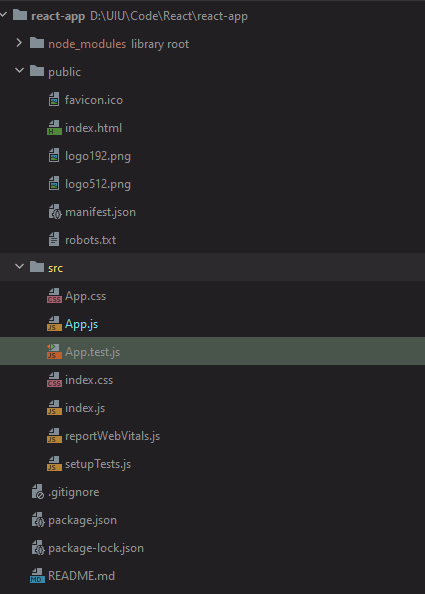
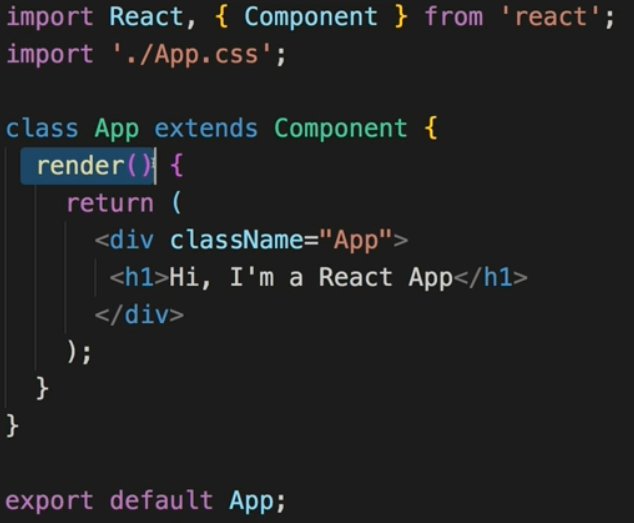
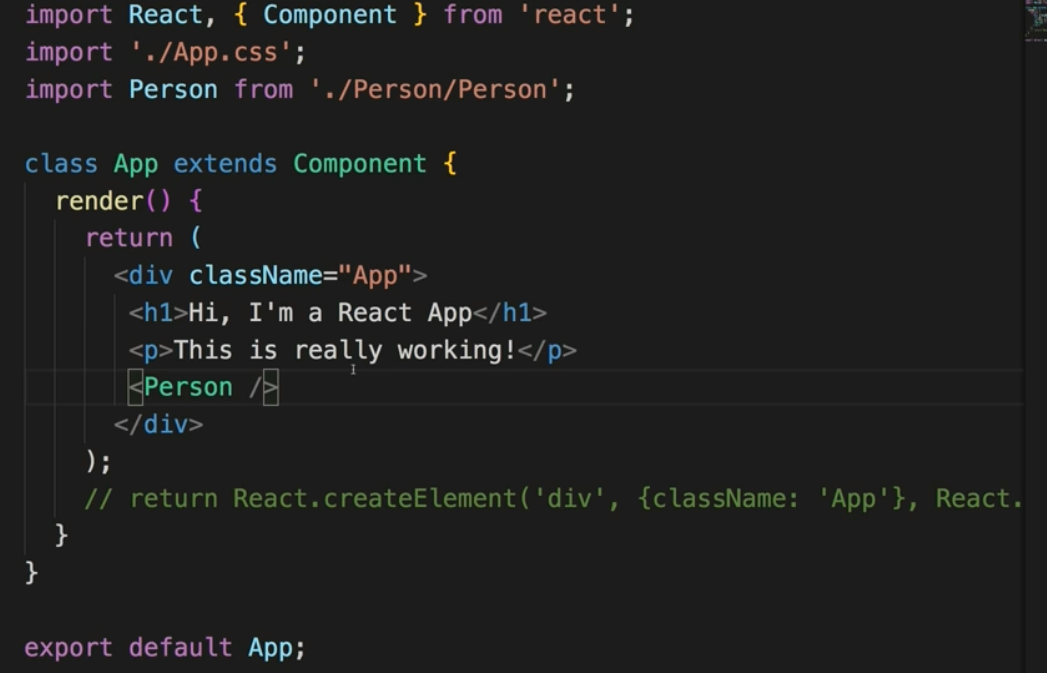
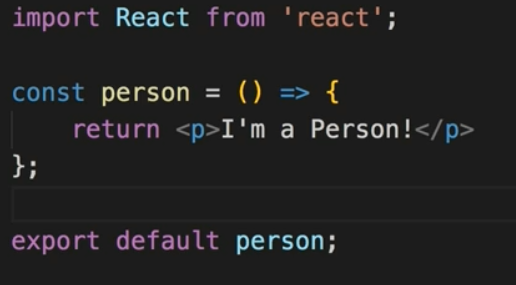
* **Creating a new react app**:
  + npm install create-react-app -g
  + create-react-app app-name-here –scripts-version 4.0.3
* **React App folder structure**



* + **.lock** files can be ignored
  + **Package.json** have the general dependency and scripts
  + **Public** folder has only one index.html file. That will be only file it has. We won’t add anymore page there as it is a single page application.
  + **Manifest.json** file have some metadata about our application.
* **Component** can be defined in two ways
  + By extending the Component class, important is that it has a render method with some JSX syntax (almost same as HTML with some difference).



* + The **React.createElement(types, [props], […children])** takes three arguments (at least). We can nest more createElement inside the children to make react app. But that is lot of hassle, in this case JSX helps with the syntax conversion. It has some limitations as well. For example, we can’t use class inside JSX cause it’s a reserved keyword in JavaScript. Instead, we use className. Another restriction is it can have only on root element.
  + Second way is making functional component and exporting it, this is more common. One Note is, we should make the component start with **capital letter** as convention (I found out hard way by not being able to show up the element).



* + We can add dynamic content with curly braces inside JSX code. We can also pass attributes inside an element with **props**. Props is an object with all the property of the component. We also have a **child props (**example: whatever we write inside the <p> <p/>**)** as well which can be accessed as props.children. The react props also can send methods. 
* **State** property we can use to change element from inside the component. State is only available to components that extends the component (in other words class-based component). Even though now we can use state in functional component as well using react hooks.
* **Event handling** for buttons can be done with onClick property. This is usually a function. Inside the function we can change state using the **setState()**method. It takes an object as argument. When this function execute it will look for which part of the state is mutated and it will update that part only. It will not change other states.

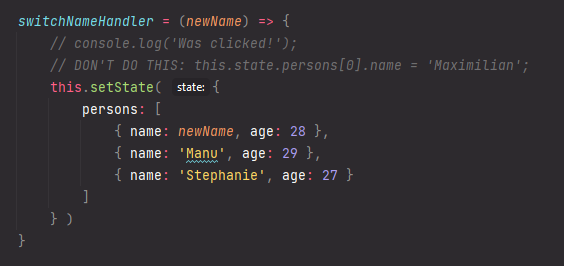
[**note**: this is asynchronous function, so we cant really get output immediately on the next line of code. If such case arises, we can use the 2nd parameter of the setstate method. This is a callback function which will run after the setstate asynchronous function is done]

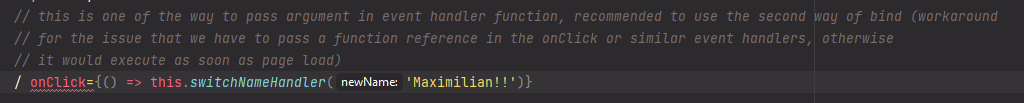
* + **For functional component** react has hook as useState() to setup state. As argument we pass the initial state. This returns an array with exactly two elements; first element is the current element and second element is a function that lets us update the state. Here we can use multiple useState.
* **Two-way binding**: For example, let’s say a component person output a paragraph with name and has an input field. The paragraph name is displayed with {props.name} which comes from the state of the component that called this person component.

Now we can pass an onChanged event listener function for the input field, which in turn will take the value of the input field via event.target.value. This function then can update the state with the name that is written in the paragraph.

So this makes it two way binding, the name is coming from the parent component, but its updating via the input field of child component.

* Two ways of passing arguments in an event listener function:
  1. This one isn’t recommended.





* 1. The bind is the recommended way.

