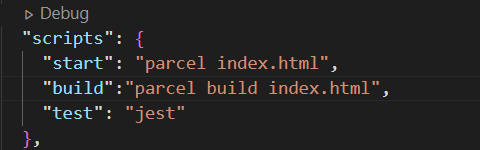
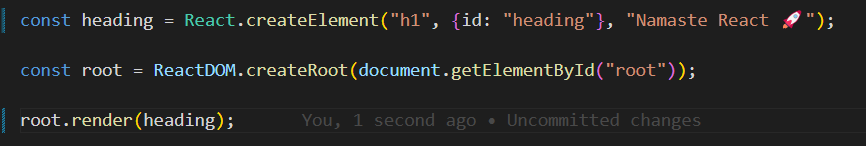
1. As soon as we do npx parcel index.html:
   * It’ll host our development build to localhost:1234.
   * It means we’re executing npm package parcel and we’re giving source file as index.html.
2. If we don’t want to run npx parcel index.html to build our application, then we can also create scripts under package.json.



* + For running the project => npm run start
    - We can also use **npm start** instead of *npm run start* but we can’t do it for building the project. It seems ‘start’ keyword is reserved by npm for starting the project.
  + For building the project => npm run build

1. **React.createElement** => gives you a ReactElement and that ReactElement is an JS object.

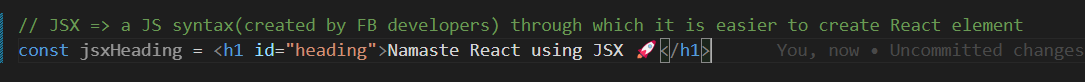
Now, when we render this JS object onto DOM, it gets rendered as an HTML element.

1. **Core react** will create the element but for displaying onto browser we’ll do **ReactDOM.createRoot**. 

When we do **root.render** => it will take object and convert that into HTML element and push that into browser.

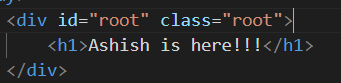
* Push it means => it will **replace** everything from root(index.html) that is inside that root. **It will replace not append.**

1. **JSX is not HTML in JS. It is a HTML like syntax. And JSX is note pure JS also.**

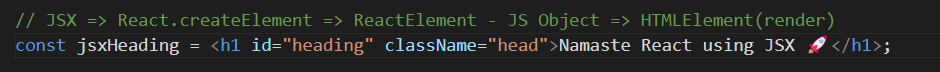
****

* Our browser won’t understand JSX. It only understands ECMAScript or ES6 (that is pure JS).
* **If browser does not understand JSX, how it is getting printed to browser?**
  + Even before this code goes to JS engine, it gets transpiled. And then JS engine receives the code that browser can understand. **This transpiling is done by PARCEL.** And this transpilation is not done by PARCEL directly rather it gives this responsibility to a package known as ***BABEL***.
  + **BABEL is a JS compiler (transpiler).**
* **How JSX code is working?**
  + JSX code is transpiled to React.createElement and this React.createElement, which is a JS object when rendered onto DOM, it gets rendered as HTML element.

1. There are some older browsers that do not understand ES6 JS code. So, BABEL helps in transpilation of the code and make those browsers understand the code.
2. **Why JSX in not HTML?**
   * When we’ve to give class name in HTML, we do the below step:

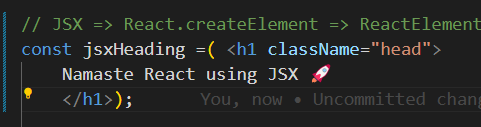


* + But when we’ve to give class name in JSX, we do the below step:

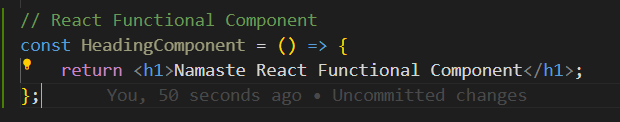


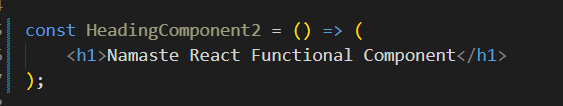
* **If you’ve to give attribute to JSX, you’ve to use camelCase.**

1. When you’re writing JSX in a single line then is perfectly fine but when you’re writing JSX in multiple line, then you’ve to wrap JSX in parentheses (). This is because **Babel needs to understand from where JSX is starting and where it is ending**.



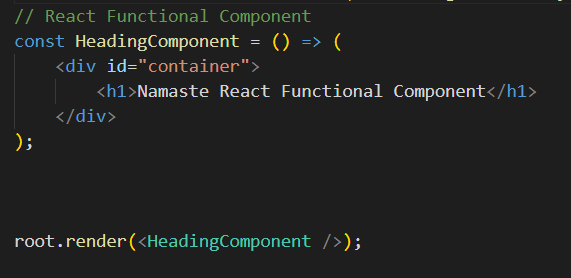
1. **2 ways of creating REACT COMPONENT:**
   * Class Based Component : based on class and it is an old way to create components.
   * Functional Component : based on JS functions and it is new way to create components.
     + Functional Component is normal JS function that returns some JSX (ReactElement).
     + It can be written in 2 different ways and both will be same.



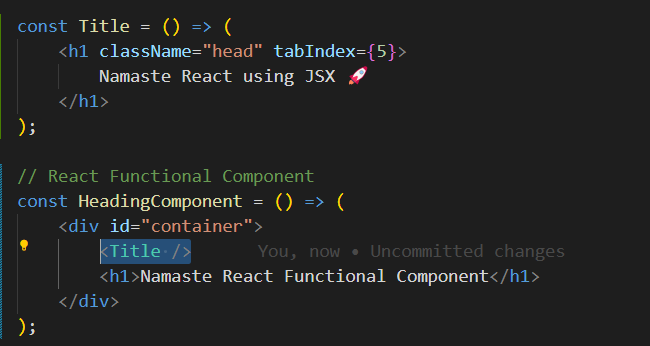


* **If you’re creating functional component or any react component, name it with capital letter or you’ll get error.**

1. **How to render React component?**
   * We’ll do => **root.render(<HeadingComponent />);**

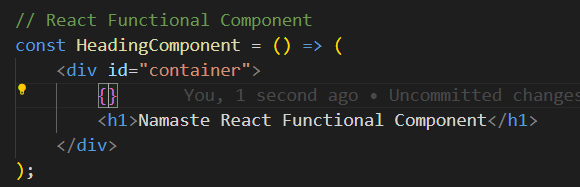


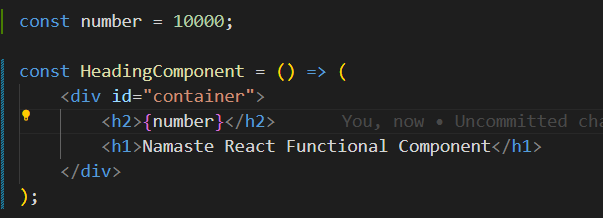
1. **Suppose we’ve more than 1 component and we want to render one component inside another component, how would we do that?**
   * We’ve named one of the react component as Title. Now, we’ll put <Title /> inside another component below <div id=”container”>



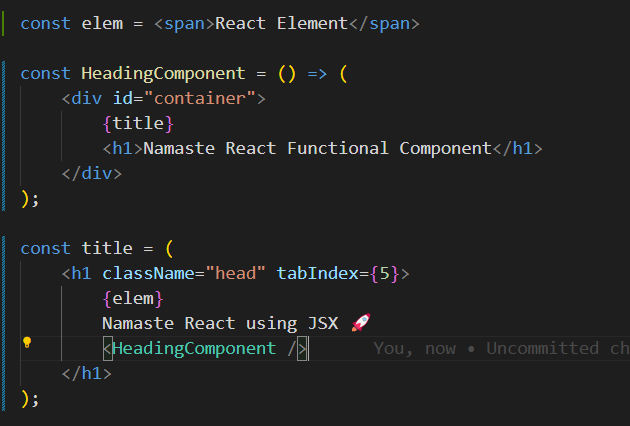
This is called **Component Composition**.

1. If you write any curly braces {} inside JSX => then you write any piece of JS code inside that braces.

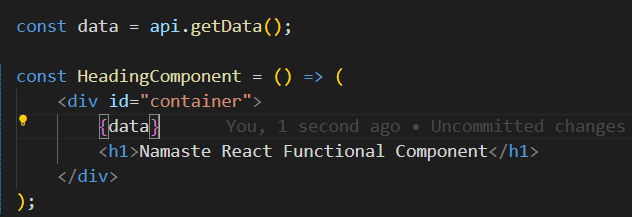




* **You can write any piece of JS code inside JSX using {}.**

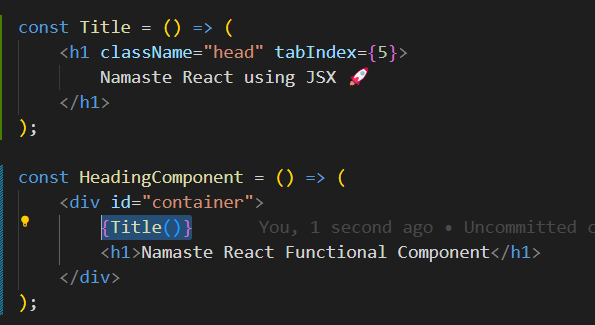
1. ****

This will go in infinite loop. Because title is inside HeadingComponent and HeadingComponent is inside title.

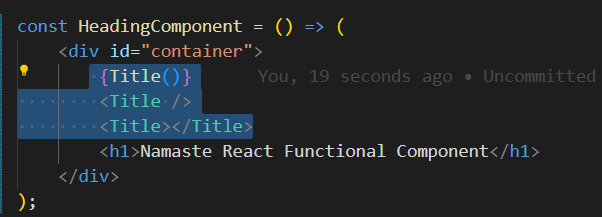
1. 

Suppose this API is a malicious one and an attacker tries to steal data/cookies by writing some piece of JS code => **So JSX is taking care of these injection attacks. Even if API is passing some malicious data to your code, JSX will escape it. It sanitizes everything.**

* ***When an attacker is trying to steal data/cookie by sending malicious data through API call => then it is called Cross-site scripting.***

1. ******

*There’s one more way to render one component inside another component. You can call 1st component inside another because at the end Functional component is JS function =>* ***{Title()}***

1. **

There are 3 different ways to render one component to another.

1. {Title()} => call it as a function
2. <Title />
3. <Title></Title>