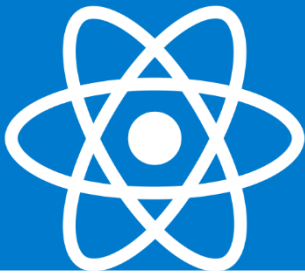




# React JS - JSX

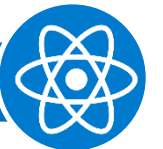
#React Notes

| JSX



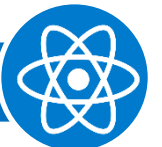
# Introduction to JSX

- JSX is a technology that was introduced by React.
- JSX stands for JavaScript XML.
- Although React can work completely fine without using JSX, it's an ideal technology to work with components, so React benefits a lot from JSX.
- At first, you might think that using JSX is like mixing HTML and JavaScript
- <https://reactjs.org/docs/introducing-jsx.html>



# What is JSX?

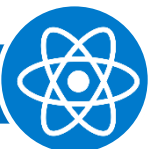
- JavaScript extension, JSX is not valid JavaScript that browsers can read.
- It's a JavaScript file that contains JSX code, very similar to HTML and the file have to be compiled before it reaches web browser, with JSX compiler that translates the JSX into JavaScript.
- JSX element are treated as JavaScript expression, they can be saved as variable, passed to function or stored in an object or array.
- Each JSX expression must have exactly one outermost element. Usually wrap the JSX expression in a `<div></div>`.



## Cont.

- JSX must have a closing tag, even if it's a self-closing.
  - `<Contact />`
- To evaluate JSX expressions you must wrap it in curly braces.  
`{variable}`

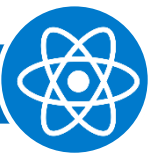
Rendering JSX element means to make it appear on screen. We render JSX expression like this:



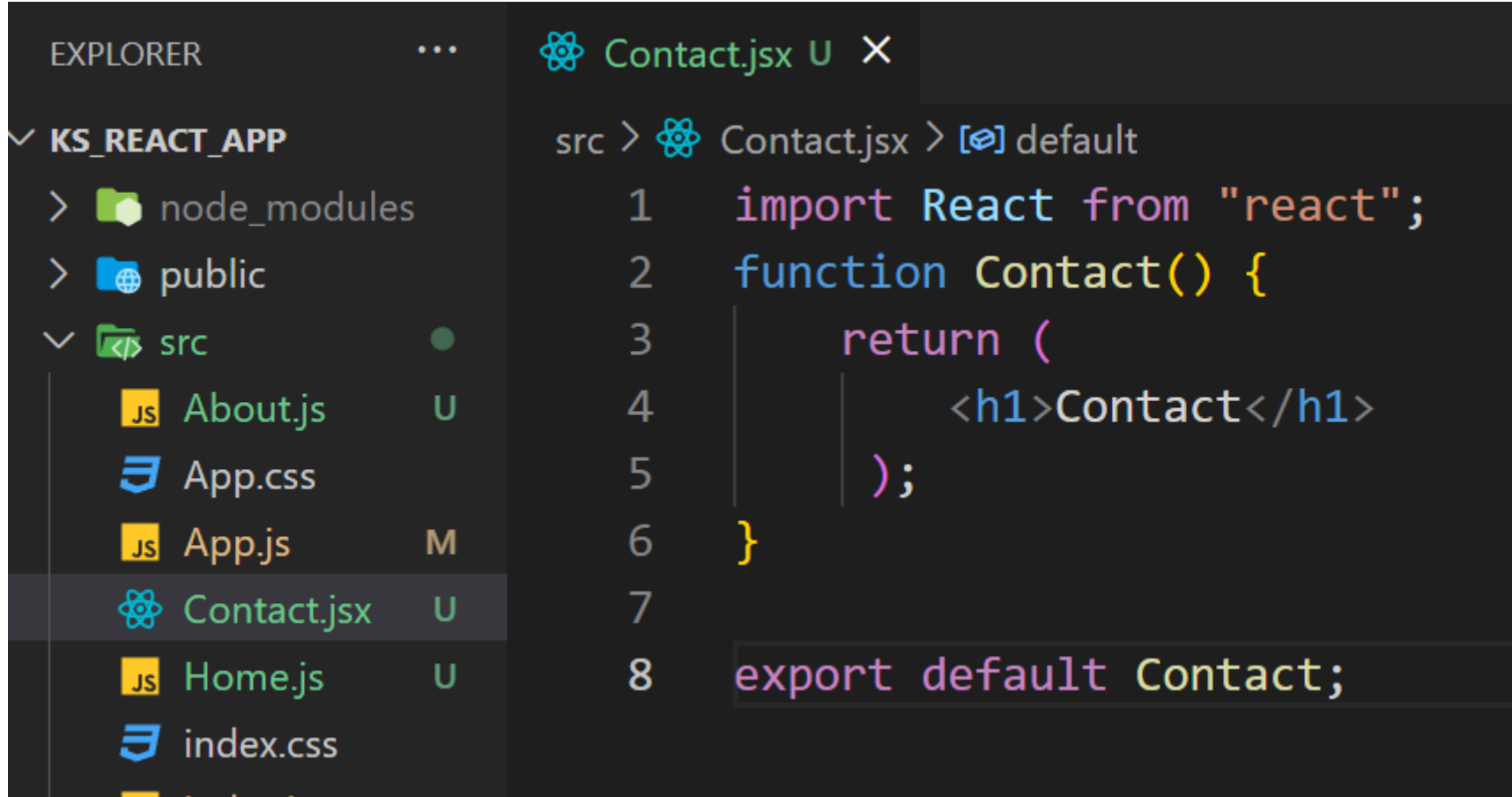
# React Element Render

- import React from 'react';
- import ReactDOM from 'react-dom';
- ReactDOM.render(<h1>Hello</h1>, document.getElementById('app'));

```
src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import './index.css';
4  import App from './App';
5  import reportWebVitals from './reportWebVitals';
6
7  const root = ReactDOM.createRoot(document.getElementById('root'));
8  root.render(
9    <React.StrictMode>
10   |   <App />
11   | </React.StrictMode>
12 );
```



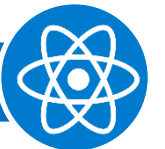
# Create File with .jsx



The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORER' sidebar displays the file structure of a project named 'KS\_REACT\_APP'. The 'src' directory is expanded, showing files: 'About.js', 'App.css', 'App.js', 'Contact.jsx' (highlighted), 'Home.js', and 'index.css'. On the right, the 'Contact.jsx' file is open in the editor. The breadcrumb navigation shows 'src > Contact.jsx > [default]'. The code in the editor is as follows:

```
1 import React from "react";
2 function Contact() {
3   return (
4     <h1>Contact</h1>
5   );
6 }
7
8 export default Contact;
```

- <https://babeljs.io/>





# JSX Convert Code Into JavaScript

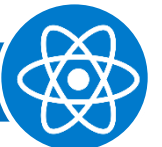
## JSX Represents Objects

Babel compiles JSX down to `React.createElement()` calls.

These two examples are identical:

```
const element = (  
  <h1 className="greeting">  
    Hello, world!  
  </h1>  
);
```

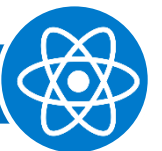
```
const element = React.createElement(  
  'h1',  
  {className: 'greeting'},  
  'Hello, world!'  
);
```



- Inside a JSX expression, attributes can be inserted very easily:

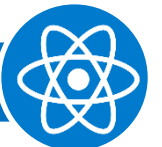


```
const myId = 'test'  
const element = <h1 id={myId}>Hello, world!</h1>
```



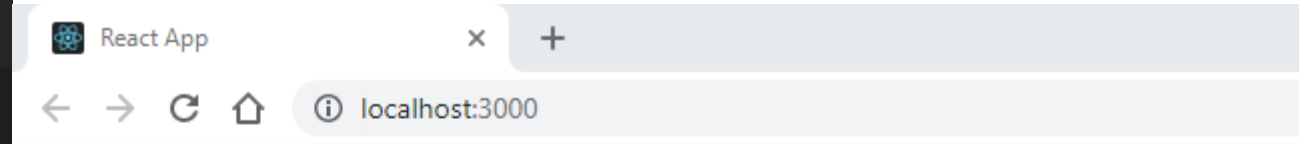
# Comment in React JSX

- You can use regular `/* Block Comments */`, but they need to be wrapped in curly braces
- Single Line Comment
  - `{/* A JSX Single Line comment */}`
- Multiline Comments
  - `{/*`
  - Multi
  - line
  - comment
  - `*/}`

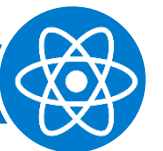


# Comments Demo

```
JS App.js M X
src > JS App.js > ...
1  import './App.css';
2  function App() {
3    return (
4      <div>
5        <h1>Comment Demo</h1>
6        {/* Single Line Comment */ }
7
8        {/* Multiline
9         Line
10        Comment */}
11      </div>
12    );
13  }
14
15
16  export default App;
17
```



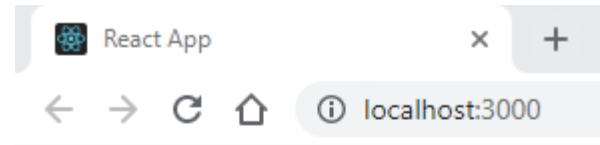
## Comment Demo



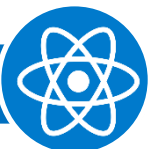
# Expression

- Arithmetic Expression in React
- Use {}

```
JS App.js M X
src > JS App.js > [default]
1 function App() {
2   return (
3     <h1>{10+10}</h1>
4   );
5 }
6 export default App;
7
```



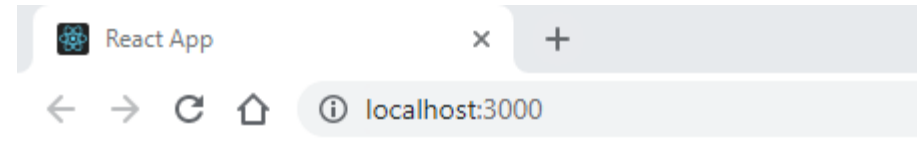
20



# Variable Expression

- Variable name must be wrap in {}.

```
JS App.js M X
src > JS App.js > ...
1 function App() {
2   var a = 10;
3   return (
4     <h1>A Value is {a}</h1>
5   );
6 }
7 export default App;
8
```

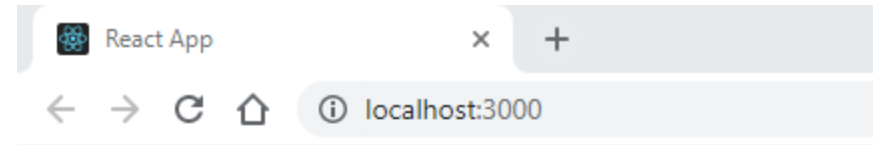


**A Value is 10**



# Sum Example

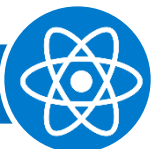
```
Js App.js M X
src > Js App.js > [default]
1  function App() {
2      var a = 10;
3      var b = 10;
4      return (
5          <div>
6              <h1>A Value is {a}</h1>
7              <h1>B Value is {b}</h1>
8              <h1>Sum is {a+b}</h1>
9          </div>
10     );
11 }
12 export default App;
13
```



**A Value is 10**

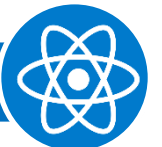
**B Value is 10**

**Sum is 20**



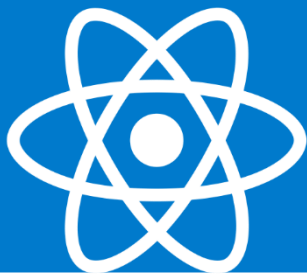
# If Demo

```
Js App.js M X
src > Js App.js > App
1  function App() {
2      var a = 10;
3      var b = 20;
4      var c = a + b;
5      var msg = "";
6      if (c > 50) {
7          msg = "Sum is < 50";
8      }else{
9          msg = "Sum is > 50";
10     }
11
12     return (
13         <div>
14             <h1>A Value is {a}</h1>
15             <h1>B Value is {b}</h1>
16             <h1>Sum is {c}</h1>
17             {msg}
18         </div>
19     );
20 }
21 export default App;
```



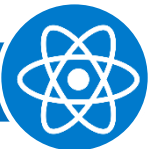


# | Component Styling



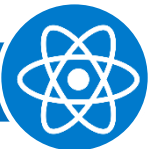
# How to style React components

- Mainly there are 3 methods to design component
  1. Inline styling
  2. styled-components (Internal)
  3. CSS Modules (External)



# Camel Case for Property

- Note : All CSS and JS properties must be written as per below format
- Rename
- HTML -> React
- Class – className
- for – htmlFor
- Value - DefaultValue
  
- onclick - onClick
- tabIndex - tabIndex



# Camel Case for Property

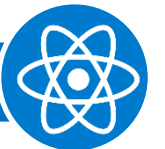
In React Css Properties Name will be change. Remove – and use Capital Letter

## CSS

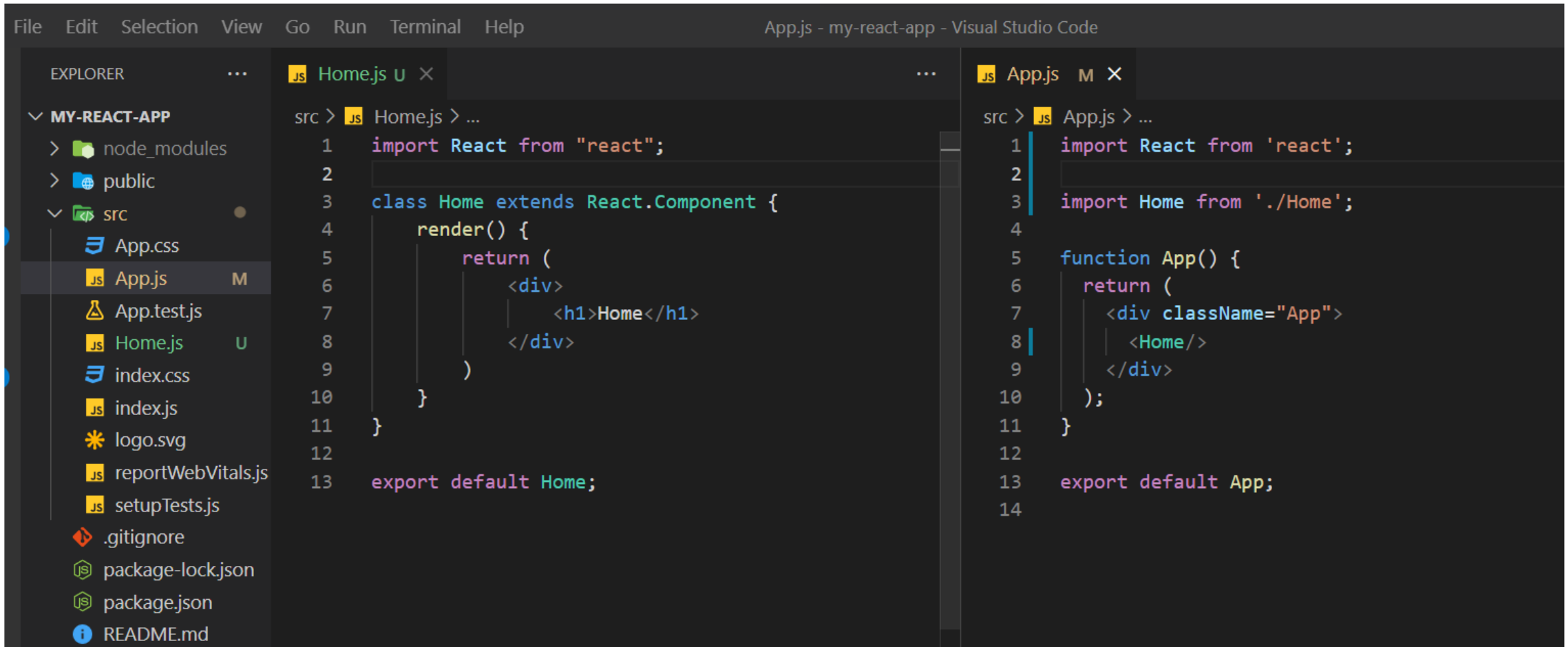
- background-color: #44014C;
- min-height: 200px;
- box-sizing: border-box;

## JS

- backgroundColor: "#44014C",
- minHeight: "200px",
- boxSizing: "border-box"



# Home Component



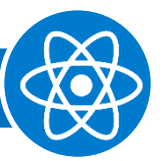
The screenshot shows the Visual Studio Code editor with two files open: `Home.js` and `App.js`. The Explorer sidebar on the left shows the project structure for `MY-REACT-APP`, including `node_modules`, `public`, and `src` folders. The `src` folder contains `App.css`, `App.js`, `App.test.js`, `Home.js`, `index.css`, `index.js`, `logo.svg`, `reportWebVitals.js`, `setupTests.js`, `.gitignore`, `package-lock.json`, `package.json`, and `README.md`.

The `Home.js` file contains the following code:

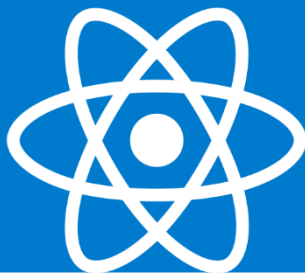
```
src > JS Home.js > ...
1  import React from "react";
2
3  class Home extends React.Component {
4      render() {
5          return (
6              <div>
7                  <h1>Home</h1>
8              </div>
9          )
10     }
11 }
12
13 export default Home;
```

The `App.js` file contains the following code:

```
src > JS App.js > ...
1  import React from 'react';
2
3  import Home from './Home';
4
5  function App() {
6      return (
7          <div className="App">
8              <Home/>
9          </div>
10     );
11 }
12
13 export default App;
```

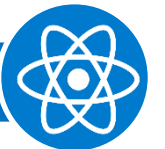


# | Inline Style

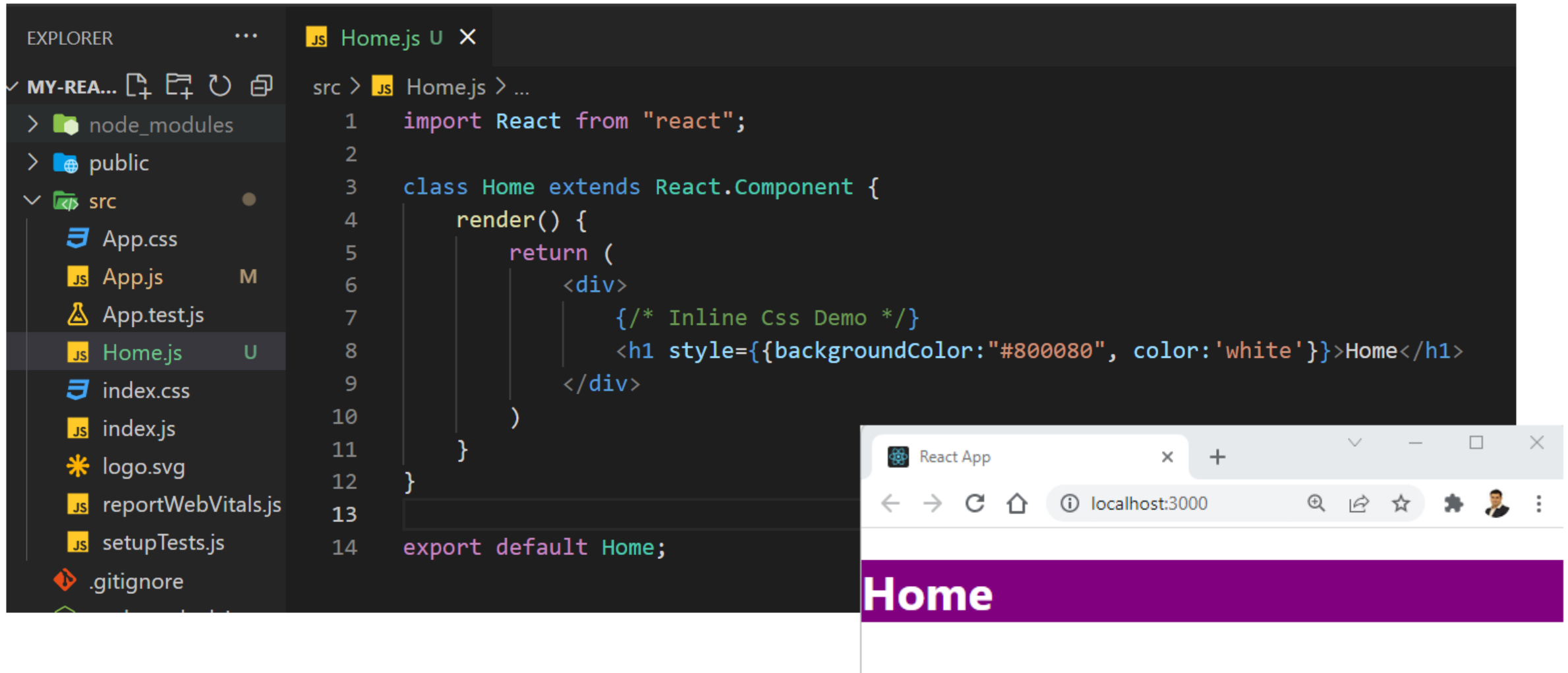


# Inline Css

- If you are familiar with basic HTML, you'll know that it is possible to add your CSS inline. This is similar in React.
- We can add inline styles to any React component we want to render. These styles are written as attributes and are passed to the element.



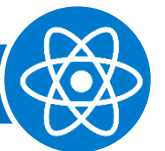
# Inline CSS Class Component Example



The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with files like `App.css`, `App.js`, `App.test.js`, `Home.js`, `index.css`, `index.js`, `logo.svg`, `reportWebVitals.js`, `setupTests.js`, and `.gitignore`. The code editor shows the `Home.js` file with the following code:

```
1 import React from "react";
2
3 class Home extends React.Component {
4   render() {
5     return (
6       <div>
7         {/* Inline Css Demo */}
8         <h1 style={{backgroundColor:"#800080", color:'white'}}>Home</h1>
9       </div>
10     )
11   }
12 }
13
14 export default Home;
```

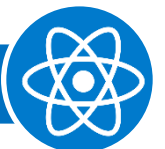
Below the code editor, a browser window titled "React App" is shown, displaying the rendered output of the component: a purple rectangle with the word "Home" in white text.





# Inline Css Demo With Function Component

```
Js App.js M X
src > Js App.js > App
1 | import './App.css';
2 | function App() {
3 |   return (
4 |     <div>
5 |       {/* InLine Css Demo */}
6 |       <h1 style={{backgroundColor: 'blue',color: 'white'}} >InLine Style</h1>
7 |     </div>
8 |   );
9 | }
10
11 export default App;
12
```

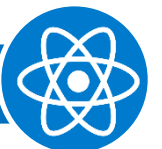


# Inline Css Code

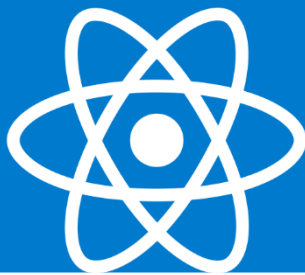
```
import React from "react";

class Home extends React.Component {
  render() {
    return (
      <div>
        {/* Inline Css Demo */}
        <h1 style={{backgroundColor:"#800080", color:'white'}}>Home</h1>
      </div>
    )
  }
}

export default Home;
```



# | Internal Css

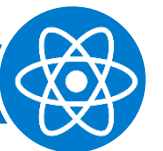


# Internal Css

- We create a style object variable the same way we create a JavaScript object.
- This object is then passed to the style attribute of the element we want to style.

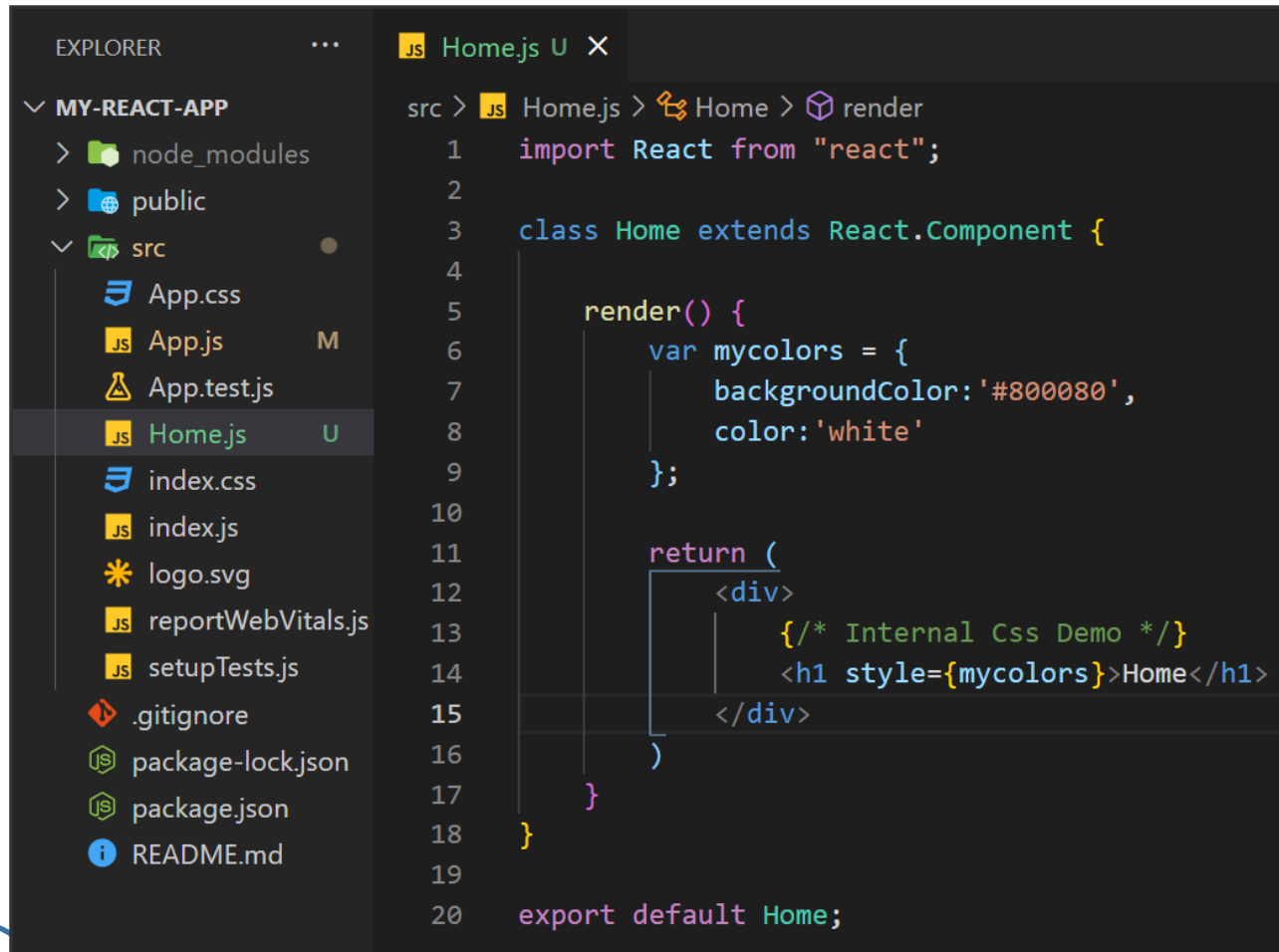
```
var mycolors = {  
    backgroundColor:'#800080',  
    color:'white'  
};
```

```
<h1 style={mycolors}>Home</h1>
```

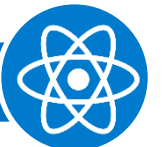
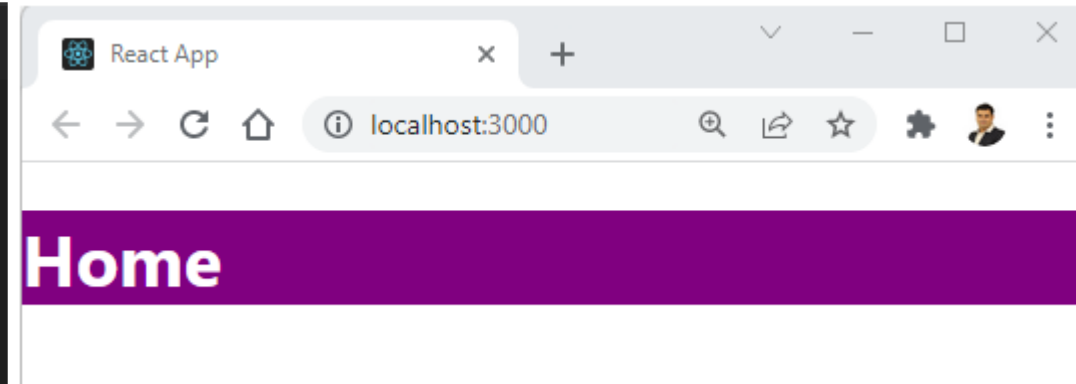


# Internal Css

- We can Create JS Object and Pass in Style

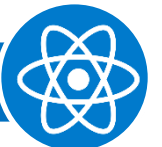


```
src > JS Home.js > Home > render
1  import React from "react";
2
3  class Home extends React.Component {
4
5      render() {
6          var mycolors = {
7              backgroundColor: '#800080',
8              color: 'white'
9          };
10
11          return (
12              <div>
13                  { /* Internal Css Demo */ }
14                  <h1 style={mycolors}>Home</h1>
15              </div>
16          )
17      }
18  }
19
20  export default Home;
```



# Functional Component

```
Js App.js M X
src > Js App.js > ...
1  function App() {
2
3      var mycolors = {
4          backgroundColor: '#800080',
5          color: 'white',
6      }
7
8      return (
9          <div>
10             { /* Internal Css Demo */ }
11             <h1 style={mycolors}>App Component</h1>
12          </div>
13      );
14  }
15
16  export default App;
17
```



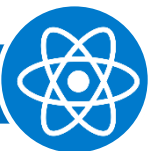
# Internal Code

```
import React from "react";
class Home extends React.Component {

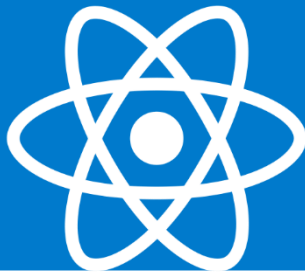
  render() {
    var mycolors = {
      backgroundColor:'#800080',
      color:'white'
    };

    return (
      <div>
        { /* Internal Css Demo */ }
        <h1 style={mycolors}>Home</h1>
      </div>
    )
  }
}

export default Home;
```



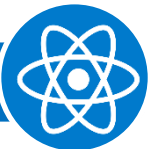
# | External





# External Css

- Create new css file and design code.
- To Add css in Component we can use below code.
  - `import './Mystyle.css';`
- To Access Class in React Component we can call using
  - `className`



# External Css

EXPLORER

MY-REACT-APP

- node\_modules
- public
- src
  - App.css
  - App.js
  - App.test.js
  - Home.js
  - index.css
  - index.js
  - logo.svg
  - Mystyle.css
  - reportWebVitals.js
  - setupTests.js
  - .gitignore
  - package-lock.json
  - package.json
  - README.md

Mystyle.css

```
1
```

src > Mystyle.css

Home.js

```
1 import React, { Component } from 'react';
2 import './Mystyle.css';
3 class Home extends Component{
4   render()
5   {
6     return(
7       <div>
8         <h1 className="mytext">Home</h1>
9       </div>
10     )
11   }
12 }
13 export default Home;
```

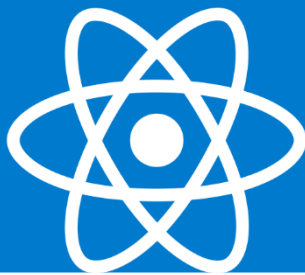
src > Mystyle.css > .mytext

```
1 .mytext{
2   color: white;
3   background-color: blueviolet;
4   padding: 15px;
5   text-align: center;
6   text-transform: uppercase;
7 }
```

HOME

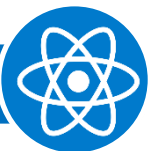


# | Component Reusability



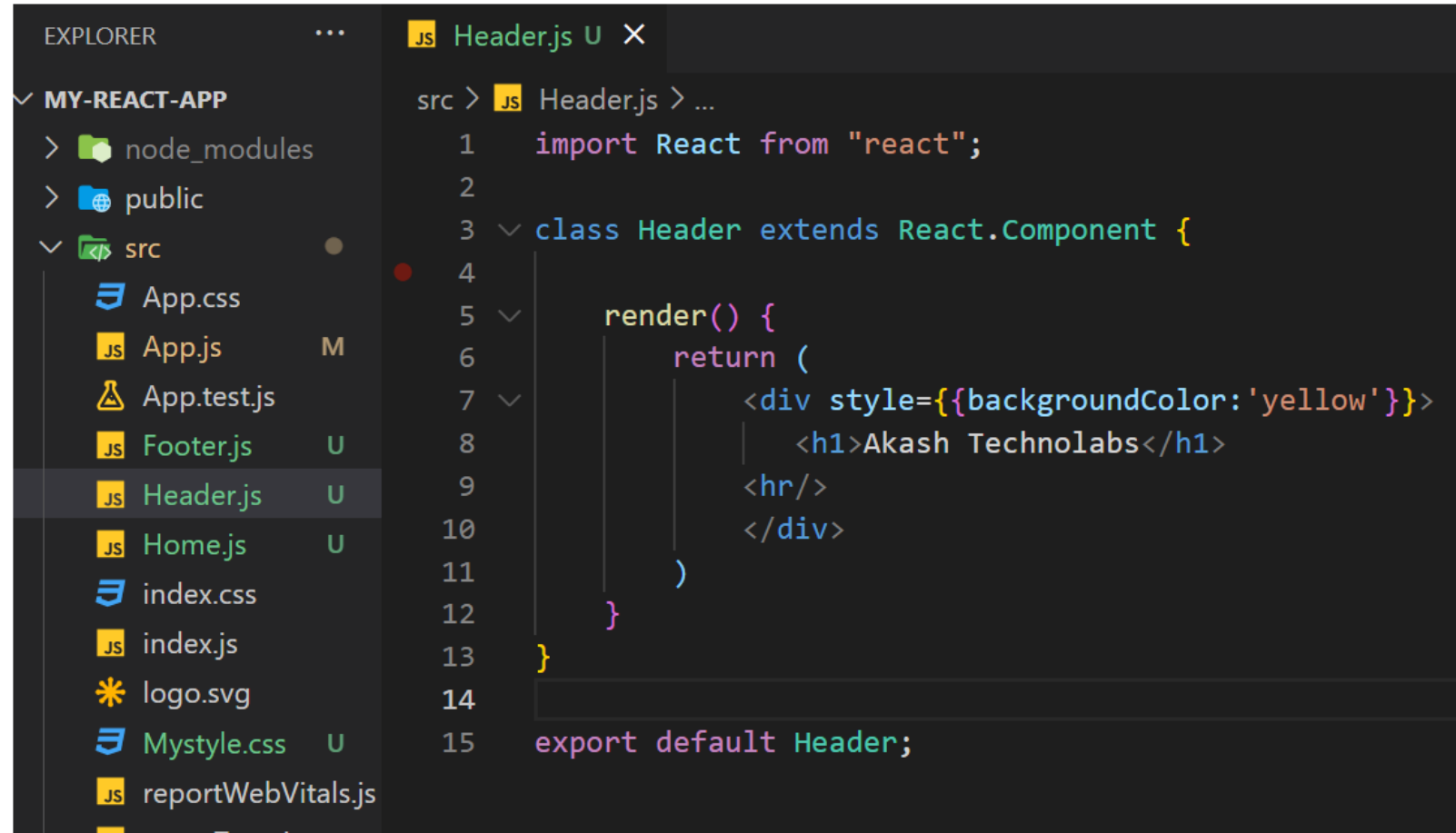
# Common Component

- We can use same component in multiple Component
- Example:
  - Header and Footer will remain same in Home,About,Contact
- We can Create below component and will reuse in Other Component.
  - Header
  - Footer



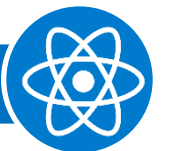
# Header.js

- Header Component having Title.



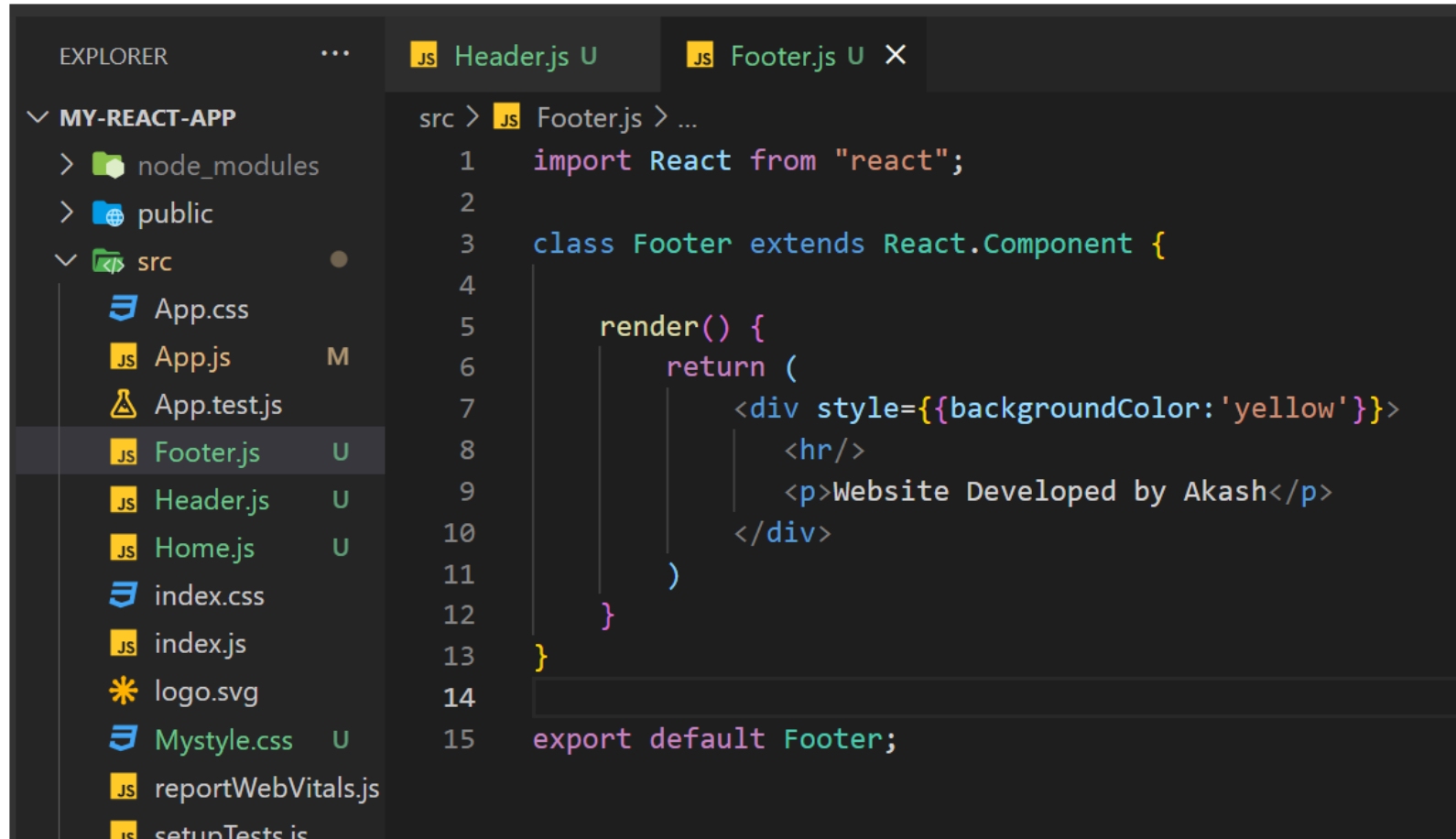
```
EXPLORER
MY-REACT-APP
  node_modules
  public
  src
    App.css
    App.js
    App.test.js
    Footer.js
    Header.js
    Home.js
    index.css
    index.js
    logo.svg
    Mystyle.css
    reportWebVitals.js

src > JS Header.js > ...
1  import React from "react";
2
3  class Header extends React.Component {
4
5      render() {
6          return (
7              <div style={{backgroundColor:'yellow'}}>
8                  <h1>Akash Technolabs</h1>
9                  <hr/>
10                 </div>
11             )
12         }
13     }
14
15     export default Header;
```



# Footer

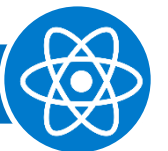
- Footer Component having Footer Text



The screenshot shows a code editor with two tabs: 'Header.js' and 'Footer.js'. The 'Footer.js' tab is active, displaying the following code:

```
src > JS Footer.js > ...
1  import React from "react";
2
3  class Footer extends React.Component {
4
5      render() {
6          return (
7              <div style={{backgroundColor:'yellow'}}>
8                  <hr/>
9                  <p>Website Developed by Akash</p>
10             </div>
11          )
12      }
13  }
14
15  export default Footer;
```

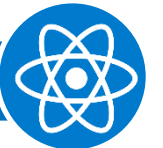
The Explorer panel on the left shows the project structure for 'MY-REACT-APP', with 'Footer.js' highlighted in the 'src' directory.



# Home

- Call Header and Footer Component in Home Component.

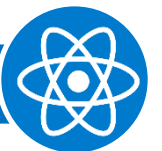
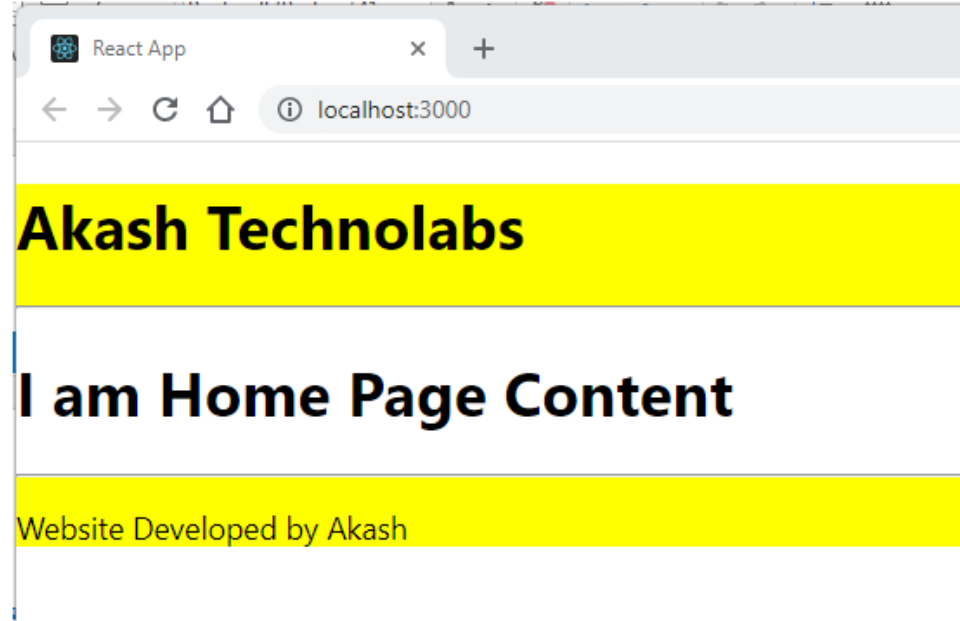
```
JS Home.js U X
src > JS Home.js > Home > render
1 import React from "react";
2 import './Mystyle.css';
3 import Header from './Header';
4 import Footer from './Footer';
5 class Home extends React.Component {
6
7   render() {
8     return (
9       <div>
10         <Header/>
11         <h1>I am Home Page Content</h1>
12         <Footer/>
13       </div>
14     )
15   }
16 }
17
18 export default Home;
```



# App

- Home Component call in App Component

```
Js App.js M X
src > Js App.js > App
1  import React from 'react';
2
3  import Home from './Home';
4
5  function App() {
6    return (
7      <div className="App">
8        <Home/>
9      </div>
10   );
11 }
12
13 export default App;
14
```

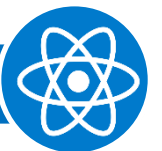




# About

- Create New Component About and Call Header and Footer.

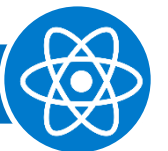
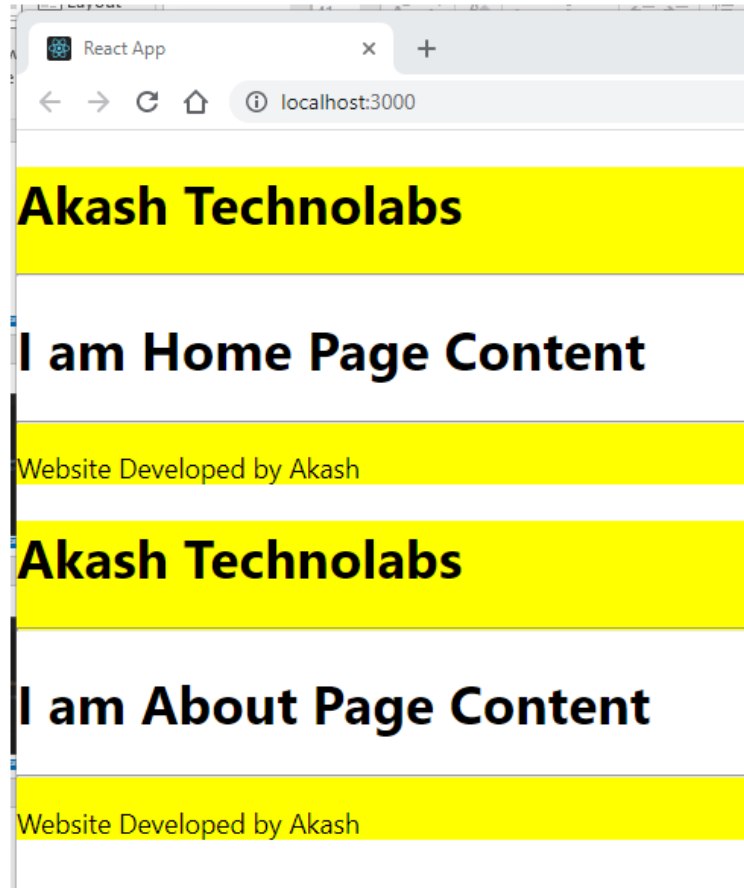
```
JS About.js U X
src > JS About.js > [?] default
1  import React from "react";
2  import './Mystyle.css';
3  import Header from './Header';
4  import Footer from './Footer';
5  class About extends React.Component {
6
7      render() {
8          return (
9              <div>
10                 <Header/>
11                 <h1>I am About Page Content</h1>
12                 <Footer/>
13             </div>
14         )
15     }
16 }
17
18 export default About;
```



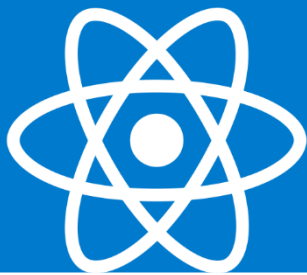
# App.js

- In App Component we can call multiple Component
- Home
- About

```
Js App.js M X
src > Js App.js > App
1  import React from 'react';
2
3  import Home from './Home';
4  import About from './About';
5
6  function App() {
7    return (
8      <div className="App">
9        <Home/>
10       <About/>
11     </div>
12   );
13 }
14
15
16 export default App;
17
```



# | `<>` Fragments `</>`

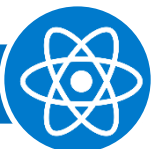


# React Fragments

- In React, whenever you want to render something on the screen, you need to use a render method inside the component.
- This render method can return **single** elements or **multiple** elements.
- The render method will only render a single root node inside it at a time.
- However, if you want to return multiple elements, the render method will require a '**div**' tag and put the entire content or elements inside it.

```
Module build failed (from ./node_modules/babel-loader/lib/index.js):  
SyntaxError: D:\code\react_code\myapp21\src\App.js: Adjacent JSX elements must be wrapped  
in an enclosing tag. Did you want a JSX fragment <>...</>? (7:8)
```

5 |

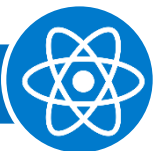


# Div Wrap

- Div required to print multiple Items.

```
JS App.js M X
src > JS App.js > [e] default
1  import React from 'react';
2  class App extends React.Component {
3    render() {
4      return (
5        <div>
6          <p>Child 1</p>
7          <p>Child 2</p>
8          <p>Child 3</p>
9          <p>Child 4</p>
10       </div>
11     )
12   }
13 }
14 export default App;
15
```

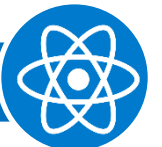
React App  
localhost:3000  
Child 1  
Child 2  
Child 3  
Child 4



# React Fragments

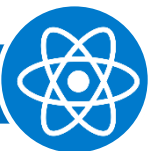
- React introduced **Fragments** from the **16.2** and above version.
- Fragments allow you to group a list of children without adding extra nodes to the DOM.

```
<React.Fragment>  
  <h2> child1 </h2>  
  <p> child2 </p>  
  .. .....  
</React.Fragment>
```



# Why to use Fragments?

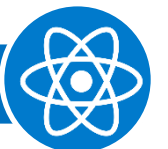
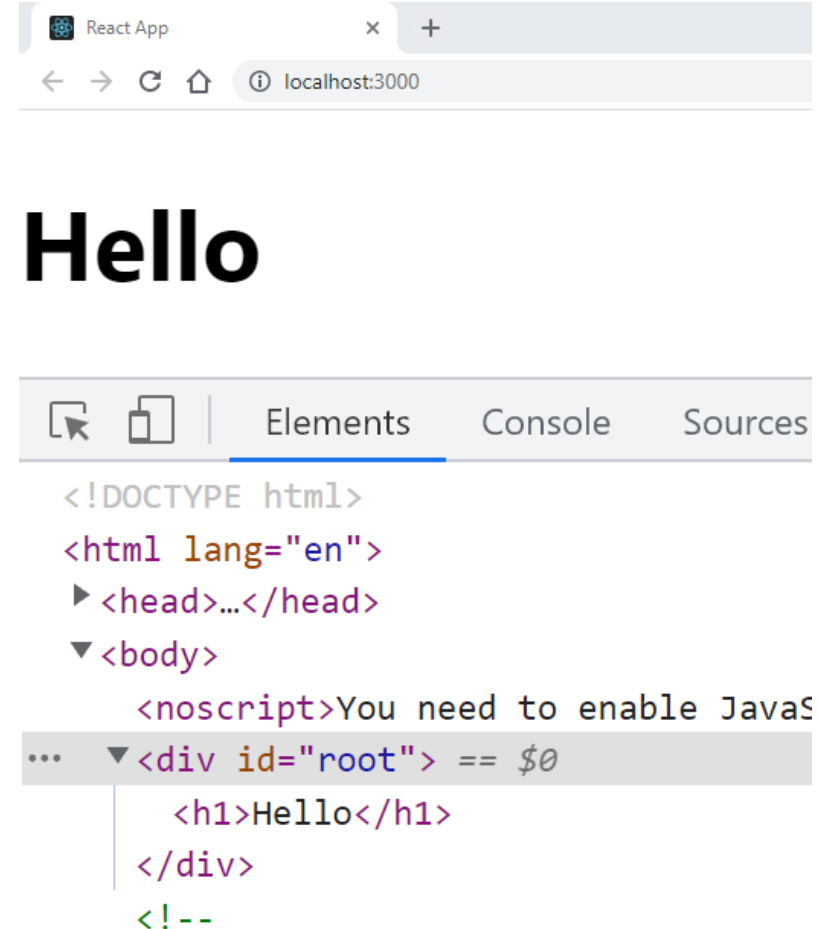
- The main reason to use Fragments tag is:
- It makes the execution of code faster as compared to the div tag.
- It takes less memory.



# Default Div Render

- By Default Component will render in Div Tag of id Root.

```
JS App.js M X
src > JS App.js > ...
1  function App() {
2      return (
3          <h1>Hello</h1>
4      );
5  }
6  export default App;
7
```





- To print multiple Items Div is required

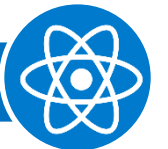
```
App.js 2, M X
src > App.js > App
1 function App() {
2   return (
3     <h1>Hello</h1>
4     <h1>Hello</h1>
5   );
6 }
7 export default App;
8

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL
node + - [ ] [X]

at Parser.jsxParseElement (D:\code\react_code\myapp1\node_modules\@babel\parser\lib\index.js:8020:17)
at Parser.parseExprAtom (D:\code\react_code\myapp1\node_modules\@babel\parser\lib\index.js:8034:19)
at Parser.parseExprSubscripts (D:\code\react_code\myapp1\node_modules\@babel\parser\lib\index.js:12648:23)
at Parser.parseUpdate (D:\code\react_code\myapp1\node_modules\@babel\parser\lib\index.js:12627:21)
at Parser.parseMaybeUnary (D:\code\react_code\myapp1\node_modules\@babel\parser\lib\index.js:12598:23)
at Parser.parseMaybeUnaryOrPrivate (D:\code\react_code\myapp1\node_modules\@babel\parser\lib\index.js:12392:61)

ERROR in [eslint]
src\App.js
  Line 4:4:  Parsing error: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>?
)

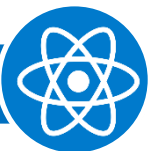
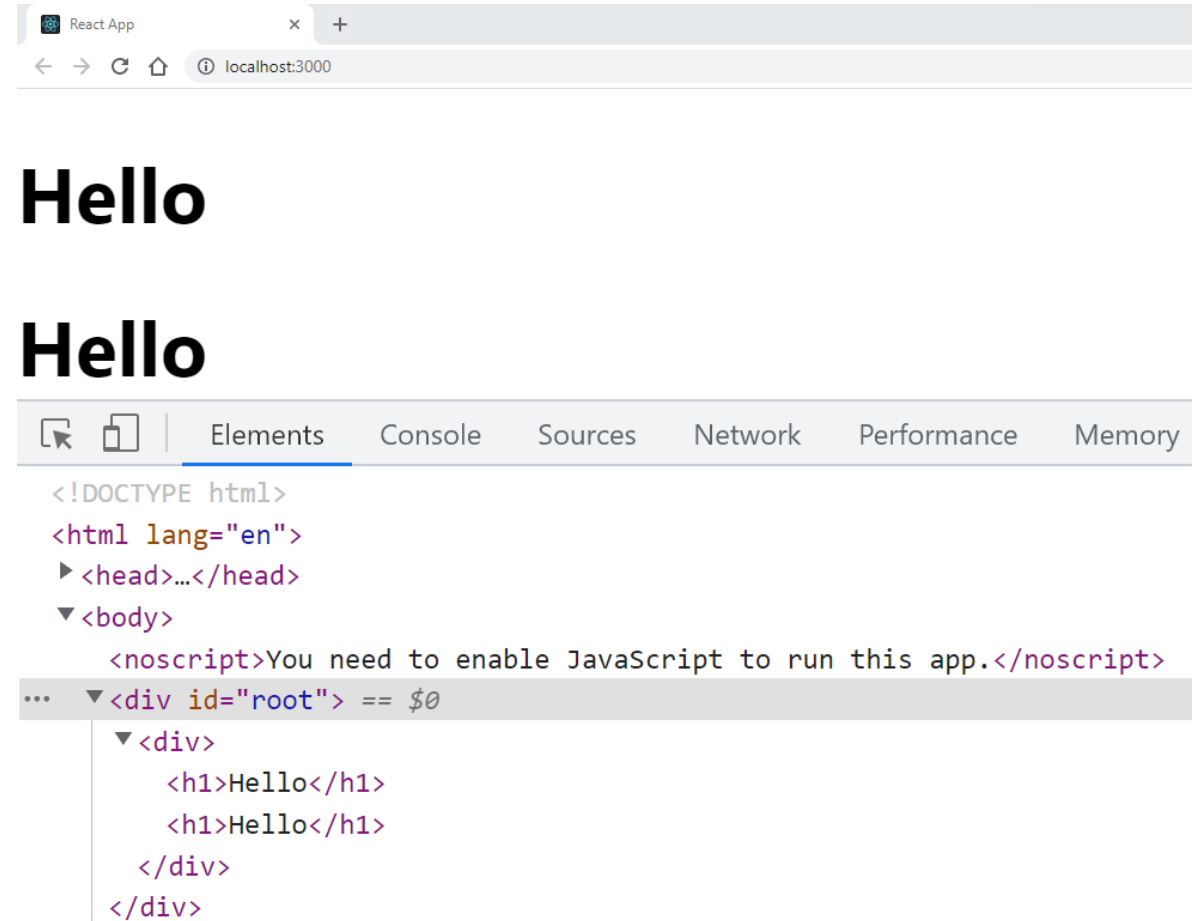
webpack compiled with 2 errors
```



# Nested Div

- Div id root having div to print multiple items.

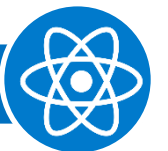
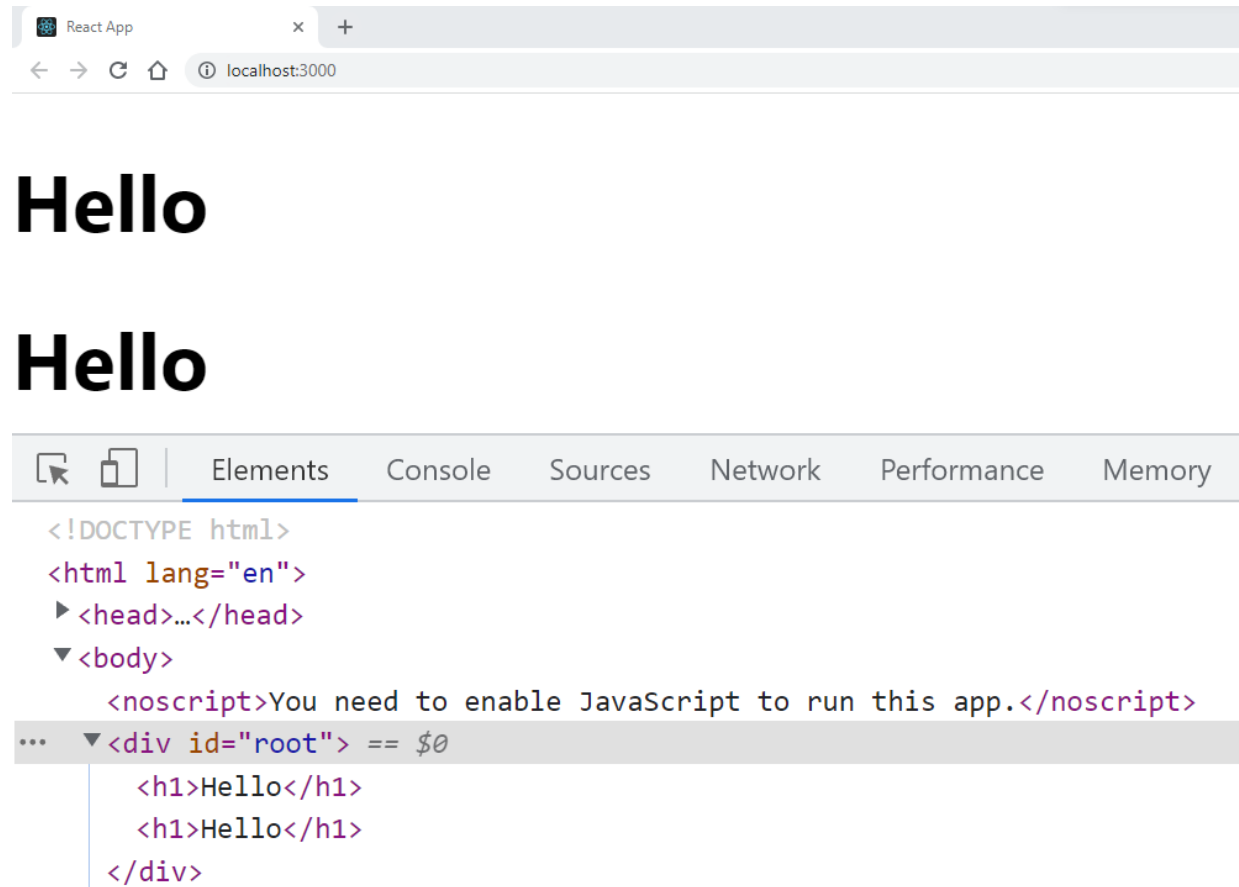
```
JS App.js M X
src > JS App.js > [🔗] default
1  function App() {
2      return (
3          <div>
4              <h1>Hello</h1>
5              <h1>Hello</h1>
6          </div>
7      );
8  }
9  export default App;
```



# React.Fragment

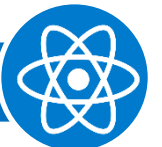
- React.Fragment will help to render multiple items.

```
JS App.js M X
src > JS App.js > App
1  import React from "react";
2
3  function App() {
4    return (
5      <React.Fragment>
6        <h1>Hello</h1>
7        <h1>Hello</h1>
8      </React.Fragment>
9    );
10 }
11 export default App;
12
```



# Fragments Short Syntax

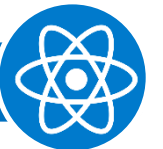
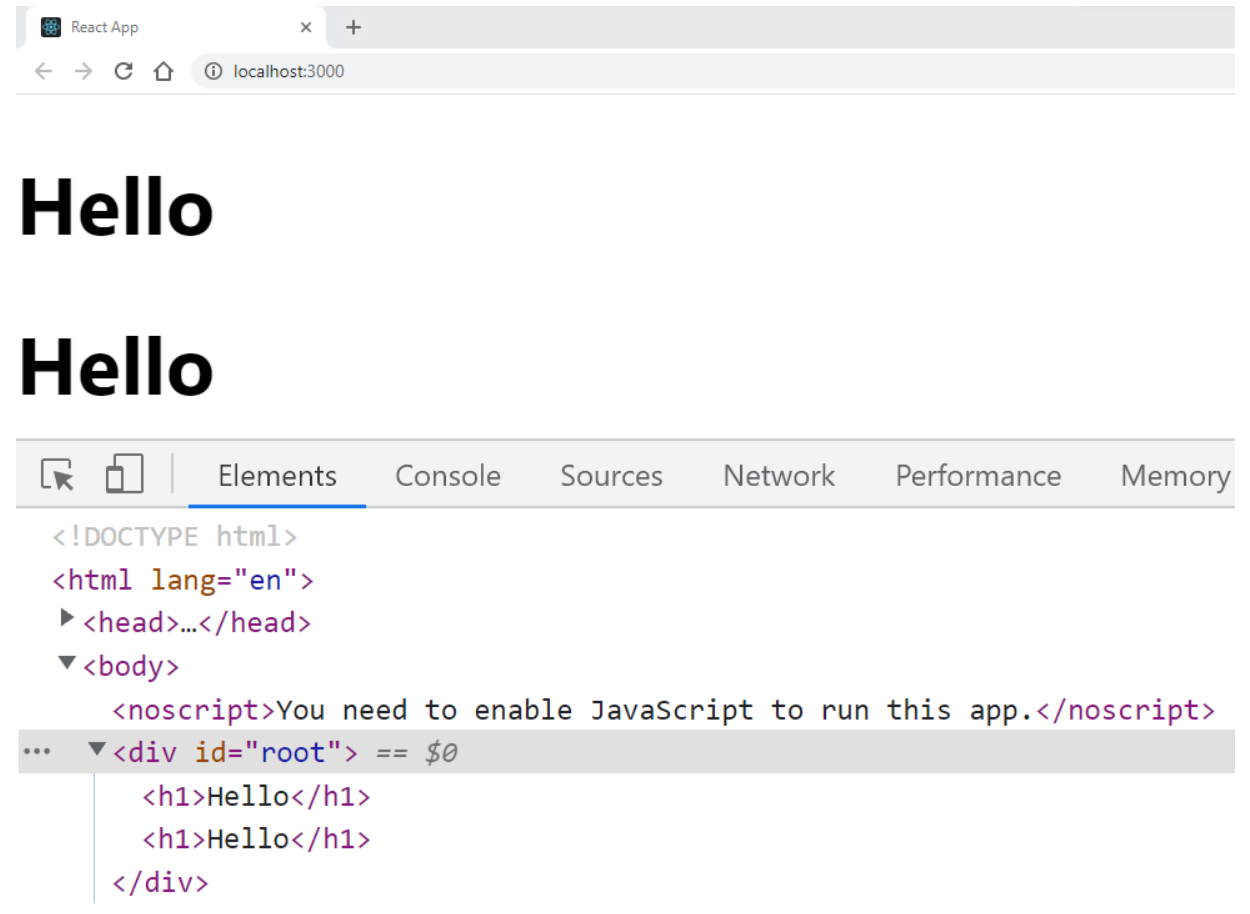
- There is also another shorthand exists for declaring fragments for the above method.
- It looks like **empty** tag in which we can use of '<>' and '' instead of the 'React.Fragment'.
- **Note:** The shorthand syntax does not accept key attributes in that case you have to use the <React.Fragments> tag.



# Empty <> tag

- We can use <></> Empty tag.

```
JS App.js M X
src > JS App.js > App
1  import React from "react";
2
3  function App() {
4    return (
5      <>
6        <h1>Hello</h1>
7        <h1>Hello</h1>
8      </>
9    );
10 }
11 export default App;
12
```



JS App.js M X

src > JS App.js > App > render

```
1 import React from 'react';
2 class App extends React.Component {
3   render() {
4     return (
5       <React.Fragment>
6         <p>Child 1</p>
7         <p>Child 2</p>
8         <p>Child 3</p>
9         <p>Child 4</p>
10      </React.Fragment>
11    )
12  }
13 }
14 export default App;
15
```

React App

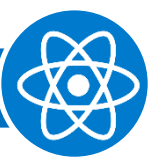
localhost:3000

Child 1

Child 2

Child 3

Child 4

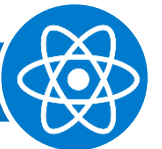


# Fragments Short Syntax

```
JS App.js M X
src > JS App.js > App > render
1  import React from 'react';
2  class App extends React.Component {
3    render() {
4      return (
5        <>
6          <p>Child 1</p>
7          <p>Child 2</p>
8          <p>Child 3</p>
9          <p>Child 4</p>
10       </>
11     )
12   }
13 }
14 export default App;
15
```

React App  
localhost:3000

Child 1  
Child 2  
Child 3  
Child 4

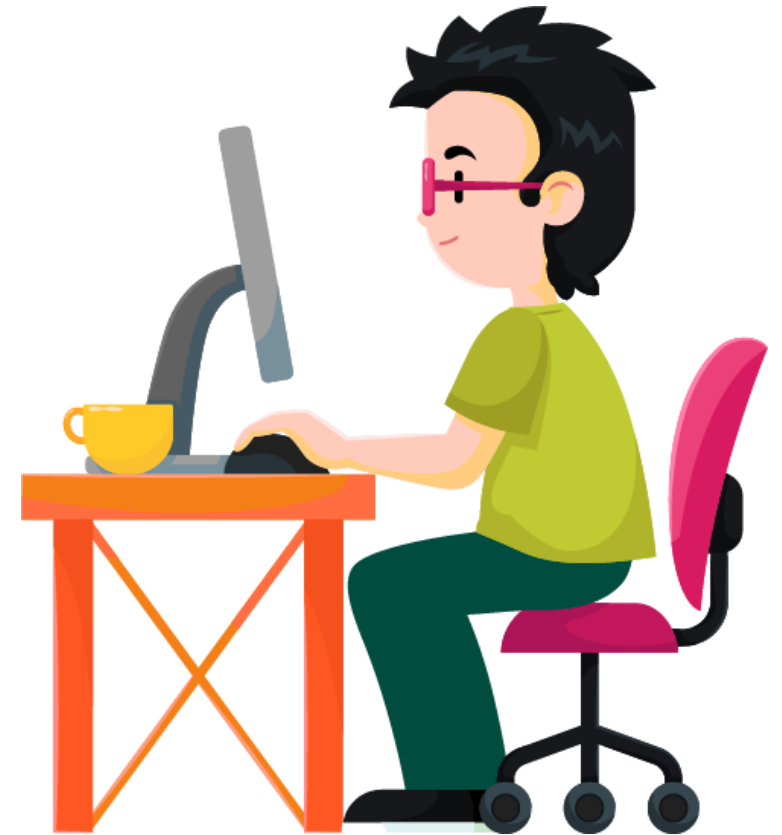


# Get Exclusive Video Tutorials



[www.apptutorials.com](http://www.apptutorials.com)

<https://www.youtube.com/user/Akashtips>







Get More Details

[www.akashsir.com](http://www.akashsir.com)



# If You Liked It !

## Rating Us Now



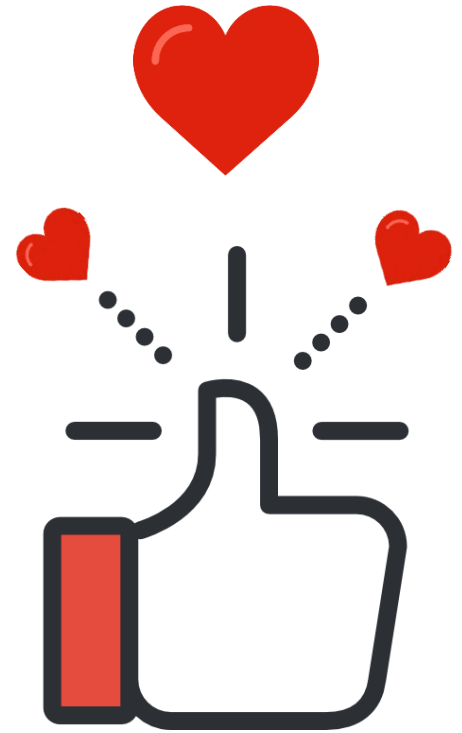
**Just Dial**

[https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4\\_BZDET](https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET)



**Sulekha**

<https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad>



# Connect With Me



Akash Padhiyar  
#AkashSir

[www.akashsir.com](http://www.akashsir.com)

[www.akashtechlabs.com](http://www.akashtechlabs.com)

[www.akashpadhiyar.com](http://www.akashpadhiyar.com)

[www.apptutorials.com](http://www.apptutorials.com)

## # Social Info



Akash.padhiyar



Akashpadhiyar



Akash\_padhiyar



+91 99786-21654



#Akashpadhiyar

#apptutorials