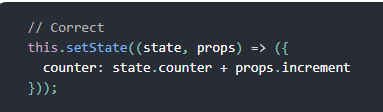
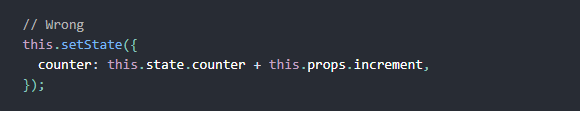
# Rest syntax and spread syntax:

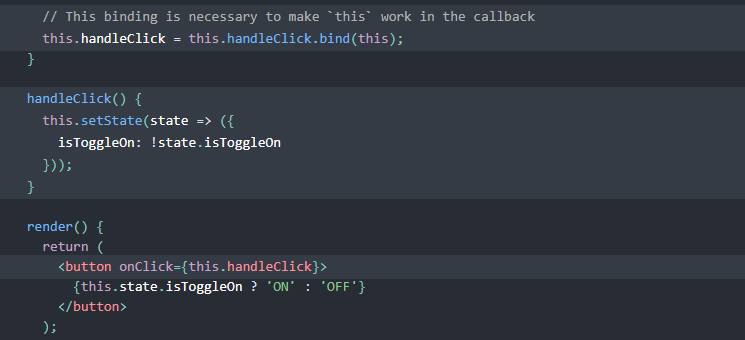
1. Stateless Component: component cant use lifecycle
2. Props is input of component: can’t change.
3. State: variables used inside component to change output.
4. componentDidMount: get external data in this function.
5. Using setState to modify state.
6. State updates maybe asynchronous so don’t use this.state.abc when setState. Instead using a function :



1. Again state updates are asynchronous so if u need code to run after the state is updated. Using callback function

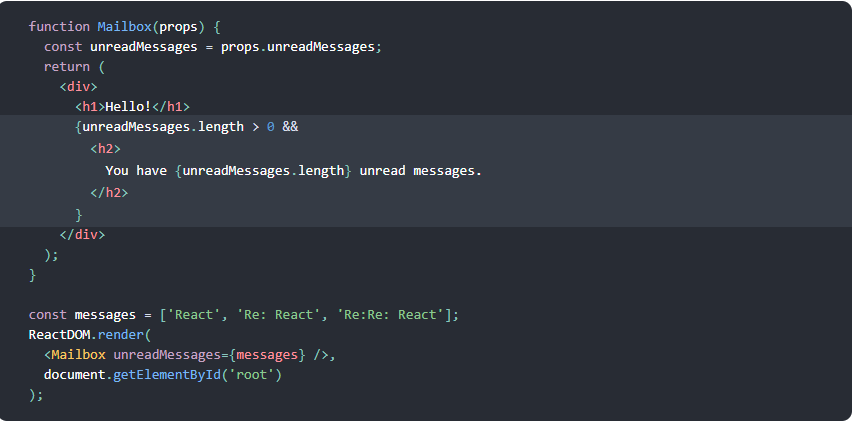


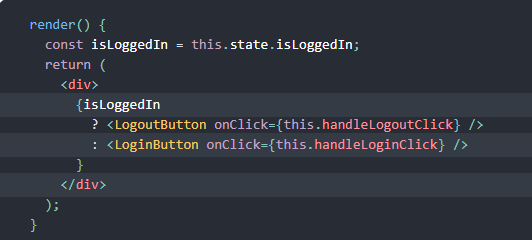
1. Don’t forget binding=> “this” will be undefined when the function is actually called

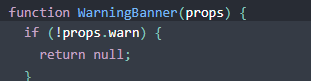


1. Conditional Rendering:

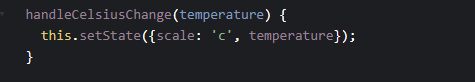
1: Wrap it in curly braces like this

=> this means if expression true => display the element right after “&&”

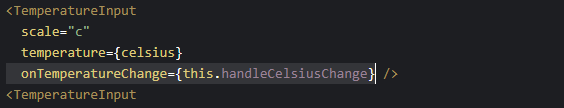
2: 

If u want to prevent render component => instead of render output, render null 

1. Using props to send data from parent to children component
2. When use want to pass data from child to parent   
    + Parent: create function with parameter (which is value send from child)

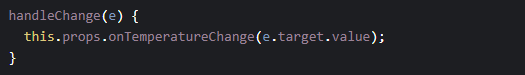


+ pass function as pros to child:

’

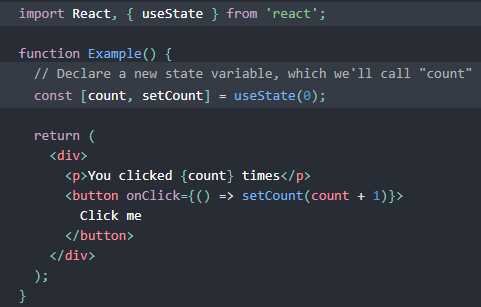
1. Child component:

+ In handle event (which want to send data to parent): call parent’s function:



* This is also a way to communicate between siblings components

1. HOOKS: They let me use state and other React features without writing a class.



1. Life cycle:

* ComponentDidMount: fetch data from api
* ComponentDidUpdate: change state when props change

1. Use react-router:

* BrowserRouter(one BrowserRoute accpects only one component so wrap </App> in BrowserComponent) -> Switch (group all components together)-> <Route exact path='/' component={Login} />: In route we have 3 props: path, component(render: used when have condition)
* Get parameter from url: /user/:username => this.props.match.params. username