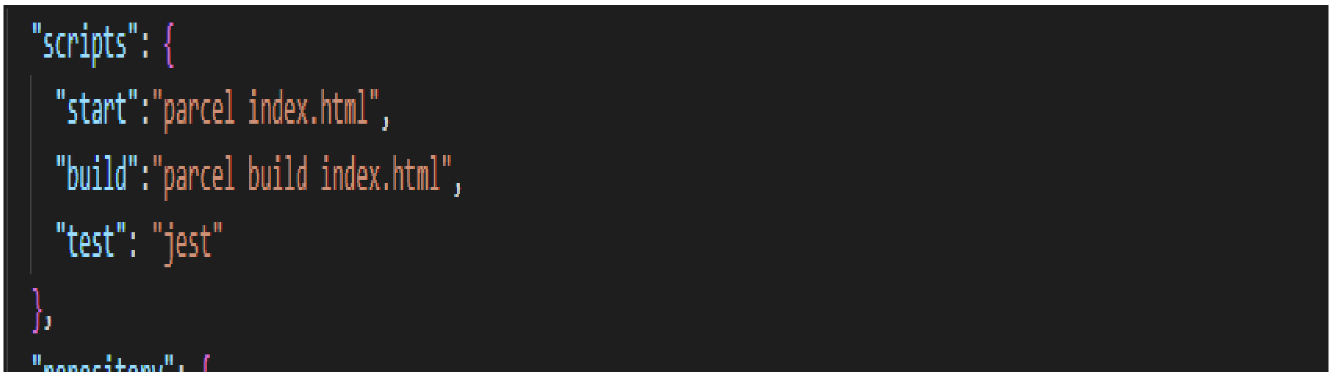
Chapter 03 - Laying the Foundation



```
"scripts": {
  "start":"parcel index.html",
  "build":"parcel build index.html",
  "test": "jest"
},
```

When we have created these scripts we no longer need npx parcel index.html to run the application.
It is an industry standard to use these scripts to run, build the application. So if you have any doubt how to run or build the application go to the package.json file . See the scripts and under scripts there are commands mentioned for run , build etc.
If we want to run command npx parcel index.html, we can type npm run start. It will do the same thing
If I had to build my app, I would have to write npm run build
In place of npm run start, we can do npm start but this does not work for build.
If i am typing npm start, it is behind the scenes calling npm run start which is eventually calling npx parcel index.html

React Element:

These React elements are objects
Const heading =React.createElement(“h1”,{id:”heading”},”This is my heading tag”);

React Element at the end of day is object. When we render this object to dom ,it becomes an html Element.

When we are doing const root= ReactDom.createRoot(“document.getElementById(“root”));
an element goes into react dom
And when we are doing root.render(heading), the heading will get converted from object to react element and will render into DOM and override the already existing dom.
It is not a good practice to do React.createElement everytime.
To solve this problem, Facebook created jsx.
JSX is like html like but it is bit different from html.
Creating an element using JSX and creating it using React.createElement is same as both makes an object which gets converted into element when we render it inside DOM.

But the problem is our JS engine cannot understand jsx. So we need a compiler named babel which converts jsx code to javascript code which can be understood by the compiler

What is JSX?

JSX stands for JavaScript XML. JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods. JSX makes it easier to write and add HTML in React. JSX converts HTML tags into react elements.
In Case of jsx elements we use camel cases for the attributes
For example for normal html tags we use class but for jsx tags we use className
So at the end of the day babel transpiler it and converts jsx to React.createElement()

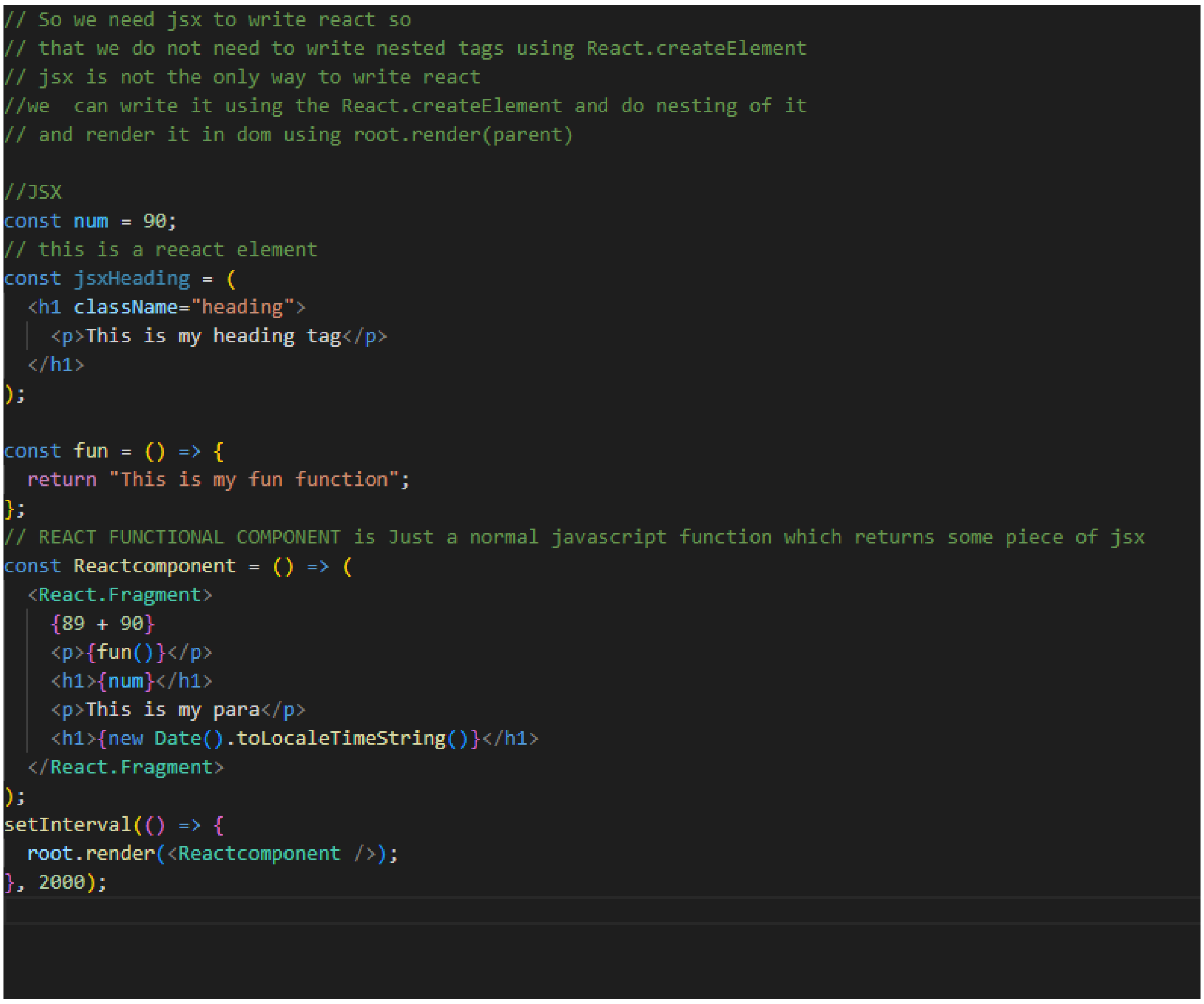
React Components

Class Based components- old way
Functional based components - new way

Functional based component:
Just a normal javascript function which returns some piece of jsx

```
const jsxHeading = (
  <h1 className="heading">
    <p>This is my heading tag</p>
  </h1>
);
```

This jsxHeading is a jsx element or a react element
And functional component is a javascript function that returns jsx or we can say it returns a React element



```
// So we need jsx to write react so
// that we do not need to write nested tags using React.createElement
// jsx is not the only way to write react
//we  can write it using the React.createElement and do nesting of it
// and render it in dom using root.render(parent)

//JSX
const num = 90;
// this is a reeact element
const jsxHeading = (
  <h1 className="heading">
    <p>This is my heading tag</p>
  </h1>
);

const fun = () => {
  return "This is my fun function";
};
// REACT FUNCTIONAL COMPONENT is Just a normal javascript function which returns some piece of jsx
const Reactcomponent = () => (
  <React.Fragment>
    {89 + 90}
    <p>{fun()}</p>
    <h1>{num}</h1>
    <p>This is my para</p>
    <h1>{new Date().toLocaleTimeString()}</h1>
  </React.Fragment>
);
setInterval(() => {
  root.render(<Reactcomponent />);
}, 2000);
```

Component composition means having one component inside another component

We can inject any javascript inside any jsx. This is the power of jsx

We can write <FirstComponent/> or we can write
<FirstComponent>
</FirstComponent>

Only useCase of writing a component as opening tag and closing tag is if we want to put some child component inside it

React.Fragment

`<React.Fragment></React.Fragment>` is a feature in React that allows you to return multiple elements from a React component by allowing you to group a list of children without adding extra nodes to the DOM. `<></>` is the shorthand tag for `React.Fragment`. The only difference between them is that the shorthand version does not support the key attribute.

```
return (
        <React.Fragment>
            <Header />
            <Navigation />
            <Main />
            <Footer />
        </React.Fragment>
    );

return (
        <>
            <Header />
            <Navigation />
            <Main />
            <Footer />
        </>
    );
```