What are error boundaries?

<https://www.google.com/amp/s/www.digitalocean.com/community/tutorials/react-error-boundaries.amp>

What are protected routes ?

<https://dev.to/mychal/protected-routes-with-react-function-components-dh#:~:text=Protected%20routes%20allow%20us%20to,components%20along%20with%20react%2Drouter>.

What is Func Vs Class components?

<https://www.twilio.com/blog/react-choose-functional-components>

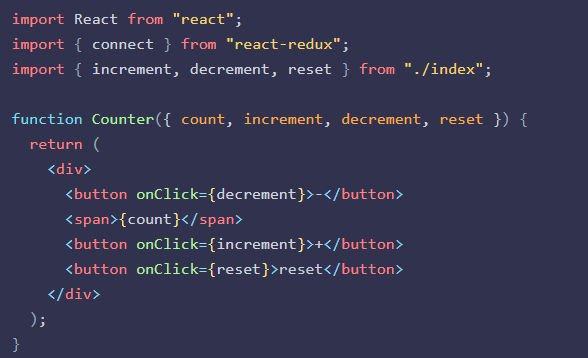
mapdispacthprops, mapstateToProps

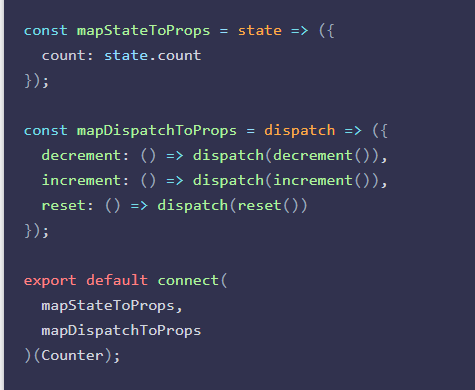
How to subscribe and unsubscribe from the store?

<https://blog.logrocket.com/react-redux-connect-when-and-how-to-use-it-f2a1edab2013/>

<https://daveceddia.com/redux-mapdispatchtoprops-object-form/>

Eg:





mapDispatchToPropss can also be a plain object



React controlled Vs uncontrolled components ?

<https://blog.bitsrc.io/react-controlled-vs-uncontrolled-components-f569f4215ac9>

This relates to stateful DOM components (form elements) and the React docs explain the difference:

* A [Controlled Component](https://facebook.github.io/react/docs/forms.html#controlled-components) is one that takes its current value through props and notifies changes through callbacks like onChange. A parent component "controls" it by handling the callback and managing its own state and passing the new values as props to the controlled component. You could also call this a "dumb component".
* A [Uncontrolled Component](https://facebook.github.io/react/docs/uncontrolled-components.html) is one that stores its own state internally, and you query the DOM using a ref to find its current value when you need it. This is a bit more like traditional HTML.

Most native React form components support both controlled and uncontrolled usage:

// Controlled: <input type="text" value={value} onChange={handleChange} /> // Uncontrolled: <input type="text" defaultValue="foo" ref={inputRef} /> // Use `inputRef.current.value` to read the current value of <input>

React Performance improvements?

<https://www.codementor.io/blog/react-optimization-5wiwjnf9hj>

1. Use Fragments
2. Don’t intialise state from props
3. Don’t use indexes as keys
4. Prefer immutability for objects and arrays, avoid reference errors
5. Prefer functional/pure components that will reduce the bundle size
6. Seperate the common modules, CommonChunksPlugin
7. Use only methods that are applicable in app , from libraries like lodash,momemt, lodash-webpack-plugin, moment-webpack-local plugin, known as dependency optimization.
8. Avoid inline functions in render, that will create new function every time, solution is to separate the component