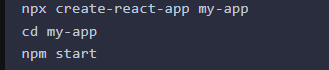
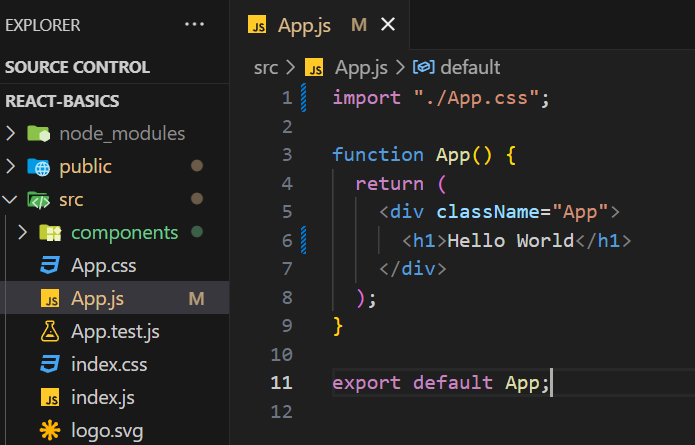
**REACT**

* ReactJS uses 3rd party libraries for routing and making http requests in react application. Axios http library used to make a rest API call
* Using create-react-app to setup react application -



* App.js contains the root component that will be rendered to the DOM

O/P – Hello world on http://localhost:3000/

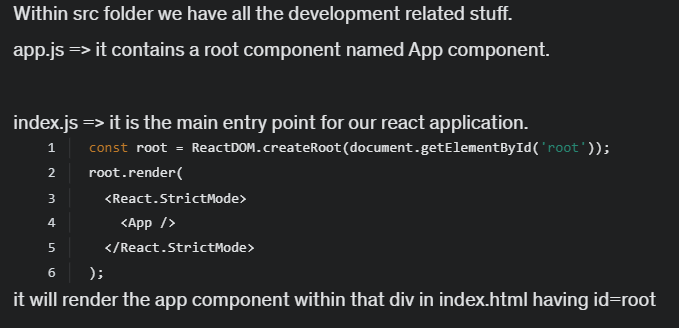
* DOM stands for Document Object Model it represents a tree structure of ur UI application. Whenever u load ur application in broswer, then browser will create a DOM for ur UI application.
* The React Virtual DOM can be thought of as a lightweight copy of the actual DOM. Whenever the state or props change, before making updates to the real DOM, React first updates the Virtual DOM, calculating the differences between the previous and current states—a process called diffing.

After identifying the differences, React only applies the necessary updates to the real DOM. This makes the rendering process more efficient since, instead of re-rendering the entire page, only the parts of the DOM that require changes are updated.

* **package.json**file contains all the dependencies and scripts required for out react application. Whenever we run npm install command, then npm will refer the package.json's dependency section and install these the dependencies in node\_modules folder.
* In single page application there only 1 html file.

Inside **public folder** we have **index.html** whose body should contain a single div and this div is responsible to render the react component. 

We may change the head section of html file to include css and script files but we don’t change body section of index.html

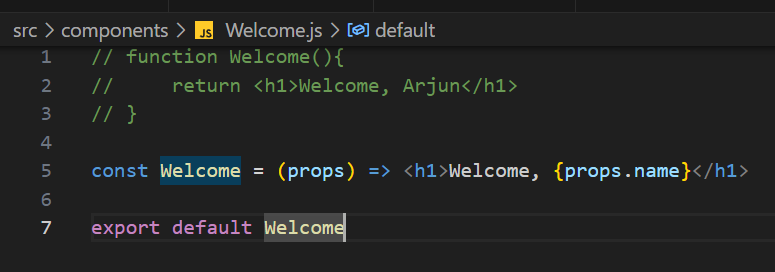


* App.css and index.css contain styling related stuffs
* **Flow of code -**Whenever we access the react application from the browser then index.html will get served in the browser. Next, the index.js file will load into the browser which will render the App component.
* App component contains JSX code which is returning some html. JSX is a syntax extension for JavaScript that allows users to write HTML-like markup in JavaScript files.

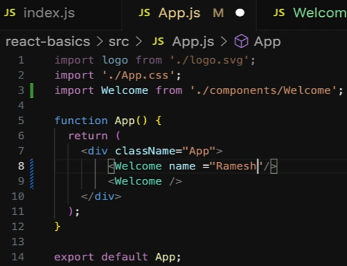
**Functional Component**

If u want to pass the data dynamically from one component to another component, then we can use props. Component file can have .js / jsx as extension.

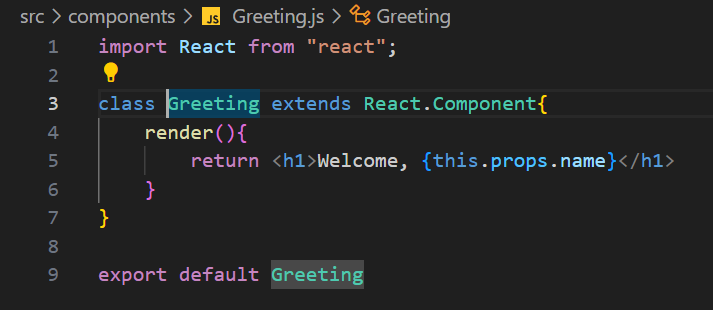
In order to use one functional component in other components, we have to export it.

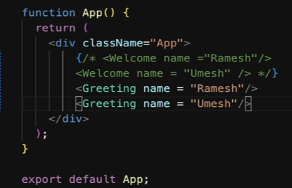


We have App component as parent and Welcome component as child component. Lets say we want to transfer the data from App component to Welcome component, then we can use **props** (properties) and we can pass the properties from parent component to child component.

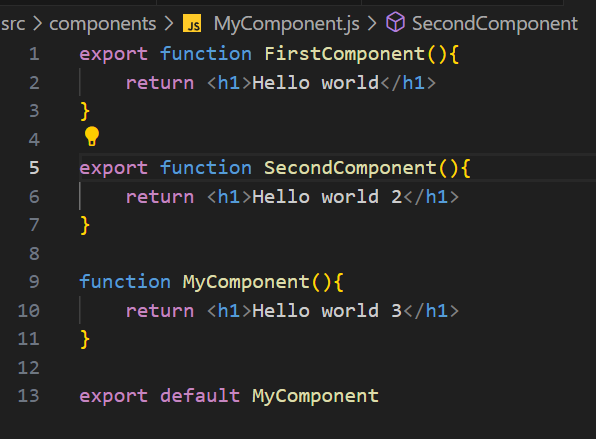


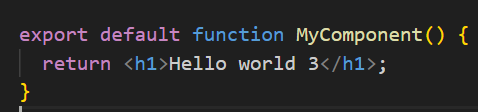
**Class component**  - props are automatically available in class component, we don’t have to pass it to render method.

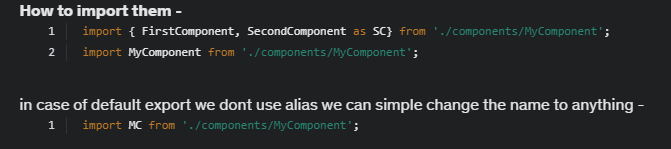




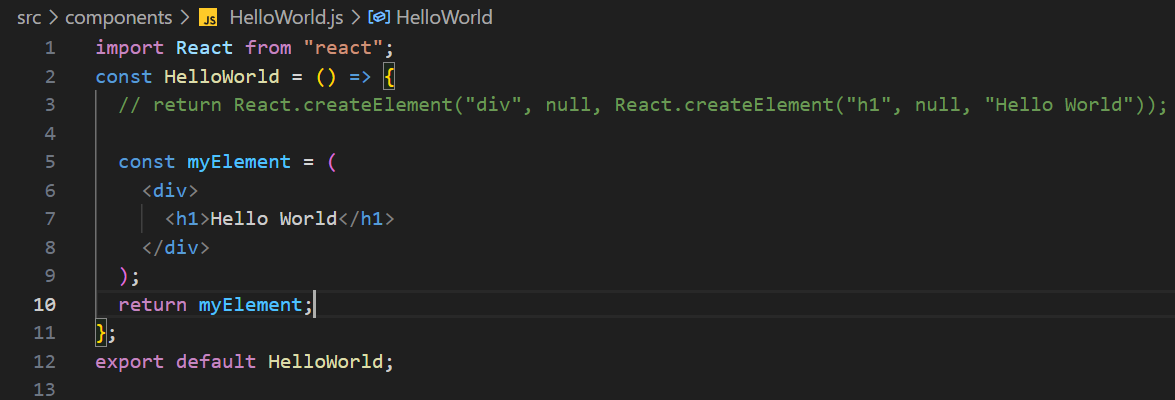
**Importing and Exporting Components**



or like this -

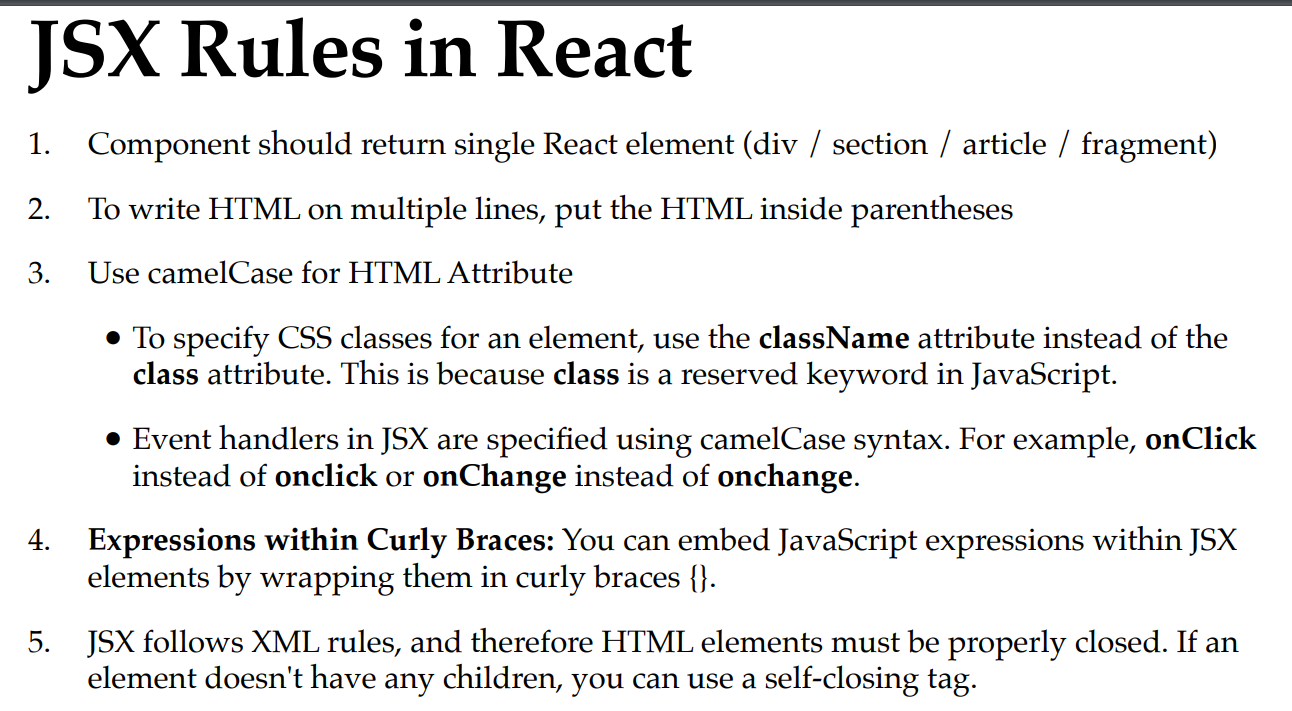


**JSX** – much easier to create element using it rather than using React.createElement() method

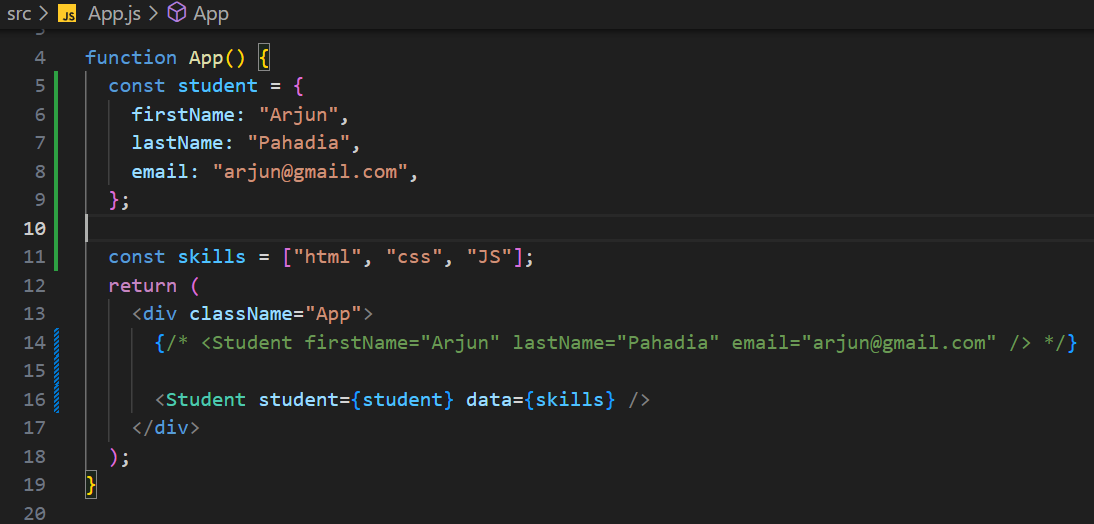


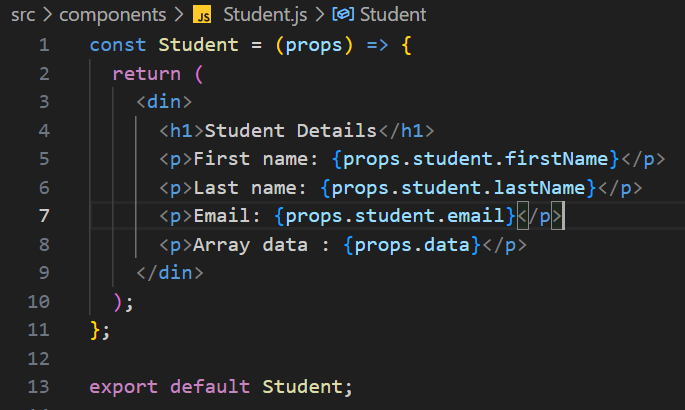
* Whenever we have multiple html elements in jsx, we have to wrap them inside an parent element like <div></div> or empty tag <> </> called react fragment.
* Use () to write html in multiple lines.



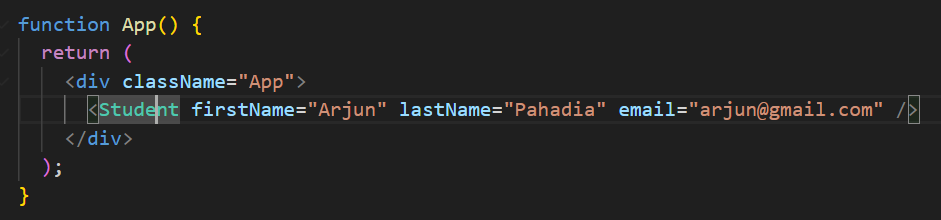


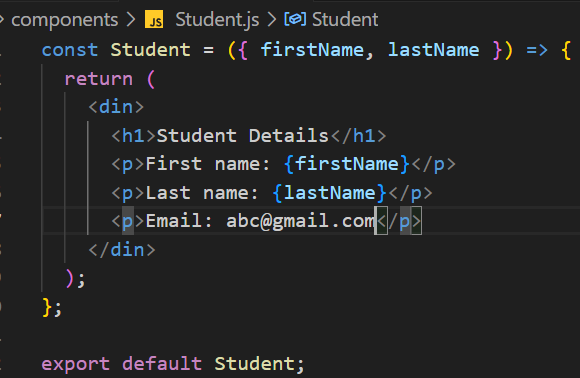
**Using** **props –**





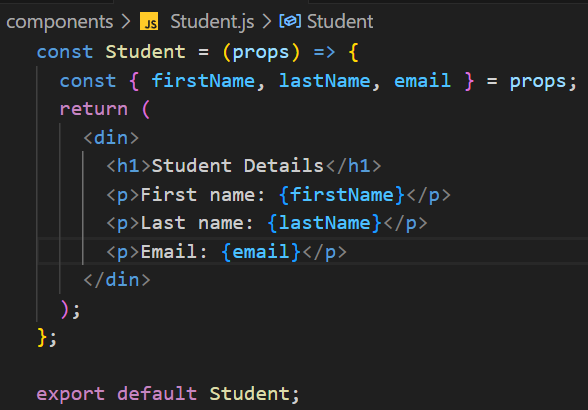
**Destructuring Props –** extracting data from array or object and assign them to their own variable





Or

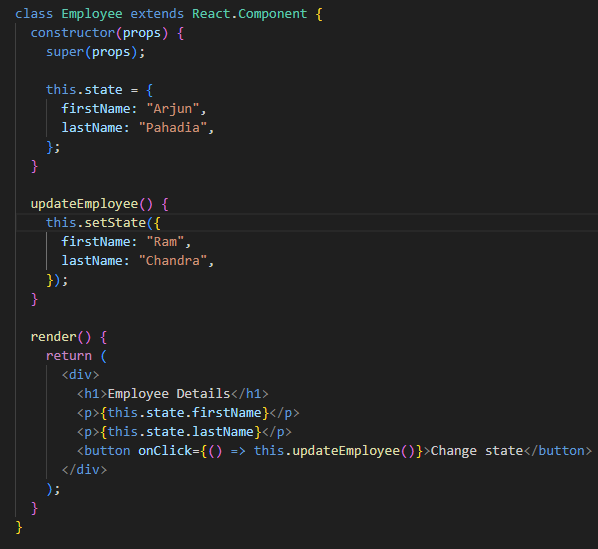
Destructuring in function body -



**State and setState in Class Components**

In class components, use arrow function to call event handler

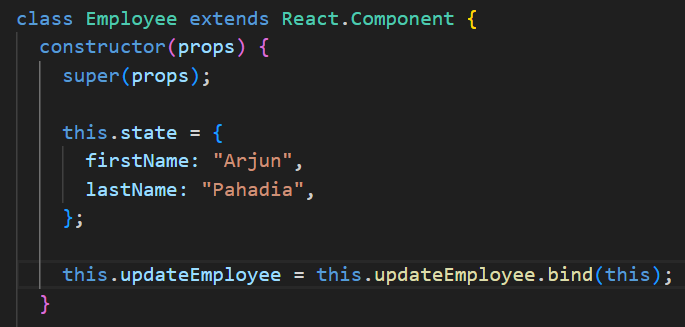
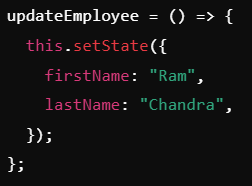
In constructor, we specify the state object. Within state object we can define the properties and values -



the arrow function () => this.updateEmployee() will **bind the correct context** (this) to updateEmployee, ensuring that the this inside updateEmployee refers to the Employee class instance. This is because arrow functions **lexically bind** this, meaning they inherit the this value from their surrounding context (the class instance).

In JS, the this context inside a normal function depends on how the function is called. When updateEmployee is passed as a callback (onClick={this.updateEmployee}), it is no longer called as a method of the Employee class instance. Instead, it is called in a different context, and the this inside updateEmployee will be undefined (in strict mode) or refer to the global object (in non-strict mode).

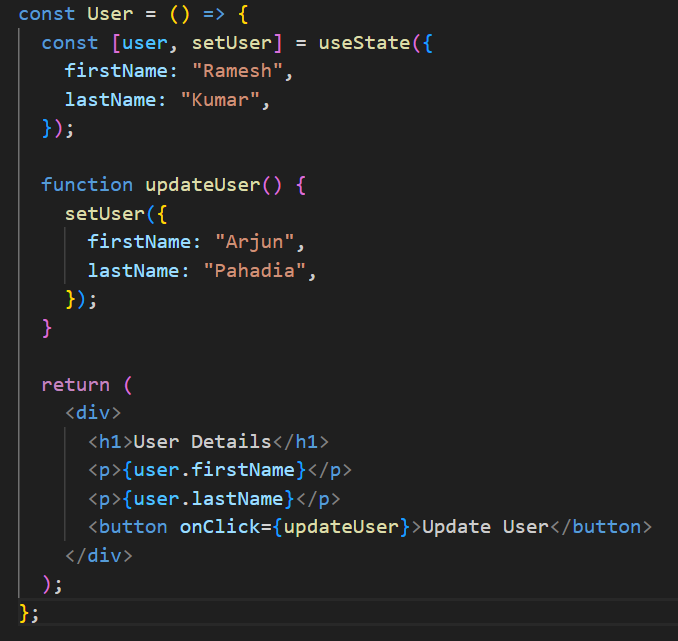
If you want pass updateEmployee directly, you need to bind it to the correct this context in the constructor or u write updateEmployee function as arrow function

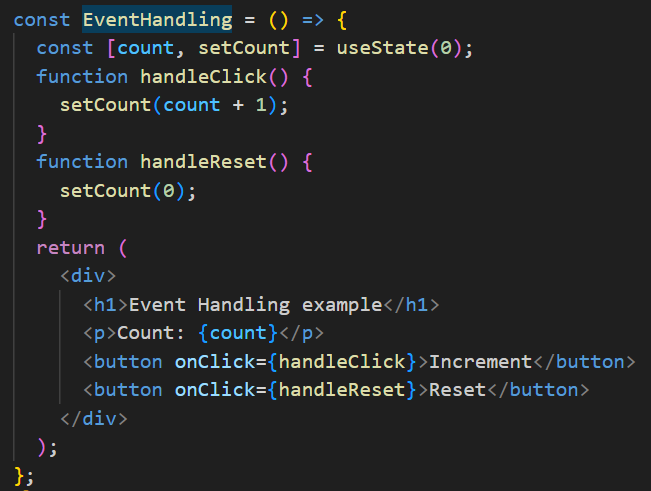
**useState Hook**

It allows us to have state variables in funcitonal components. useState hook is itself a functional component in “react” library imported using named import. It takes one argument as initial state and returns an array which contain 2 values - current state and function that lets you update it.





**Event Handling**



**Conditional Rendering**

Using Short circuit operator

{ condition && <Component /> }

* If condition is **truthy**, <Component /> will be rendered.
* If condition is **falsy**, **nothing will be rendered** (the second part after && is skipped).

