Stateless(Presentational) and statefull(container) components

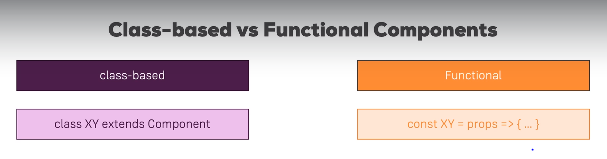
Statefull does not means class based components. It is a component that manages state.

Stateless is just a dumb component. Since 16.8 we have hooks where a functional component can also use states using useState().

It is better to have majority of stateless components than statefull components.

Why? By splitting up app into container components which manage the state in the app and the dumb presentational components, we have predict the flow of the data where the state changes in the couple of components and the other componetns just used for nice UI. They only define on props. So they can be thrown anywhere in our application, just we need to pass the right props for that component.

**Class based vs functional based.**

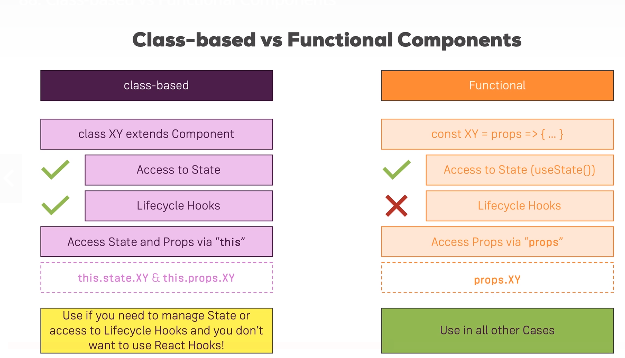


Class based are those where we extend the component class from the react library.

Functional based are those where we get the props for the parent component and deal with the presentational logic.

Class based consists of state where we can manage the state data using setState() where as in functional based we cannot manage the state data. But that is a thing of past. Because we have hooks now from 16.8 version where we can manage state in the functional components also using useState().

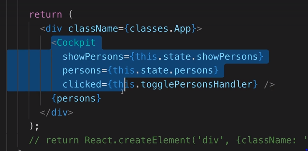
So now wahat is the difference?



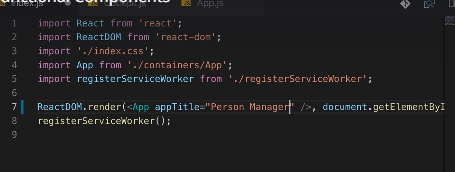
In class based we use this.state and this.props. Here we use “this” because state and props are properties of the component class.

In functional components we get the props as a functional argument so we can use that without using “this”.

Thus far we have done something like this.

 sending state data to the cockpit component as props. Here our app.js file which is a statefull component doewsnot receive any props. But we can change that.

In our index.js file where we call our App component in the render function we can send props like.



So if we want to use that prop appTitle in our app.js file we write something like this.



Using {this.props.appTitle} in the rendering of cockpit component tag.

In the cockipit component file we can access it like this below. Without the “this” operator because it is not a class based.it is a functional based and we get the props as an argument.

