- To create scripts for starting the project in dev mode :

package.json

```
"scripts": {
  "start": "parcel index.html",
```

- To create scripts for starting the project in prod mode :

```
"build": "parcel build index.html",
```

- To start , npm run start / npm start

- To build, npm run build.

JSX

JSX : It's a javascript syntax to create react element

- JSX is not a part of React.

- JSX make our developer life easily

To create h1 tag in JSX :

```
// JSX
const jsxHeading = <h1>Namaste React using JSX </h1>;
```

JSX

- This is not HTML inside JavaScript

→ This become React Element.

- This is JSX.

- JSX is HTML-like Syntax. / XML-like Syntax

```
import React from "react";
import ReactDOM from "react-dom/client";

const heading = React.createElement("h1", { id: "heading" }, "Namaste React");

// JSX way of writing it
const jsxHeading = <h1 id="heading">Namaste React using JSX </h1>;

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(heading);
```

→ Core React (this is not developer friendly)

→ JSX

11

↳ this code get transpiled before going to browser

JSX → transpiled (brower can understand) → Brawser

↓

Done by parcel.

↓

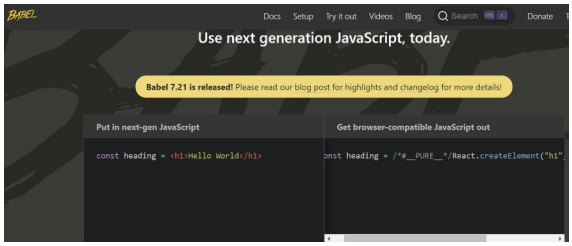
Babel (parcel installed it)

→ Babel is a javascript compiler.

↳ JSX

JSX → React createElement → React Element JS Object → HTML Element (Render)

Babel



in JSX if you need to class then,

```
// JSX way of writing it
const jsxHeading = <h1 className="head">Namaste React using JSX </h1>;
```

If you have to give attribute in JSX, the follow camel case naming.

For Multiple line wrap with ( ).

```jsx
const jsxHeading = (
  <h1 className="head">
    Namaste React using JSX
  </h1>
);
```

React Components:

- Everything is component in React (button, search box, img etc)

Two types of components:

- Class Component (old way)
- Functional Components (New way)

React Functional Components:

- Normal JS functional.
- Name it with captial letter.
- Function which return JSX code.

```jsx
// React Component
const HeadingComponent = () => {
  return <h1>Namaste React Functional Component</h1>;
};
```

with arrow f().

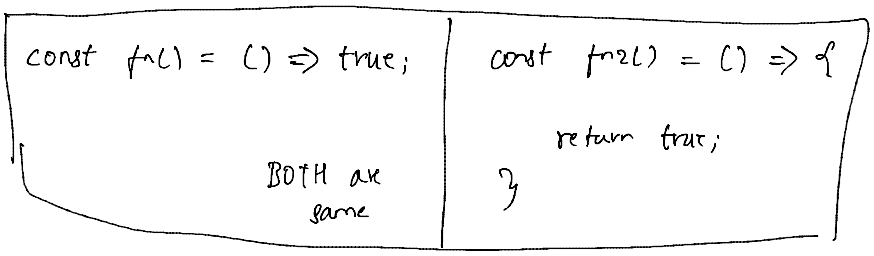
React Element.

React Element is JSX.

```jsx
// React Element
const heading = <h1 className="head">Namaste React using JSX</h1>;
```

you can skip return.

```jsx
// React Component
const HeadingComponent = () => <h1>Namaste React Functional Component</h1>;
```



Using class Name :

```
// React Component
const HeadingComponent = () => {
  return <h1 className="heading">Namaste React Functional Component</h1>;
};
```

Using Id :

```
const HeadingComponent = () => (
  <div id="container">
    <h1 className="heading">Namaste React Functional Component</h1>
  </div>
);
```

How to render Heading Component ?

```
// React Component
const HeadingComponent = () => (
  <div id="container">
    <h1 className="heading">Namaste React Functional Component</h1>
  </div>
);

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<HeadingComponent />);
```

↳ This how component is rendered

Namaste React Functional Component

other way of rendering is root.render(Header Component());

- Suppose there is two component :

now if Title has to rendered inside the container,

```
const Title = () => (
  <h1 className="head" tabIndex="5">
    Namaste React using JSX
  </h1>
);

// React Component
const HeadingComponent = () => (
  <div id="container">
    <Title />
    <h1 className="heading">Namaste React Functional Component</h1>
  </div>
);
```

This known as Composition component.

o/p

Namaste React using JSX

Namaste React Functional Component

→ Using without Arrow f(), but return is needed

```
const Title = function () {
  return (
    <h1 className="head" tabIndex="5">
      Namaste React using JSX
    </h1>
  );
};
```

Superpower of JSX: (Writing JS inside JSX) Important. Using of {}.

```
const HeadingComponent = () => (
  <div id="container">
    {} ⟵ Inside this you can run any JS code.
    <h1 className="heading">Namaste React Functional Component</h1>
  </div>
);
```

→ Now what if element inside the component?

```
const title = (
  <h1 className="head" tabIndex="5">
    Namaste React using JSX
  </h1>
);

// React Component
const HeadingComponent = () => (
  <div id="container">
    {title}
    <h1 className="heading">Namaste React Functional Component</h1>
  </div>
);
```

This is JS object so, we can call any JS code here

→ React Element inside React Element?

```
const elem = <span>React Element inside react element </span>;

const title = (
  <h1 className="head" tabIndex="5">
    {elem}
    Namaste React using JSX
  </h1>
);
```

React Element inside react element Namaste React using JSX

Namaste React Functional Component

- Some hacker want to inject something from the api and it return malicious code the attack is known as Cross Site Scri (CSS).

const data = api.getData(); //just some data.

{data}

↳ if it's inside script (it's still secure JSX make it safe)

```jsx
const Title = () => (
  <h1 id="title" key="h1">
    Namaste React
  </h1 >
);

const HeaderComponent = () => {
  return (
    <div>
      <Title />
      <h2>Namaste React Functional component</h2>
      <h2>This is a h2 tag</h2>
    </div>
  );
};
```

Functional

This is known as Component Composition.

```jsx
const Title = () => (
  <h1 className="head" tabIndex="5">
    Namaste React using JSX
  </h1>
);

// React Component
const HeadingComponent = () => (
  <div id="container">
    <Title></Title>
    <h1 className="heading">Namaste React Functional Component</h1>
  </div>
);

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<HeadingComponent />);
```

// you can call this way too.

You can call the Title function inside it.

```jsx
const Title = () => (
  <h1 className="head" tabIndex="5">
    Namaste React using JSX
  </h1>
);

// React Component
const HeadingComponent = () => (
  <div id="container">
    {Title()}
    <h1 className="heading">Namaste React Functional Component</h1>
  </div>
);

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<HeadingComponent />);
```

as we can call any JS code so we can call it as fn().

```
    <h1 className="heading">Namaste React Functional Component</h1>
  </div>
);

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<HeadingComponent />);
```

as f().

JSX can only have single parent on the root.

Const jsx = <h1> JSX </h1> <h1> second </h1> //Invalid

To fix this make a div and write it

```
Const jsx = <div>
              <h1>JSX </h1>
              <h1> second </h1>
            </div>
```

It will work but already one root is there in HTML.

Better way to do it :

By Using React Fragments :

```
// React Fragments
const HeadingComponent = () => (
  <React.Fragment>
    <div id="container-1">
      <Title />
      <h1 className="heading">Namaste React Functional Component</h1>
    </div>
    <div id="container-2"></div>
  </React.Fragment>
);
```

- React Fragment is now only one parent of the JSX

```
<div id="root"> == $0
  <div id="container-1"> ... </div>
  <div id="container-2"></div>
</div>
```

} you can see only one root

- React fragment is an empty tag.

React Fragment == <>

```jsx
// React Fragments
const HeadingComponent = () => (
  <>
    <div id="container-1">
      <Title />
      <h1 className="heading">Namaste React Functional Component</h1>
    </div>
    <div id="container-2"></div>
  </>
);
```

Can we have React fragment inside React fragment?

=> Yes

E.g.

```
<>
    <>
        <Header/>
    </>
        <Body>
</>
```

Inline Style in JSX :

```jsx
const styleObj = {
  backgroundColor: "red",
}

// Inline styling in React
const jsx = (
  <div style={styleObj}>
    <h1>JSX</h1>
    <h2>Second JSX</h2>
  </div>
);
```

OR

```jsx
// Inline styling in React
const jsx = (
  <div
    style={{
      backgroundColor: "red",
    }}
  >
    <h1>JSX</h1>
    <h1>Second JSX</h1>
  </div >
);
```

1:02

```jsx
const styleObj = {
  backgroundColor: "red",
}

// Inline styling in React
const jsx = (
  <div style={styleObj}>
    <h1>JSX</h1>
    <h2>Second JSX</h2>
  </div>
);


const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(jsx);
```

// because here only you can write any pieces of code (JS code)

2nd way of styling Using className :

```
68 // Inline styling in React
69 const jsx = (
70   <div
71     className="jsx"
72   >
73     <h1>JSX</h1>
74     <h2>Second JSX</h2>
75   </div>
76 );
```

```
17 .jsx {
18     border: 1px solid red
19 }
```

JSX

Second JSX