリアルタイムで状態を表示したい情報を入れる。

ポリモーフィズム

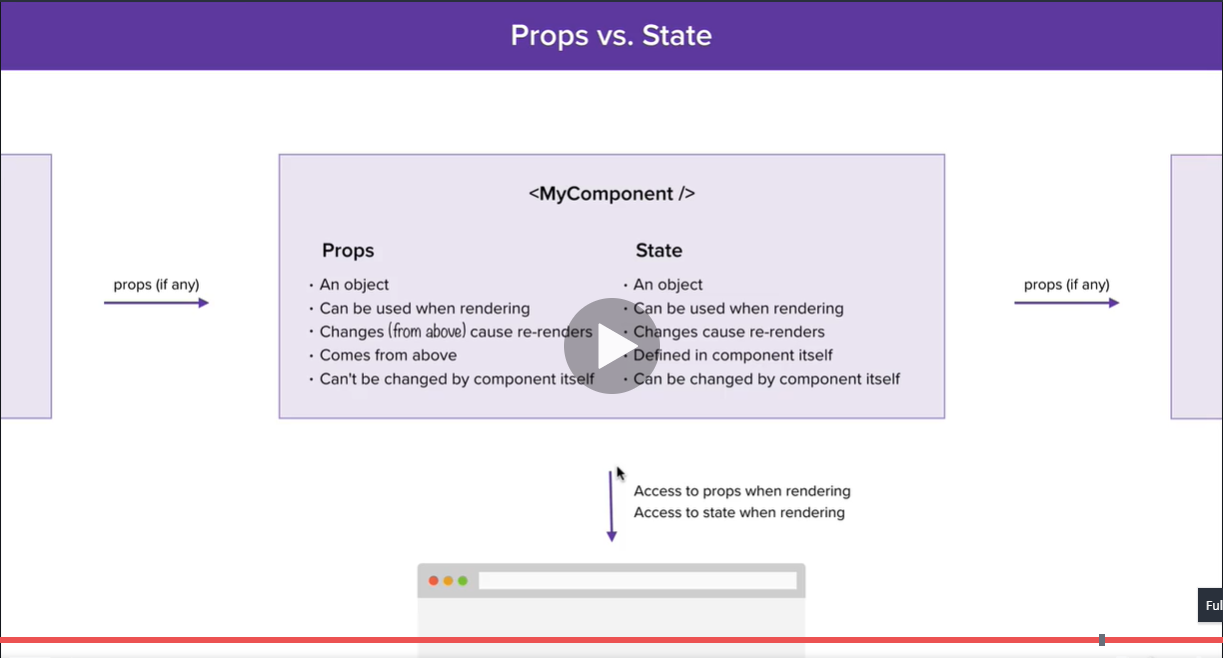
オブジェクト指向のプログラミングでは同名のメソッドを異なるクラスに定義できる。

また子クラスのインスタンスを親クラス型の変数に代入することが可能

Pushメソッドは返り値が配列のlengthと同じものになる。例えばstateのプロパティにある要素を追加した配列を設定したいときに使ったりするとプロパティには配列は入らず、配列の要素数のみが返ることになるため注意しなくてはならない。

そのような場合はconcatメソッドを使う。Concatは配列a,bに対してc = a.concat(b)と使うことでa,bを結合した配列を返すため、プロパティにも設定することが出来る。

Concatは元の配列には影響を与えず、あくまで新しい配列を返り値として返している点に注意。



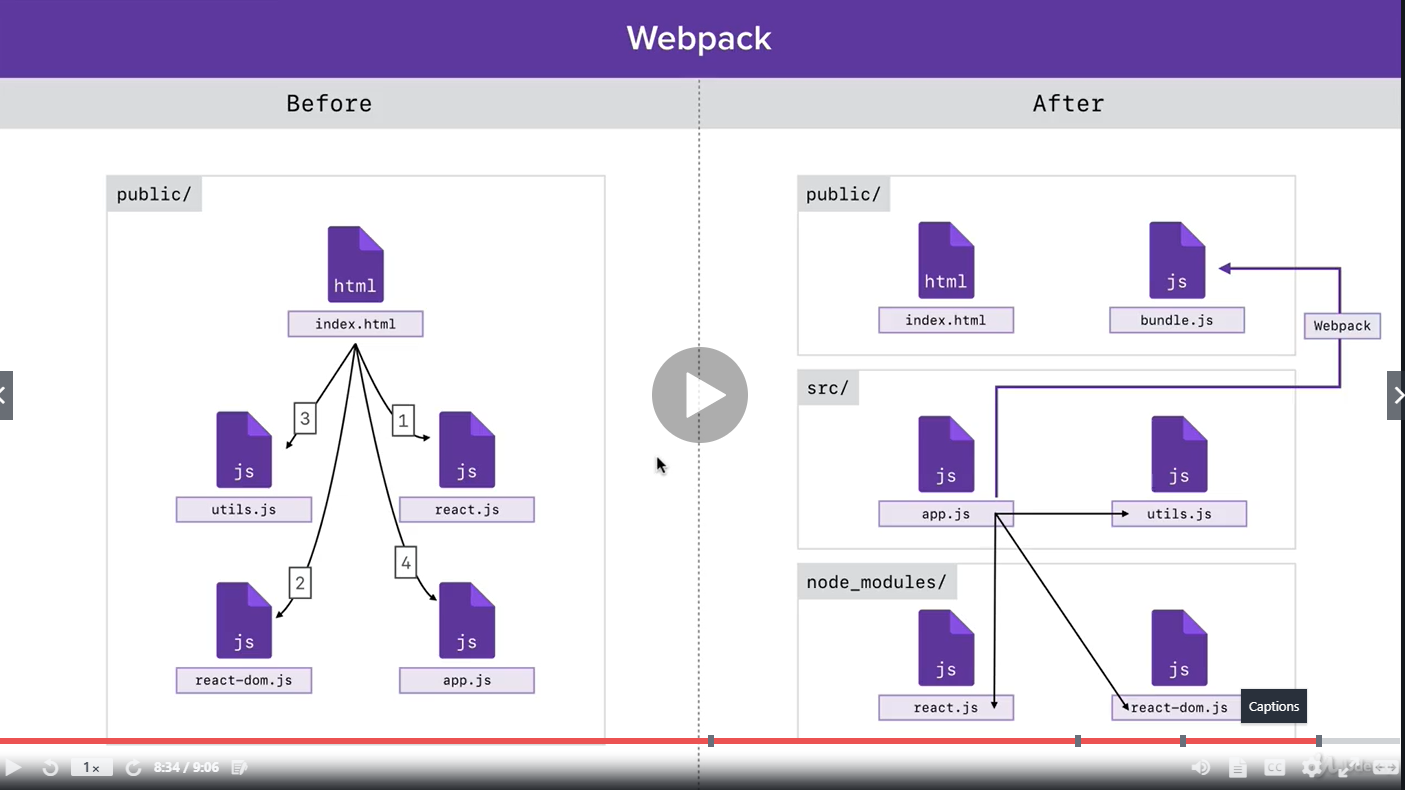
LocalStorageにstateの状態を保存する際、JSON形式にデータを置き換える必要があるが、配列やオブジェクトを保存するときはJSON.stringifyを使う。配列やオブジェクトではない場合はstringifyは使わなくて良い。

Dependencies(babelとかlive-serverといったツールのこと)はグローバルでなくローカルでインストールして使ったほうが良い。ローカルでインストールした方がバージョン名がpackage.jsonのdependenciesに記述されるし、ツールを起動するときのために短いコマンドを割り当てることが出来る(scriptsオブジェクトに記述する)。人とプロジェクトを共有する際にもツールのバージョン名などを共有できるのでローカルでインストールするとメリットが多い(おそらく職場のnuxt.jsも同じようにインストールしている)。

Npm run build(package.jsonで設定したコマンド)

でwebpackが起動し、それによってbabelによるコンパイルのwatchが始まる。

Live-serverも同じだったはず



Webpackを使うとたくさんあるモジュールをまとめたりjsファイルを単一でなくコンポーネントごとに複数ファイルに展開して使うことが出来、大規模な開発をするときに便利。cheap-module-eval-source-map

を使うとデバッグもしやすくなる。

React-modalをaddしておくとModalを実装することが出来るようになる。

その際は

import Modal from 'react-modal';

とするのを忘れずに

パッケージを探すときは<http://npmjs.com/package/(パッケージ名)>で探すとよいらしいお

Addしたnode modulesについてはpackage.jsonのdependenciesに記載してあるので、新たに環境を構築するときはそこからまとめてaddするとよい気がするけどどうやろね

========================== SCSS ==============================

Style loaderとcss-loaderを使うことでcssをDOMの中にdumpすることが出来るみたいです

Node-sassはsassのコンパイラ。これもnpm add もしくはyarn add して使う。

html{

font-size:62.5%

}

としておくことでデフォルトの1remが10pxになるので、remでのフォントサイズの変更が容易になる。Remで調整した方がページのフォントサイズを変えたりできるようなのでアクセシビリティ的にベターらしいよ、

ブラウザによってcssの差が出るのをresetするためnormalize.cssをaddし、app.js(Webpackのenter file)にimportするようにする。

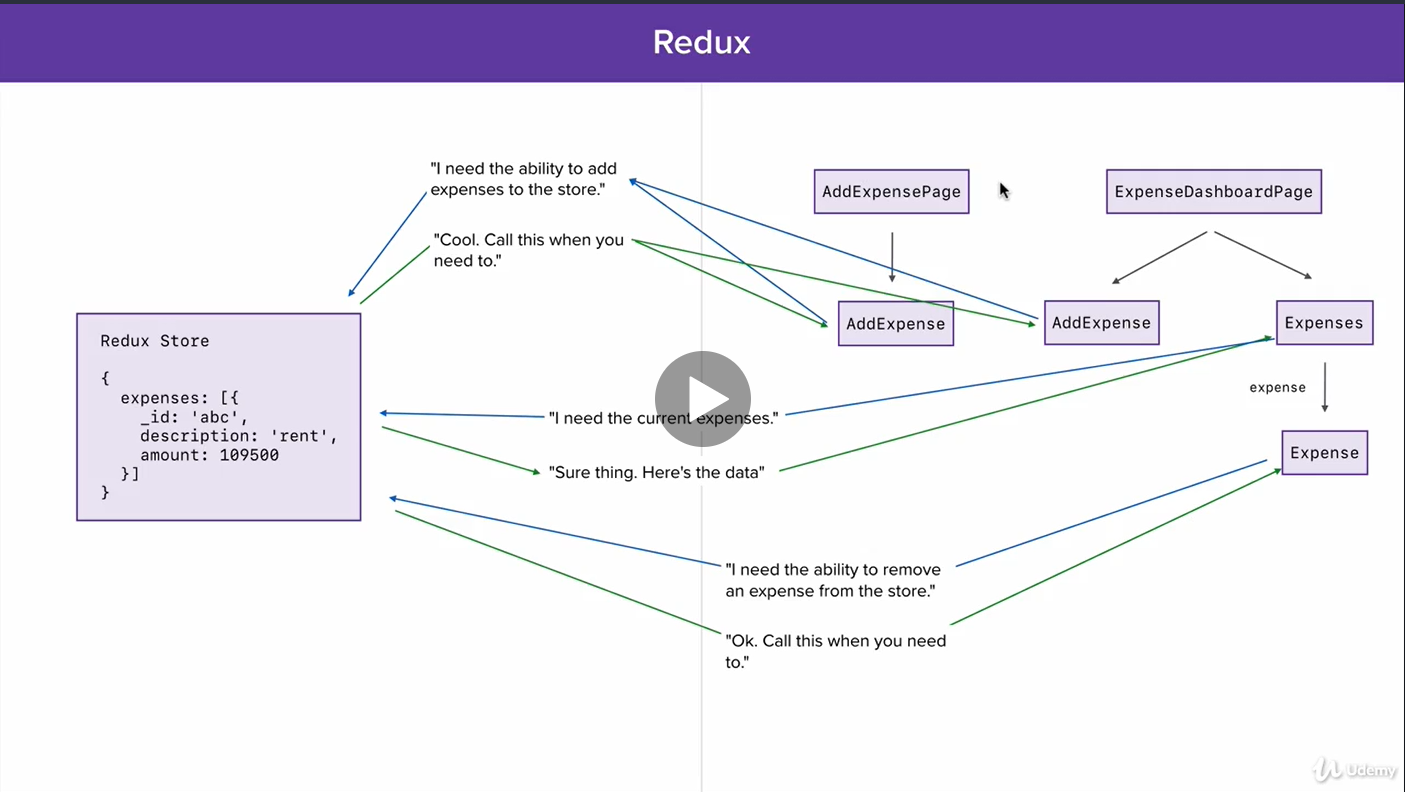
<br>ってbreakのことやったんやね！！！！！！！！！！！！！！！！！！！

React-routerの<Link>や<NavLink>はHTMLでいう<a>と同じで、クリック時に指定したURLに飛ばす機能を持つ。<NavLink>はactive(URL訪問時)の状態でのClass名をつけられたりするので便利。

<Route>はURLによって表示する内容を切り分けるタグ。たとえるなら川から水を引いてきて流れを分けるようなもの

<Switch>はif文のようなもので、現在のURLが<Route>のto属性で指定したURLで始まる(matchする)かどうかを確認していき、matchしたものを表示するタグ。なので、/だけのurlなどの場合はexact={true}を設定しておかないと表示がおかしくなる。

Redux



ES6では[…配列名]と書くことで配列名に書いた配列をコピーすることが出来る。

[配列A, …配列名, 配列B]などとすることでほかの配列と組み合わせて配列を作ることも可能。

Spread operator(上記の…)は配列の[]やオブジェクトの{}を取り払って新たに配列・オブジェクトを生成・マージする役割があると考えるとわかりやすい。

Actionはオブジェクト！(戒め)

Uuidをnpm add・importすることでIDを自動で(ランダムで)生成することが出来る。

HOC(Higher Order Component)

const withAdminWarning = (WrappedComponent) => {

  return (props) => (

    <div>

      {props.isAdmin && <p>This is private info. Please don't share!</p>}

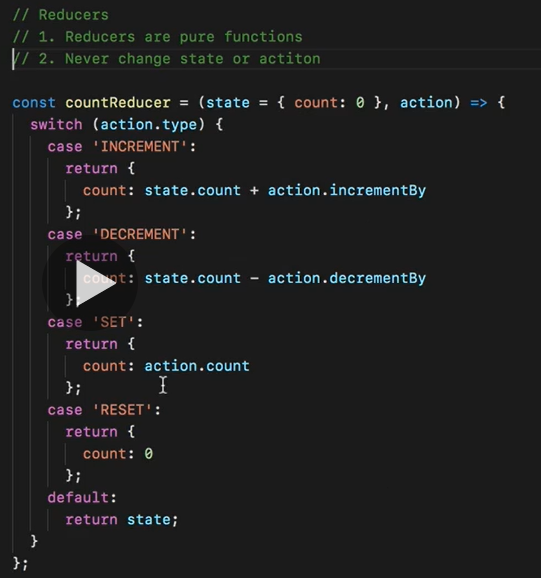
      <WrappedComponent {...props}/>

    </div>

  );

};

上の画像の(props)以下はコンポーネント。これを他の場所で定義したコンポーネントに代入することでHigher Order Component の役割を持たせる。



getStateは現在のStoreのツリー状態を取得するメソッド。

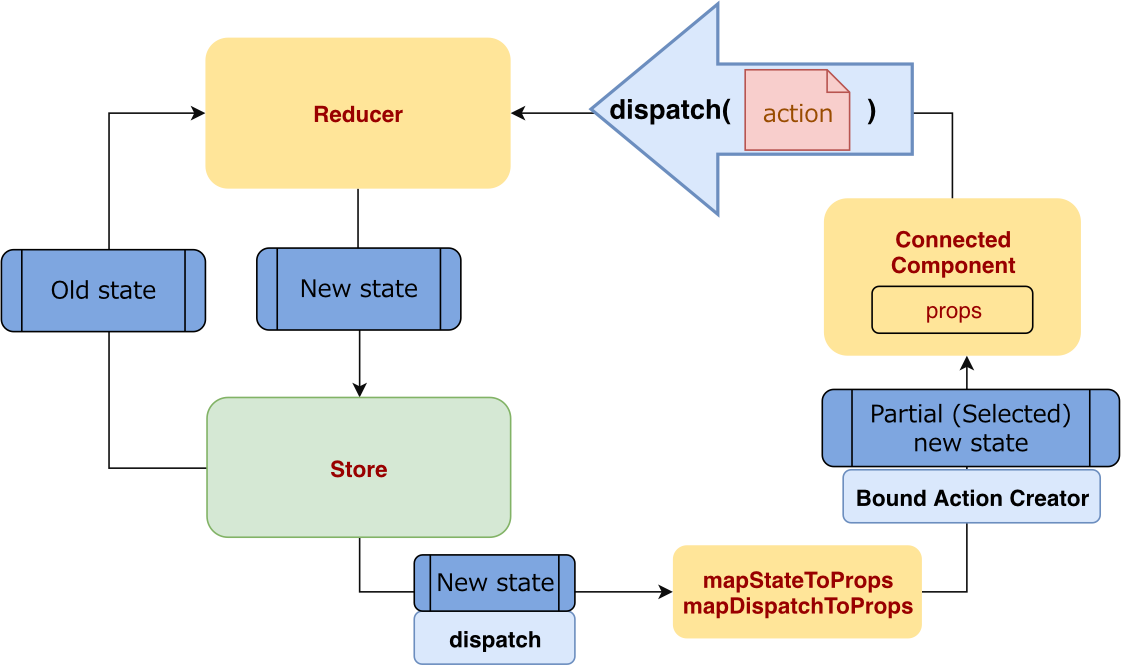
Componentを引数に持ってきて使うことが出来る(Higher Order Componentというやつ)

Reduxのdevtoolは下のソースを使う。追加するBasic Storeに書いてあるコードは随時変わるので、何か問題が起きたらサイトを見て更新してみる。

<https://github.com/zalmoxisus/redux-devtools-extension>

JEST

Test suitesの内容を記述するとき、オブジェクトや配列についてmatchしているかどうかをtestしたい場合はtoEqualを使う。BooleanやNumber、String型の場合はtoBeを使うと良い。



コンポーネントの中にJSX要素の配列がある場合、それぞれの要素にはKeyが必要(要素に追加や変更があった時に対応するKeyの要素だけ抜き出して処理を行うため)。

参考→https://ja.reactjs.org/docs/lists-and-keys.html

import 'react-dates/lib/css/\_datepicker.css'

"react-router-dom": "^4.2.2",

を使ってたけど最新版にしたらエラーでなくなった

Spyでは関数を偽装してテストを行うが、あくまで関数(関数の機能は問わない)が動作しているか、引数が正しく渡されているかを調べることが出来るだけであり、dispatchされているかどうかなど、reduxの機能が使われているかどうかなどは調べることが出来ないようだ。つまりコンポーネントの中で関数が役割を果たせているかどうかを見ている？