



# ► Software Structure

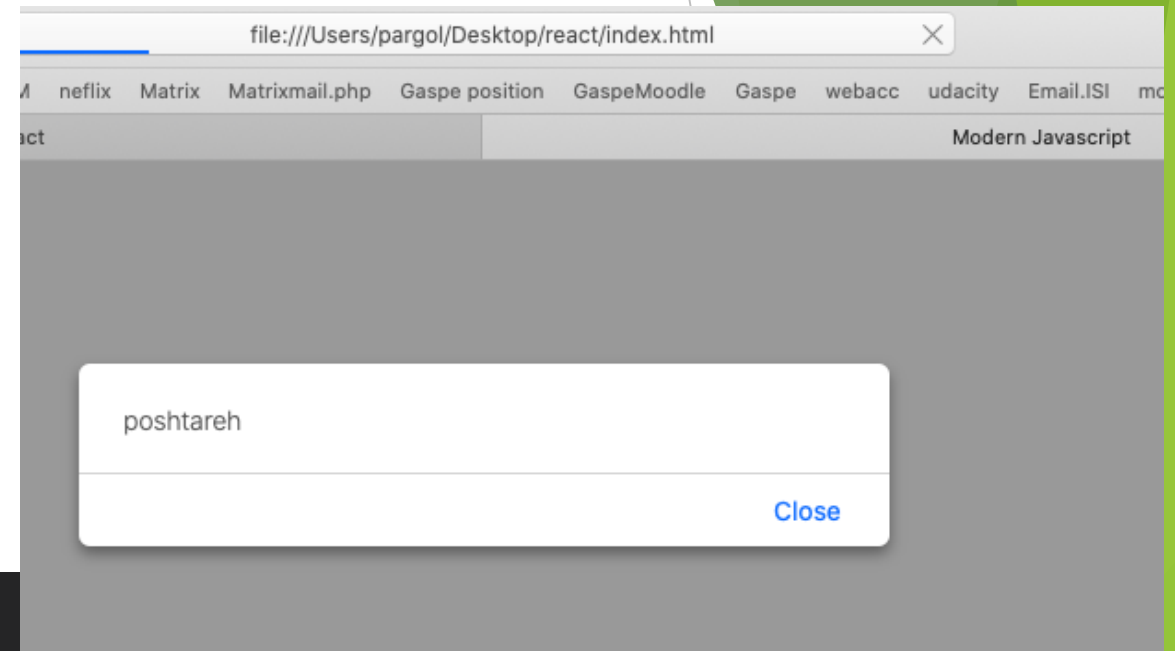
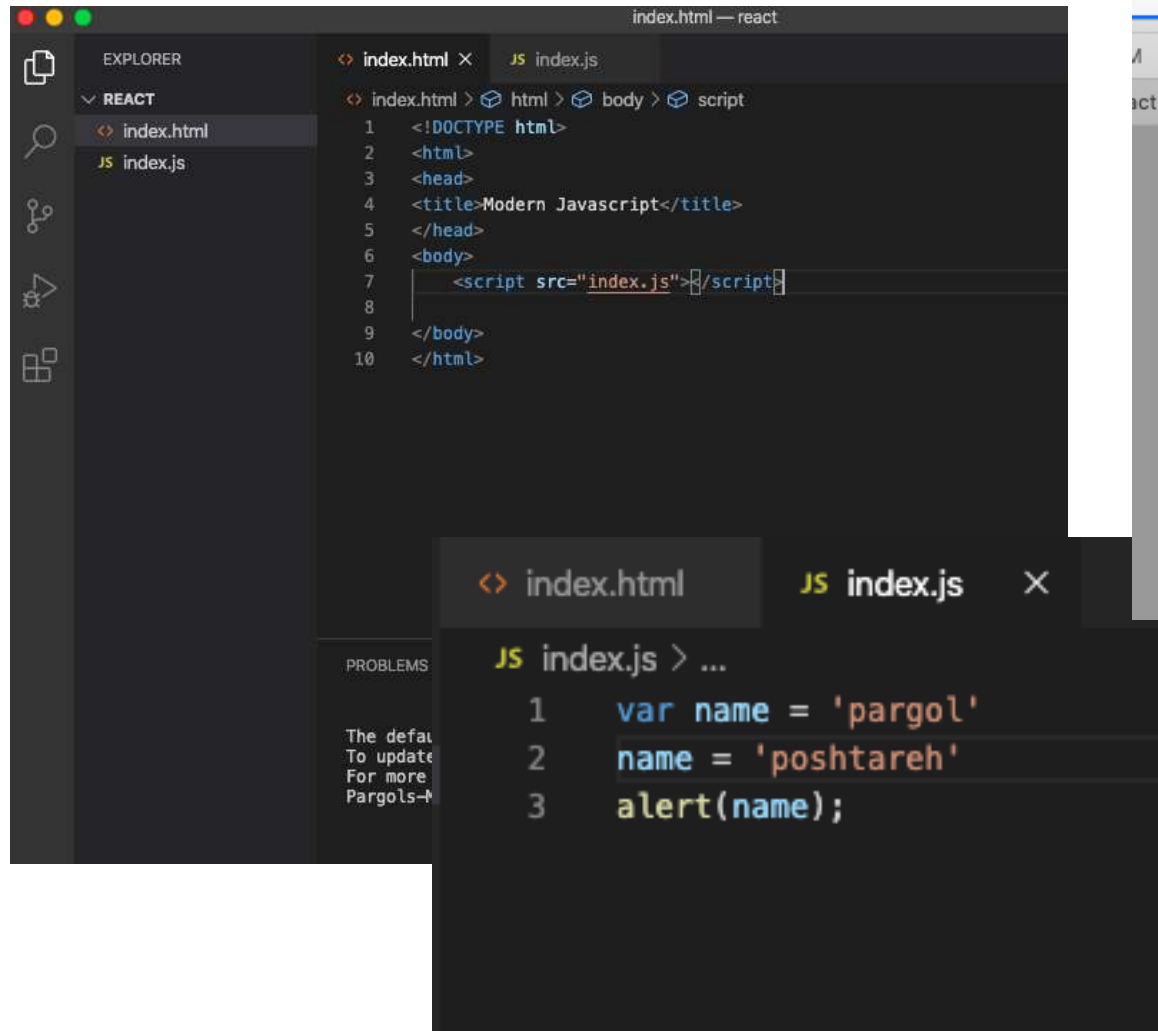
By Pargol Poshtareh

Lecture 3

## *Outline*

- Modern JavaScript
- What is React
- Components
- Props/State
- ES6

# Modern JavaScript



## *What is React?*

- Open source library for build user interfaces
- Not framework
- Focus On UI

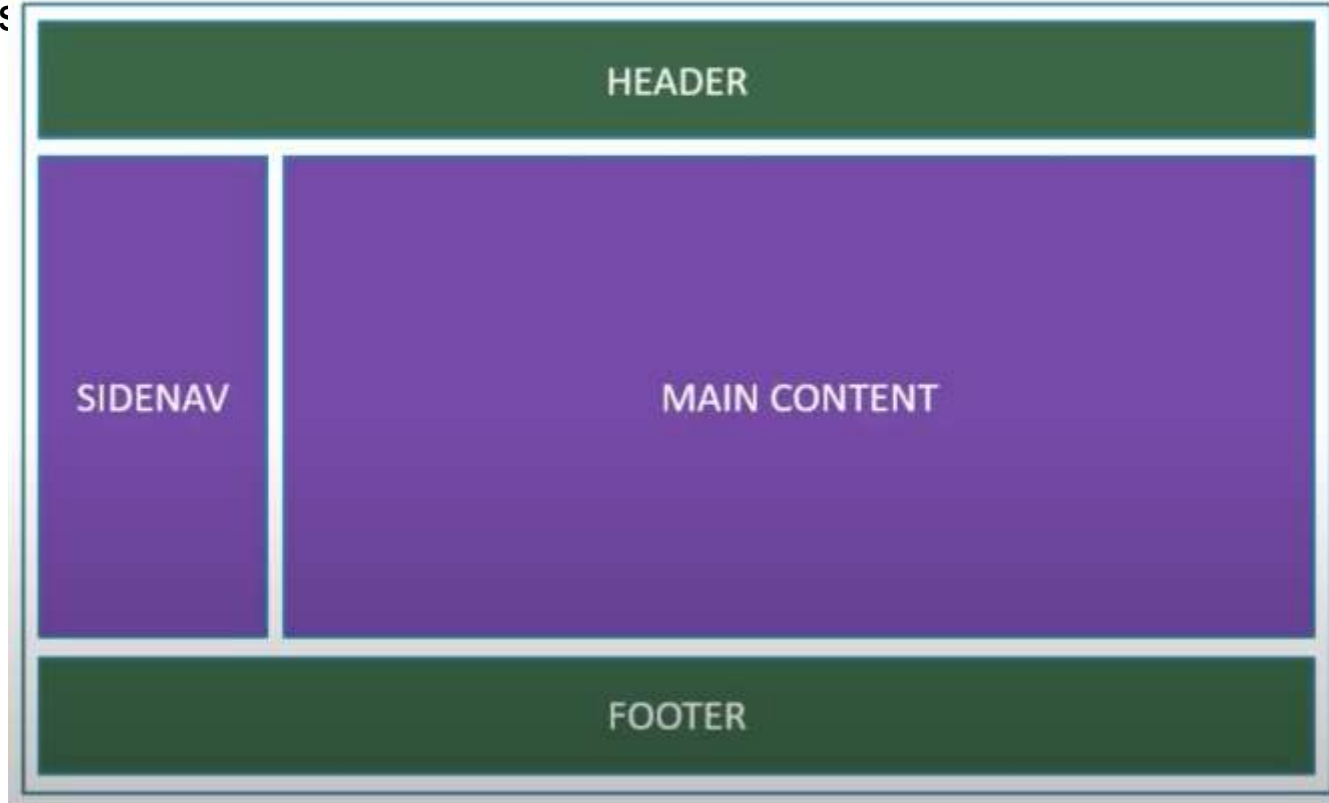
## *Why React?*

- Created and maintained by facebook
- More than 100k starts on Github
- Huge community

## ***React ? Component based Architecture***

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”.

React has

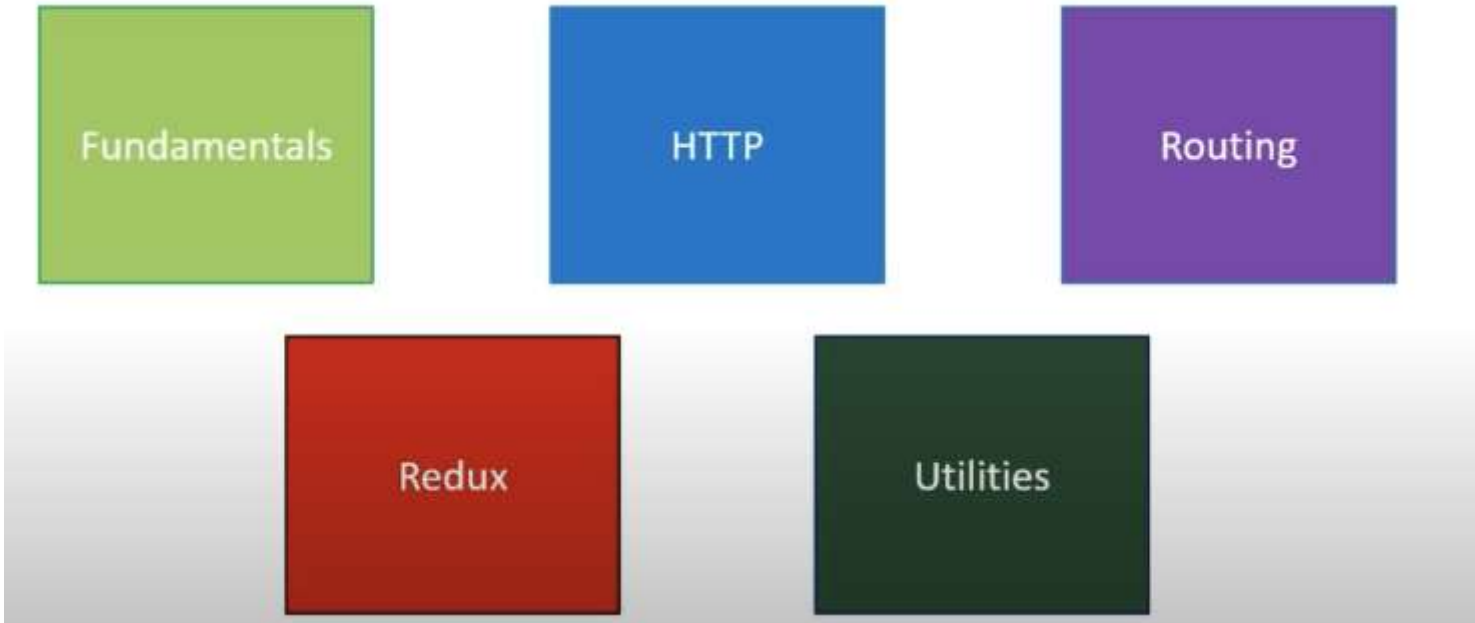


Component

## ***Prerequisites***

- HTML,CSS AND JavaScript fundamentals

## ***Channel Content – React***

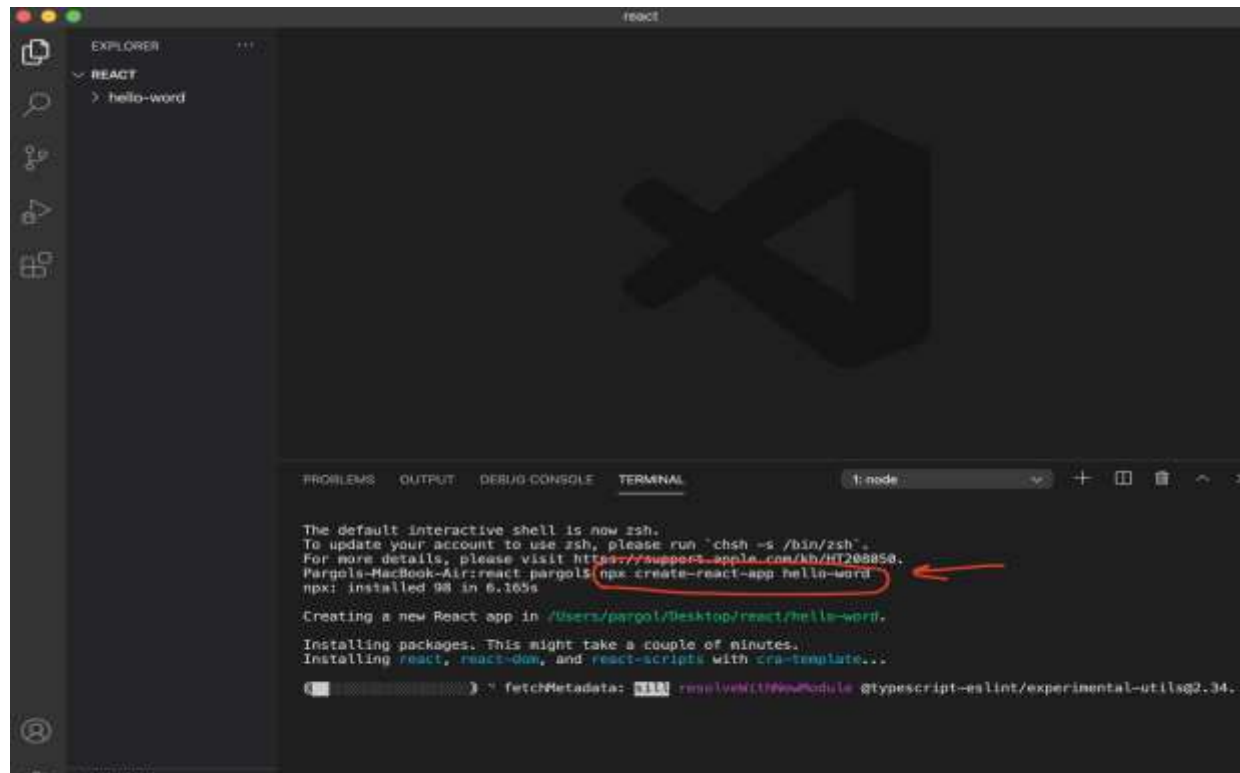


## Hello World :

Download Node and text editor or update it

Download Create React App (command line interface tools ,  
Run react without any configuration)

Command : `npx create-react-app nameOfProject`



The screenshot shows a Visual Studio Code editor window with a terminal at the bottom. The Explorer sidebar on the left shows a project named 'REACT' with a subdirectory 'hello-word'. The terminal window is titled 't: node' and contains the following text:

```
The default interactive shell is now zsh.  
To update your account to use zsh, please run "chsh -s /bin/zsh".  
For more details, please visit https://support.apple.com/kb/HT208858.  
Pargols-MacBook-Air:react pargols$ npx create-react-app hello-word  
npx: installed 98 in 6.165s  
  
Creating a new React app in /Users/pargol/Desktop/react/hello-word.  
Installing packages. This might take a couple of minutes.  
Installing react, react-dom, and react-scripts with cra-template...  
[Progress bar] fetchMetadata: 11.1 resolveWithFfiModule @typescript-eslint/experimental-utils@2.34.1
```

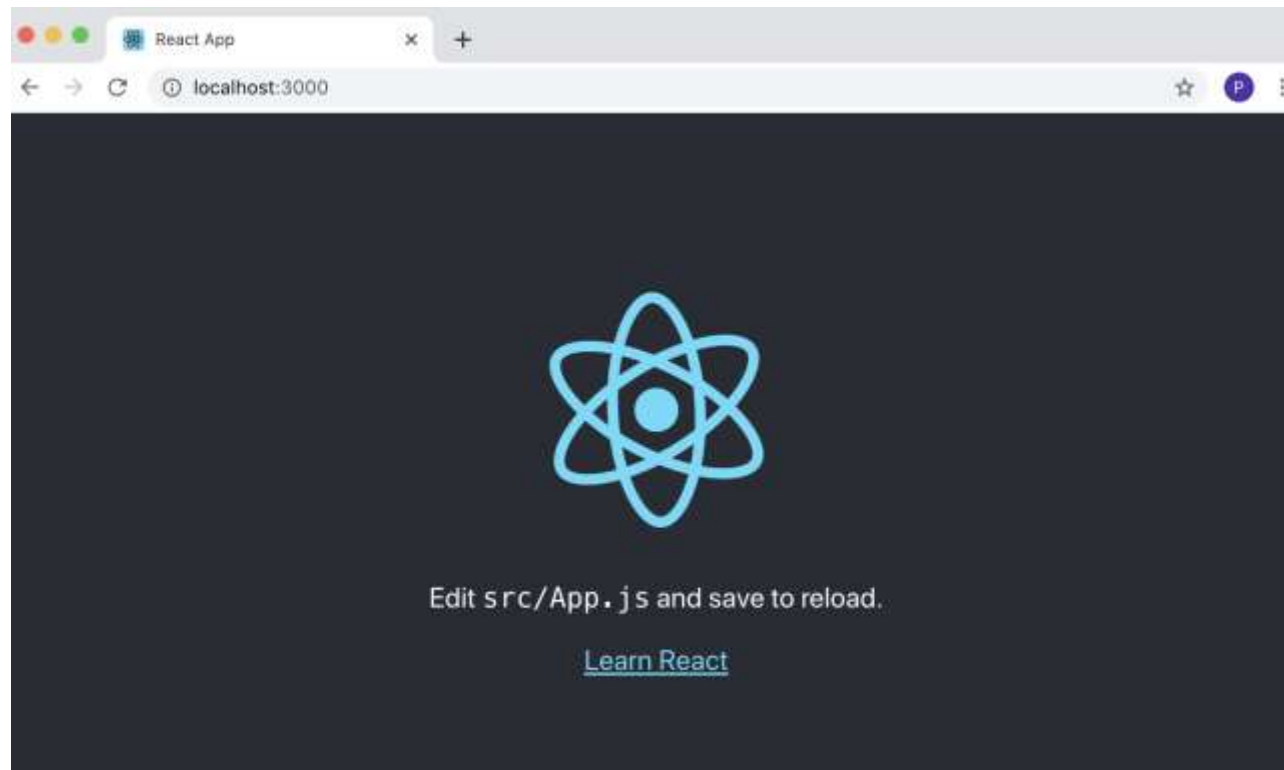
A red arrow points to the command `npx create-react-app hello-word` in the terminal.

Now Navigate inside folder :

```
cd hello-word/
```

Now Run the app :

```
npm start
```





App.js — react

EXPLORER

- REACT
  - hello-word
    - node\_modules
    - public
    - src
      - App.css
      - App.js ←
      - App.test.js
      - index.css
      - index.js
      - logo.svg
      - serviceWorker.js
      - setupTests.js
    - .gitignore
    - package.json
    - package-lock.json
    - README.md

```
1 import React from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4
5 function App() {
6   return (
7     <div className="App">
8       <header className="App-header">
9         <img src={logo} className="App-logo" alt="logo" />
10        <p>
11          Hello world. ←
12        </p>
13        <a
14          className="App-link"
15          href="https://reactjs.org"
16          target="_blank"
17          rel="noopener noreferrer"
18        >
19          Learn React
20        </a>
21      </div>
22    );
23  }
24}
```

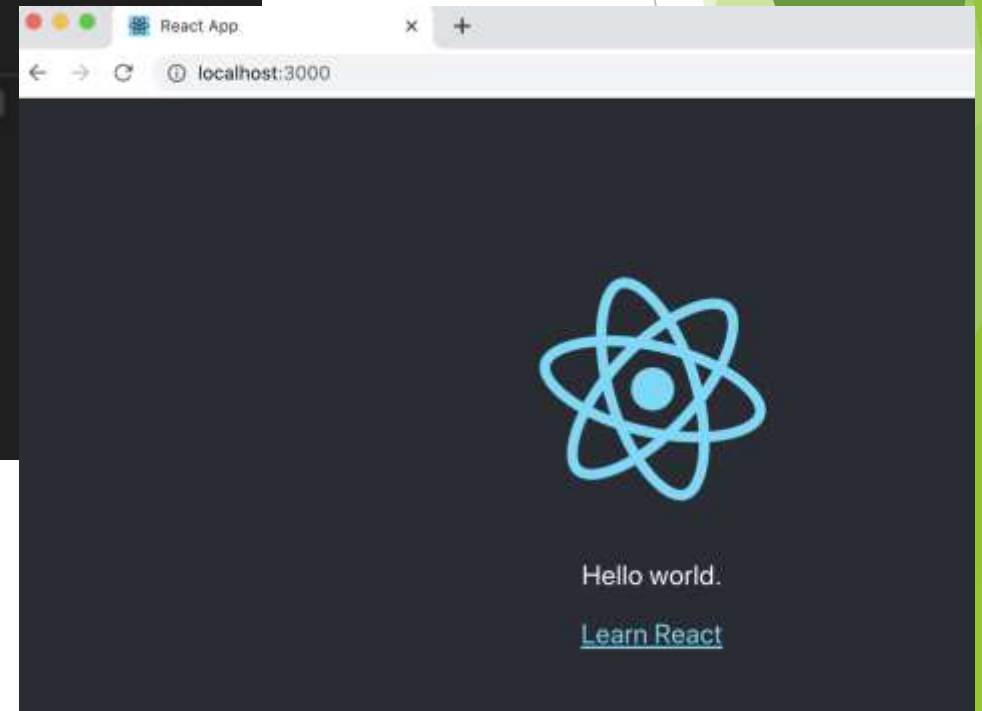
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: node

Compiled successfully!

You can now view **hello-word** in the browser.

Local: http://localhost:3000  
On Your Network: http://192.168.0.152:3000

Note that the development build is not optimized.  
To create a production build, use `npm run build`.



# Create-react-app

---

## **npx**

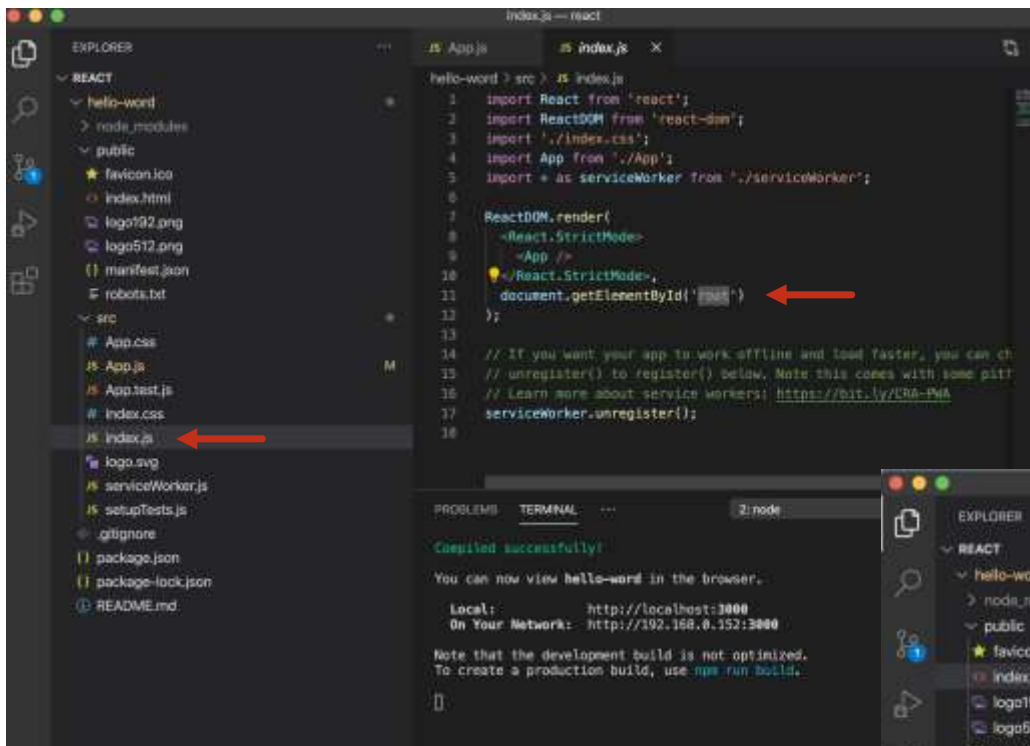
```
npx create-react-app <project_name>
```

npm package runner

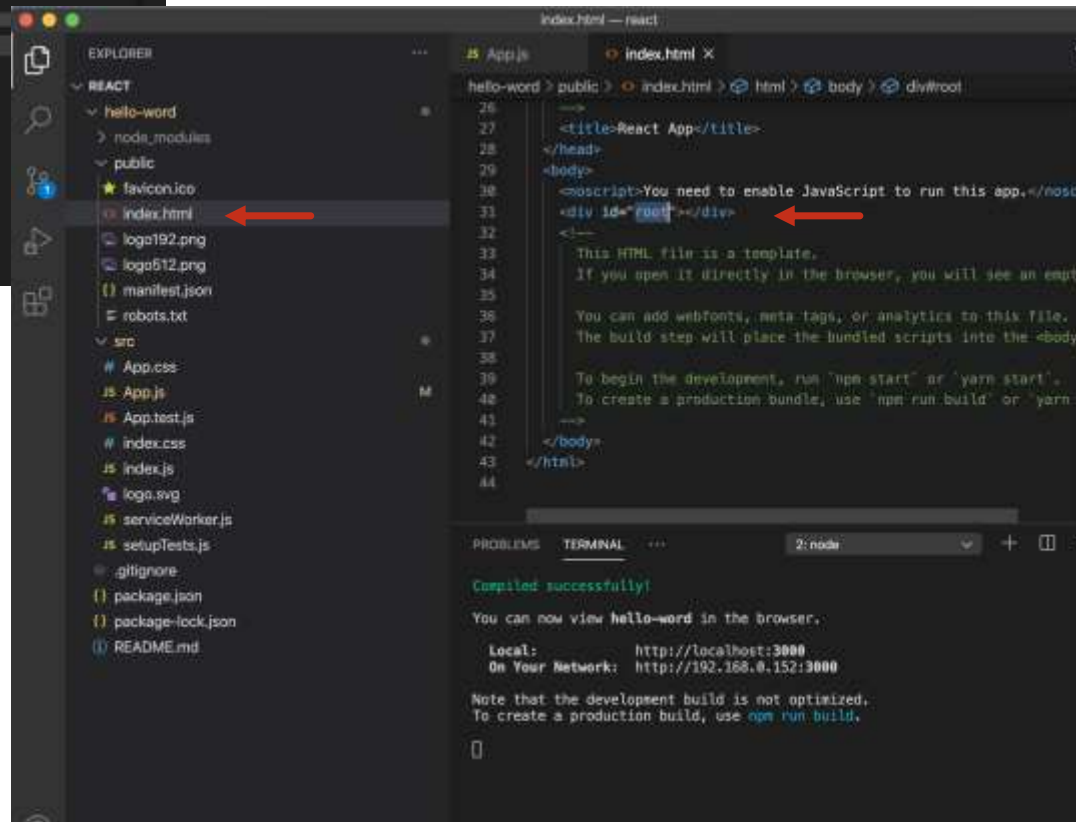
## **npm**

```
npm install create-react-app -g
```

```
create-react-app<project_name>
```

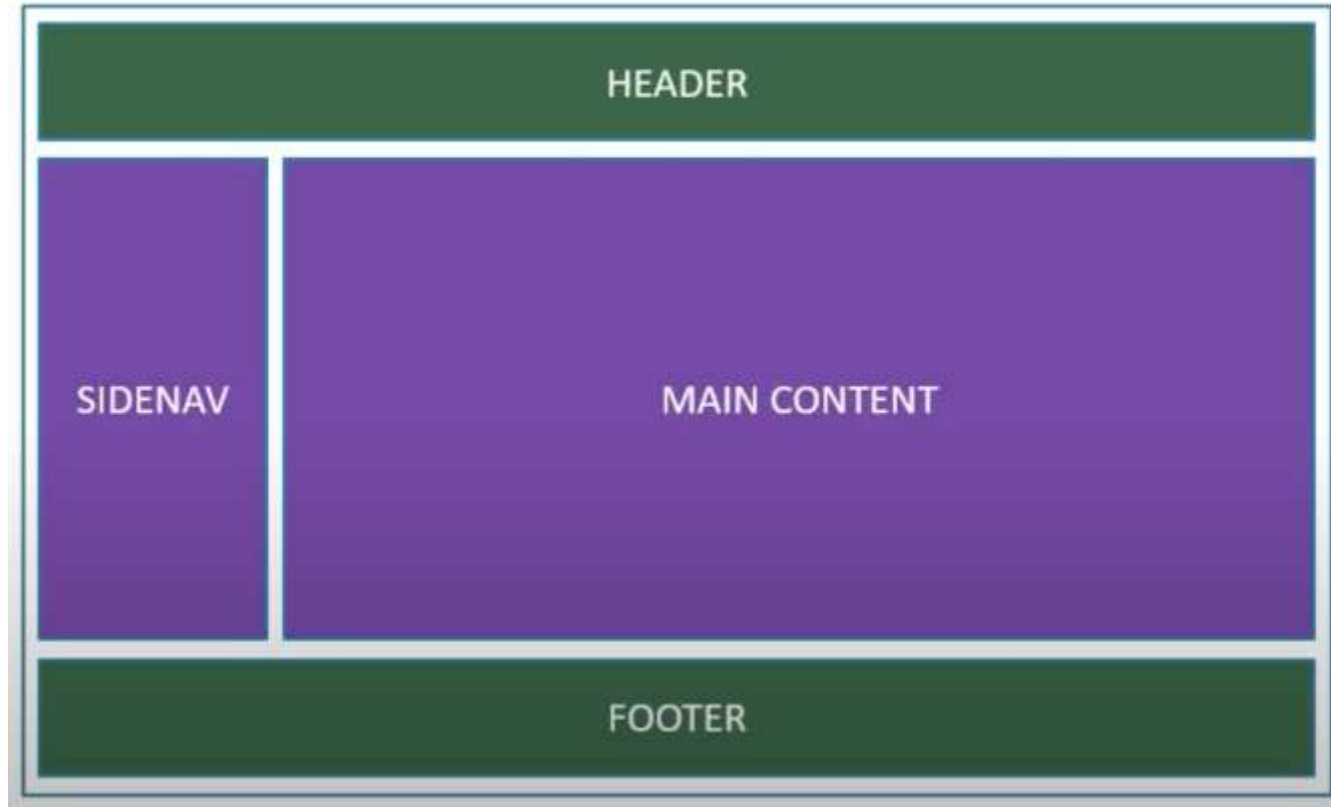


Same root which control by React



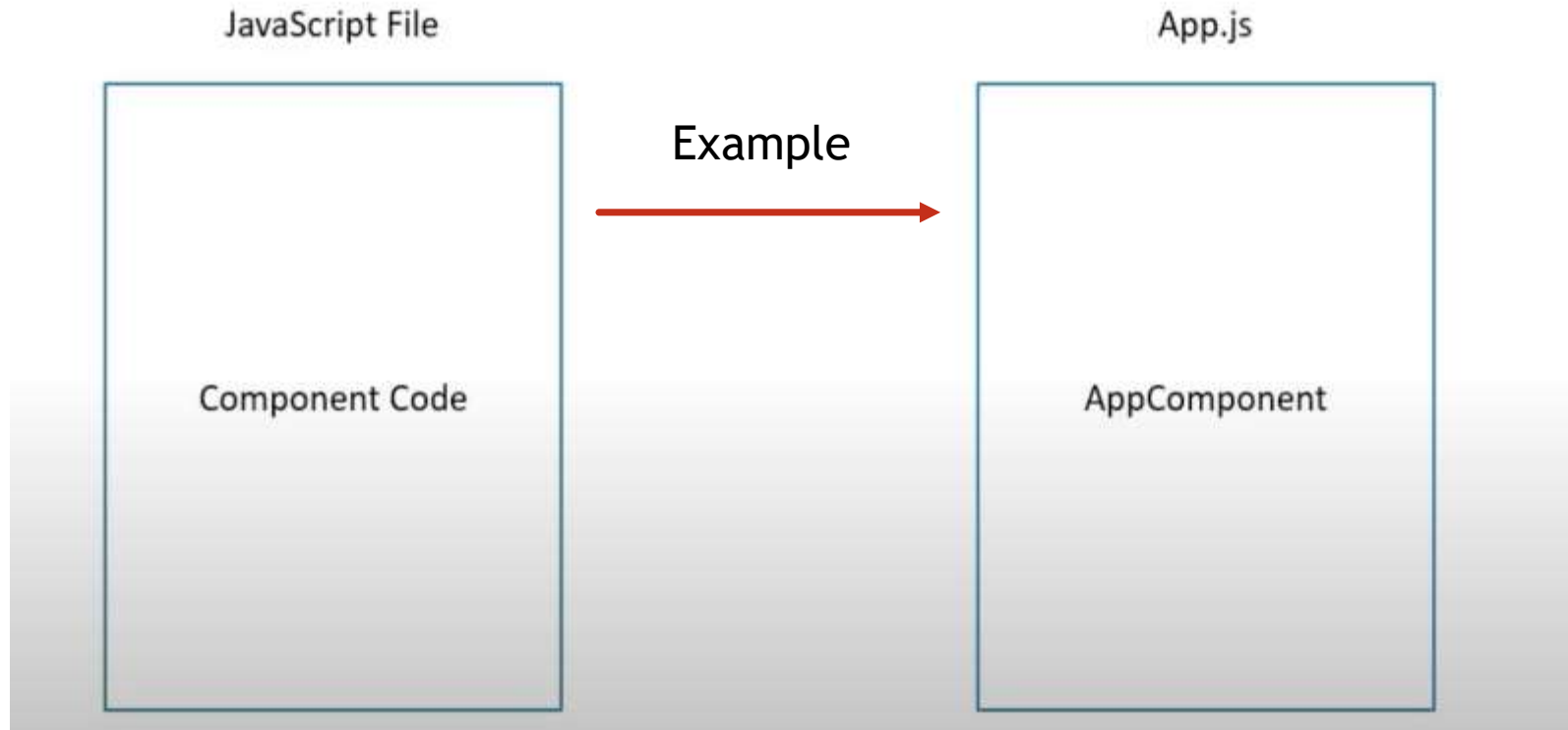
## ***Component :***

Root(App)  
Component



Component are usable again  
Component can contain other component

## ***Component in Code :***



## Type Component:

### Stateless Functional Component

JavaScript Functions

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

Return HTML which define UI

### Stateful Class Component

Class extending Component class

Render method returning HTML

```
class Welcome extends React.Component {  
  render() {  
    return <h1>Hello, {this.props.name}</h1>;  
  }  
}
```

Must content Render method

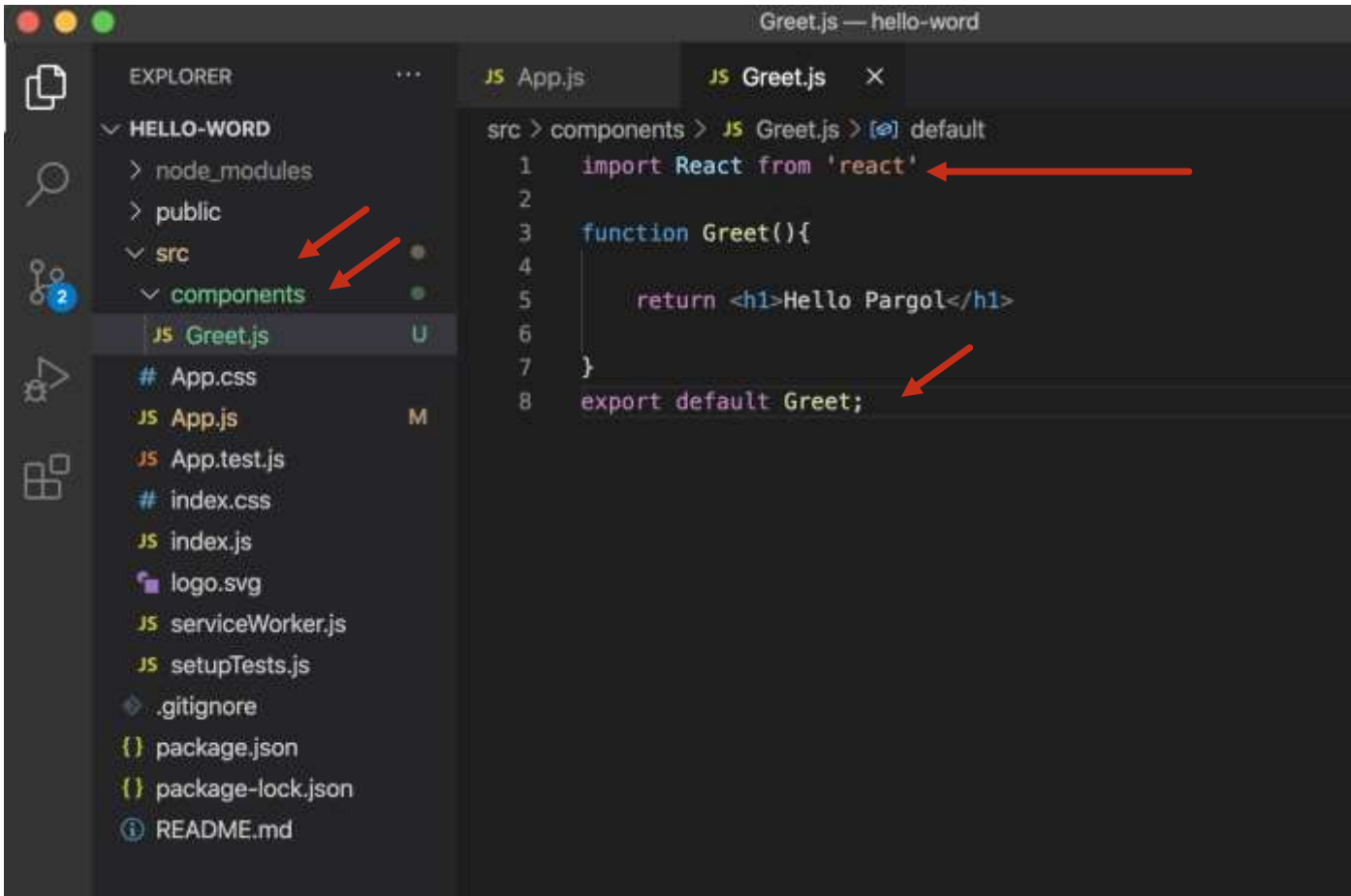
## *Functional Components:*



## Example

Create the folder named component in src.

Create a file inside component folder name Greet.js

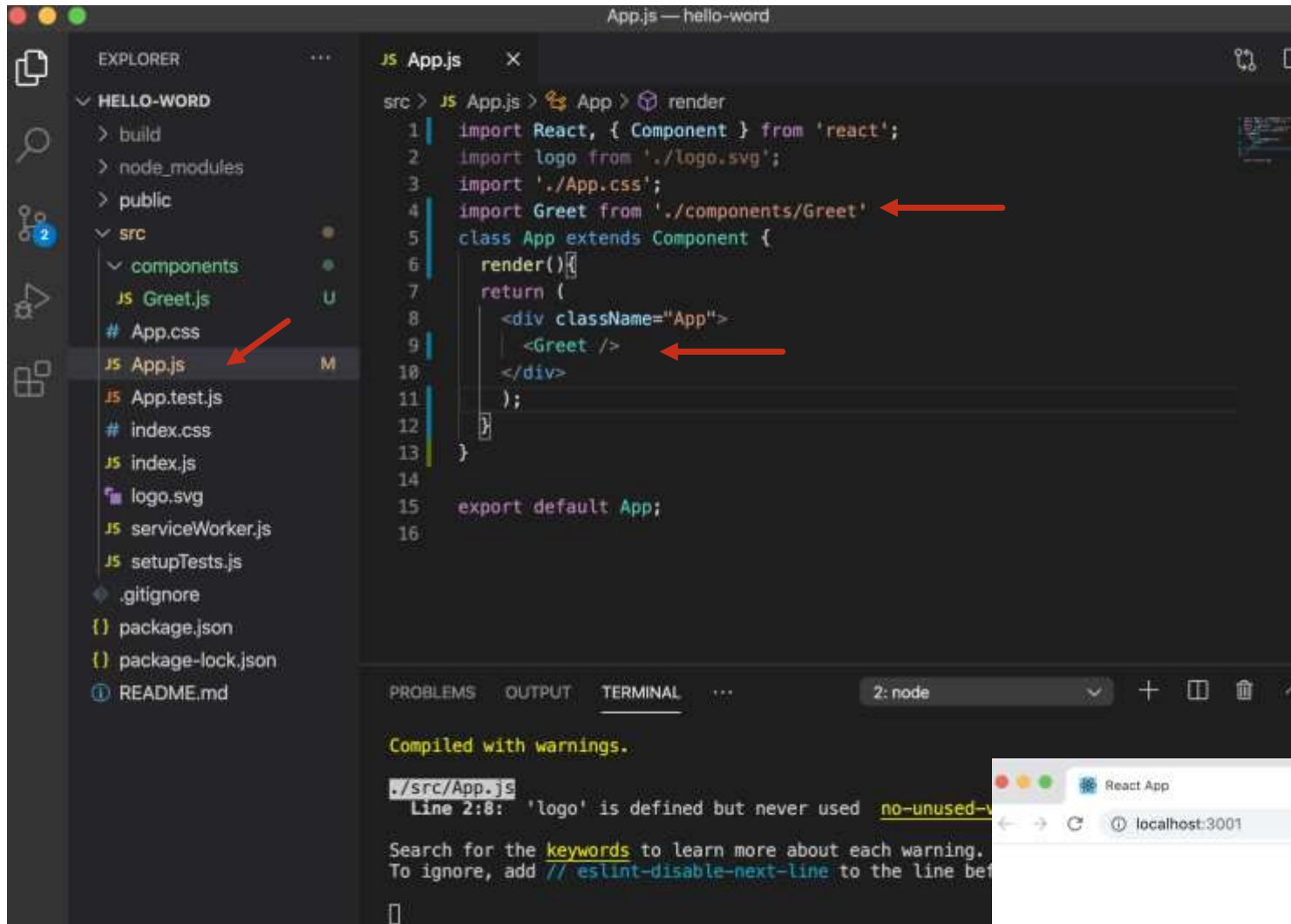


The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure for 'HELLO-WORD'. The 'src' folder is expanded, showing a 'components' subfolder. Two red arrows point to the 'components' folder and the 'Greet.js' file within it. The main editor area shows the code for 'Greet.js'. The code is as follows:

```
src > components > JS Greet.js > [?] default
1  import React from 'react'
2
3  function Greet(){
4
5      return <h1>Hello Pargol</h1>
6
7  }
8  export default Greet;
```

Two red arrows highlight specific parts of the code: one points to the `import React from 'react'` statement on line 1, and the other points to the `export default Greet;` statement on line 8.





The image shows a VS Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'src' directory containing 'components', 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', 'logo.svg', 'serviceWorker.js', and 'setupTests.js'. The 'App.js' file is selected. The code editor shows the following code:

```
src > JS App.js > App > render
1 | import React, { Component } from 'react';
2 | import logo from './logo.svg';
3 | import './App.css';
4 | import Greet from './components/Greet'
5 | class App extends Component {
6 |   render() {
7 |     return (
8 |       <div className="App">
9 |         <Greet />
10 |       </div>
11 |     );
12 |   }
13 | }
14 |
15 | export default App;
16 |
```

Two red arrows point to the import statement on line 4 and the Greet component on line 9. The terminal at the bottom shows the following output:

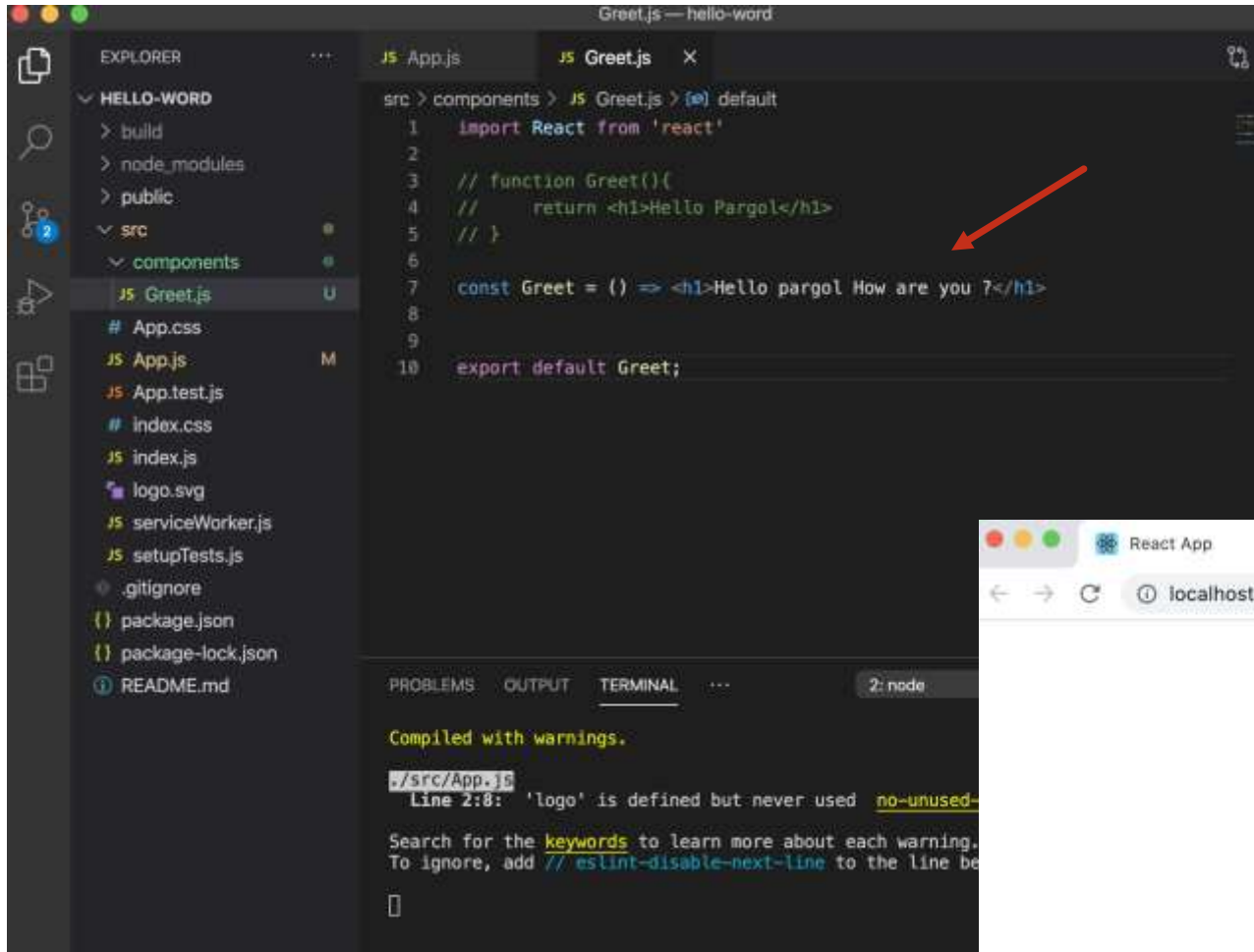
```
Compiled with warnings.

./src/App.js
Line 2:8: 'logo' is defined but never used  no-unused-va

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before
```



OR :



The screenshot shows the VS Code editor with the file explorer on the left. The file explorer shows a project structure with a 'src' directory containing 'components' and 'Greet.js'. The 'Greet.js' file is selected, and its content is displayed in the editor. The code defines an arrow function 'Greet' that returns a JSX element. A red arrow points to the arrow function syntax. The terminal at the bottom shows the output of the command 'npm run build', which is 'Compiled with warnings.' and a warning about an unused variable 'logo'.

```
src > components > JS Greet.js > default
1  import React from 'react'
2
3  // function Greet(){
4  //   return <h1>Hello Pargol</h1>
5  // }
6
7  const Greet = () => <h1>Hello pargol How are you ?</h1>
8
9
10 export default Greet;
```

PROBLEMS OUTPUT TERMINAL ... 2: node

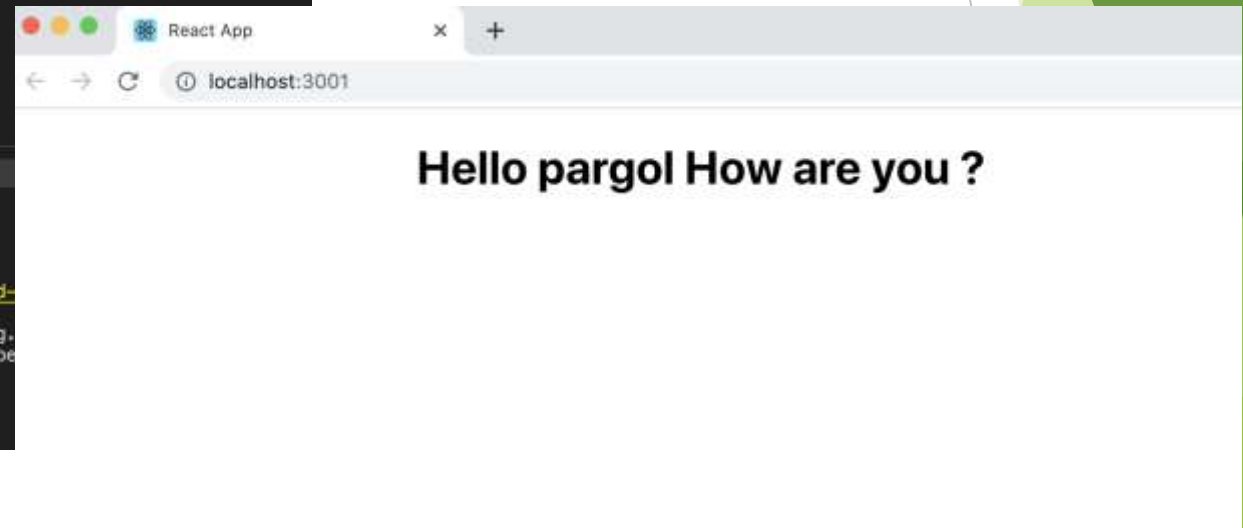
Compiled with warnings.

./src/App.js  
Line 2:8: 'logo' is defined but never used no-unused-

Search for the keywords to learn more about each warning.  
To ignore, add // eslint-disable-next-line to the line be

Using Arrow Function

Please review :  
[Here](#)



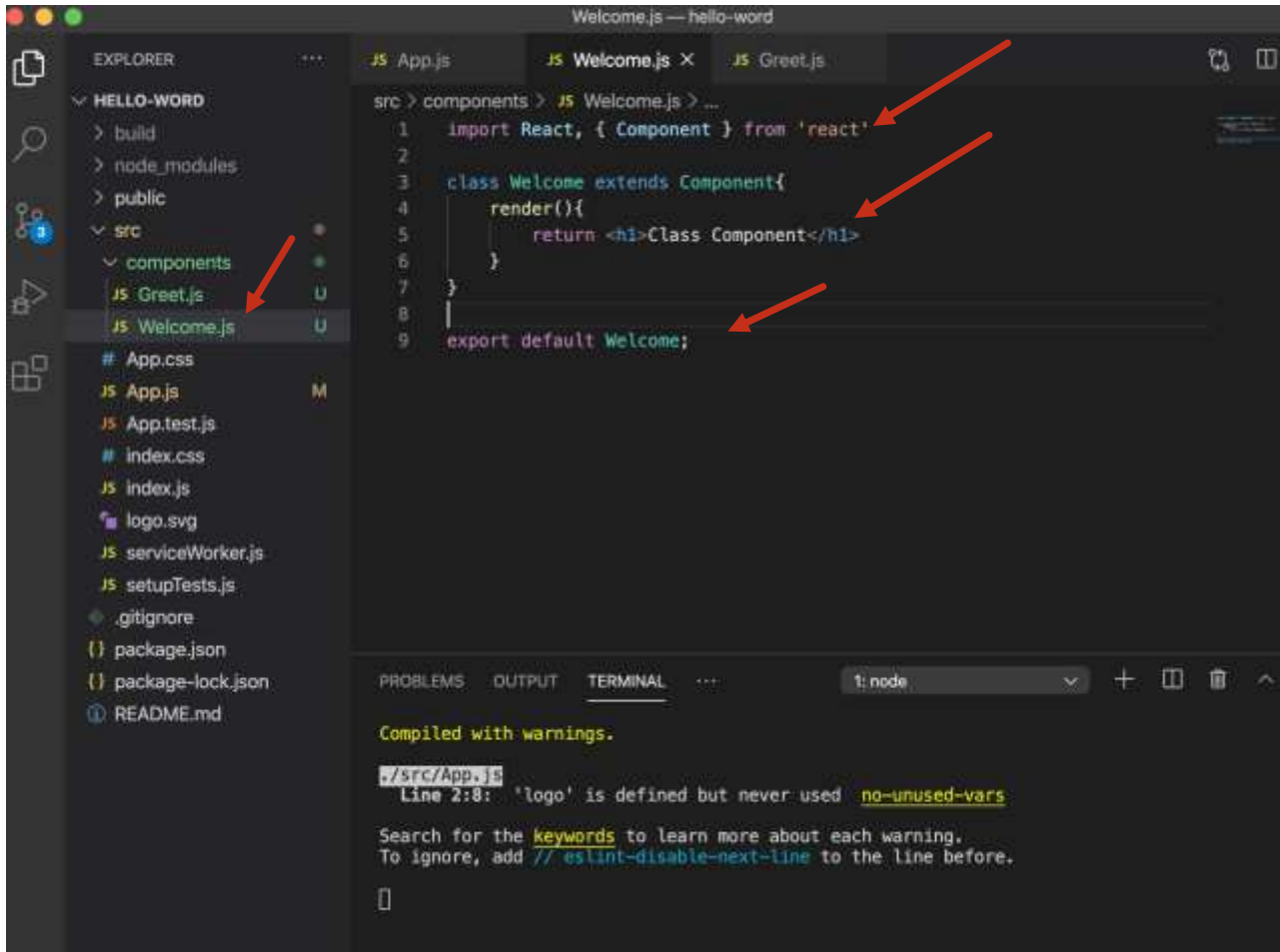
## ***Class Components:***



## Example

Create a file inside component folder name Welcome.js

For Class Components we need 2 imports



The screenshot shows a VS Code editor with a project named 'hello-word'. The Explorer sidebar on the left shows the file structure, with the 'components' folder expanded. A red arrow points to 'JS Welcome.js' in the file list. The main editor shows the code for 'Welcome.js' with three red arrows highlighting key parts: the import statement, the class definition, and the default export.

```
src > components > JS Welcome.js > ...  
1  import React, { Component } from 'react'  
2  
3  class Welcome extends Component {  
4      render() {  
5          return <h1>Class Component</h1>  
6      }  
7  }  
8  
9  export default Welcome;
```

The bottom panel shows the terminal output:

```
Compiled with warnings.  
  
./src/App.js  
Line 2:8: 'logo' is defined but never used  no-unused-vars  
  
Search for the keywords to learn more about each warning.  
To ignore, add // eslint-disable-next-line to the line before.  
  
[]
```

```
App.js — hello-word
src > JS App.js > JS Welcome.js > JS Greet.js
1 import React, { Component } from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4 import Greet from './components/Greet'
5 import Welcome from './components/Welcome'
6 class App extends Component {
7   render(){
8     return (
9       <div className="App">
10         <Greet />
11         <Welcome />
12       </div>
13     );
14   }
15 }
16
17 export default App;
18
```

Hello pargol How are you ?

Class Component

## Functional vs Class Components

### Functional

- Simple functions
- Use Func components as much as possible
- Absence of 'this' keyword
- Solution without using state
- Mainly responsible for the UI
- ~~Stateless~~/ Dumb/ Presentational

### Class

- More feature rich
- Maintain their own private data - state
- Complex UI logic
- Provide lifecycle hooks
- Stateful/ Smart/ Container

Stateless : Hooks are a new features that let you use state and other react features without writing a class  
**DON'T NEED TO USE IF YOU DON'T WANT TO**

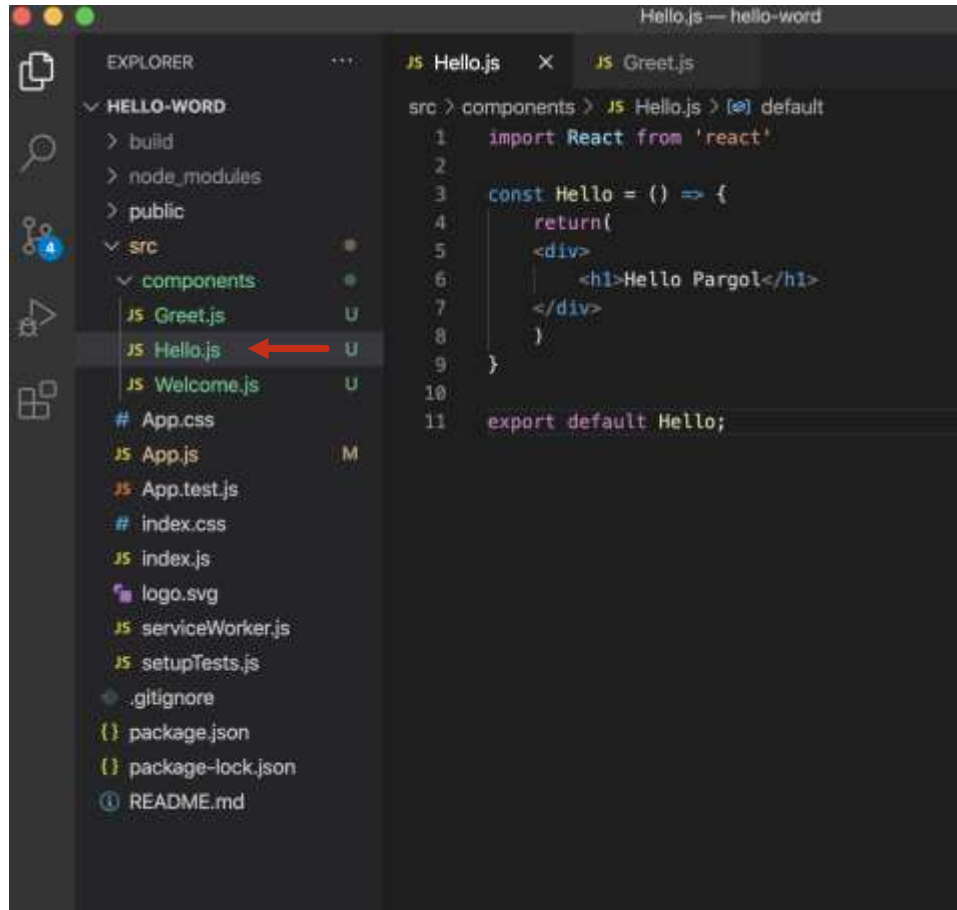
# JSX

- JavaScript XML - extension to the JavaScript language syntax
- Write XML - like code for elements and components
- JSX tags have a tag name, attributes, and children
- JSX is not a necessity to write React application
- JSX makes your react code simpler and elegant
- JSX ultimately trans piles to pure JavaScript which is understood by the browsers



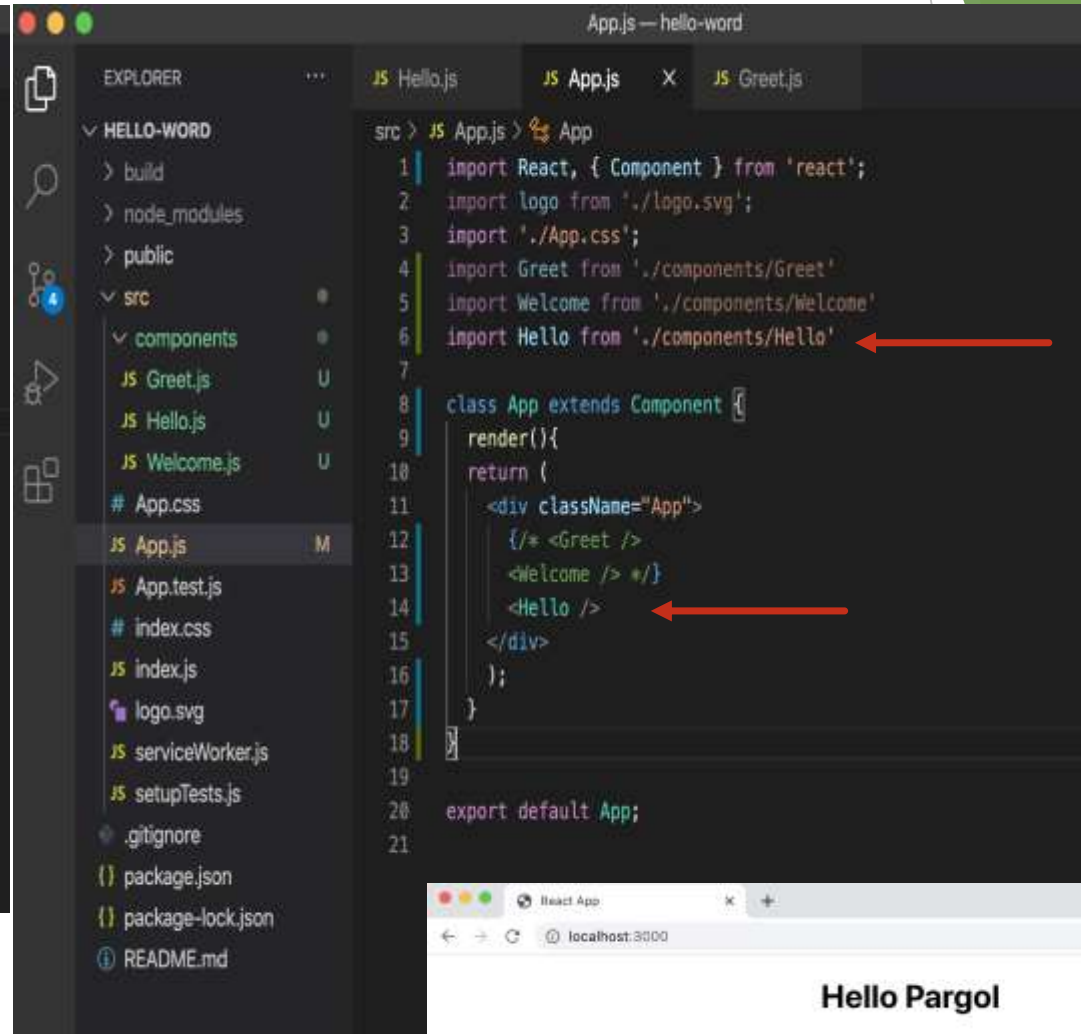
Let's see:

Create new components named Hello.js



The image shows a VS Code editor window titled 'Hello.js — hello-word'. The Explorer sidebar on the left shows a project structure with a 'src' directory containing a 'components' subdirectory. Inside 'components', there are files 'Greet.js', 'Hello.js', and 'Welcome.js'. 'Hello.js' is selected and highlighted with a red arrow. The main editor area shows the code for 'Hello.js' with the following content:

```
1 import React from 'react'
2
3 const Hello = () => {
4   return(
5     <div>
6       <h1>Hello Pargol</h1>
7     </div>
8   )
9 }
10
11 export default Hello;
```



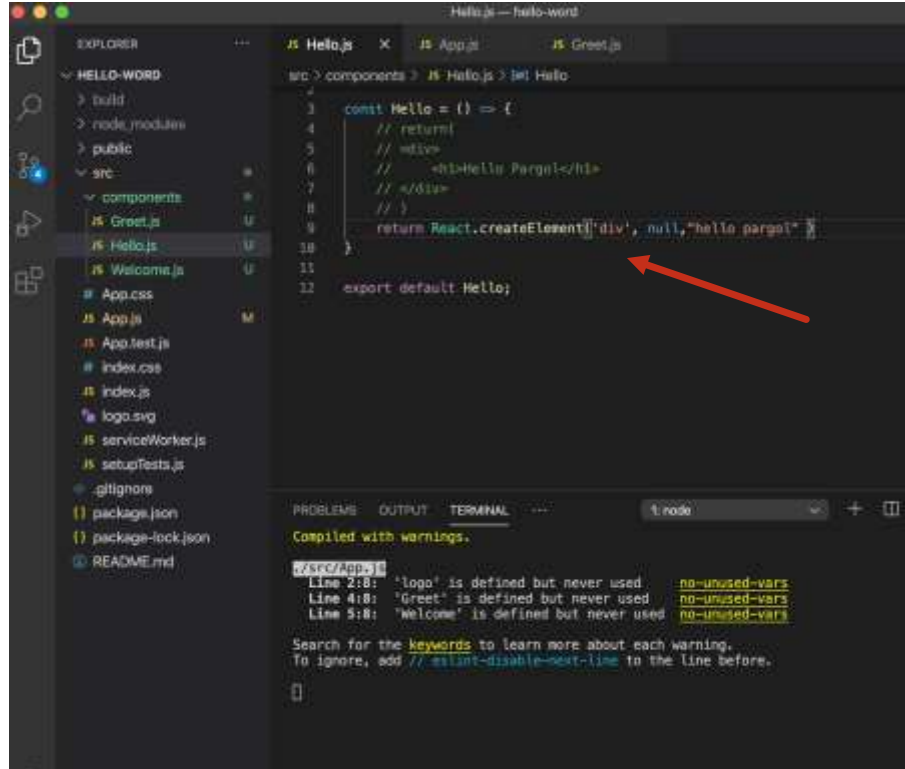
The image shows a VS Code editor window titled 'App.js — hello-word'. The Explorer sidebar on the left shows the same project structure as the previous image, but now 'App.js' is selected and highlighted with a red arrow. The main editor area shows the code for 'App.js' with the following content:

```
1 import React, { Component } from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4 import Greet from './components/Greet';
5 import Welcome from './components/Welcome';
6 import Hello from './components/Hello';
7
8 class App extends Component {
9   render(){
10     return (
11       <div className="App">
12         { /* <Greet />
13           <Welcome /> */ }
14         <Hello />
15       </div>
16     );
17   }
18 }
19
20 export default App;
```

Below the editor, a browser window titled 'React App' is shown, displaying the text 'Hello Pargol'.



## Now Check without JSX:



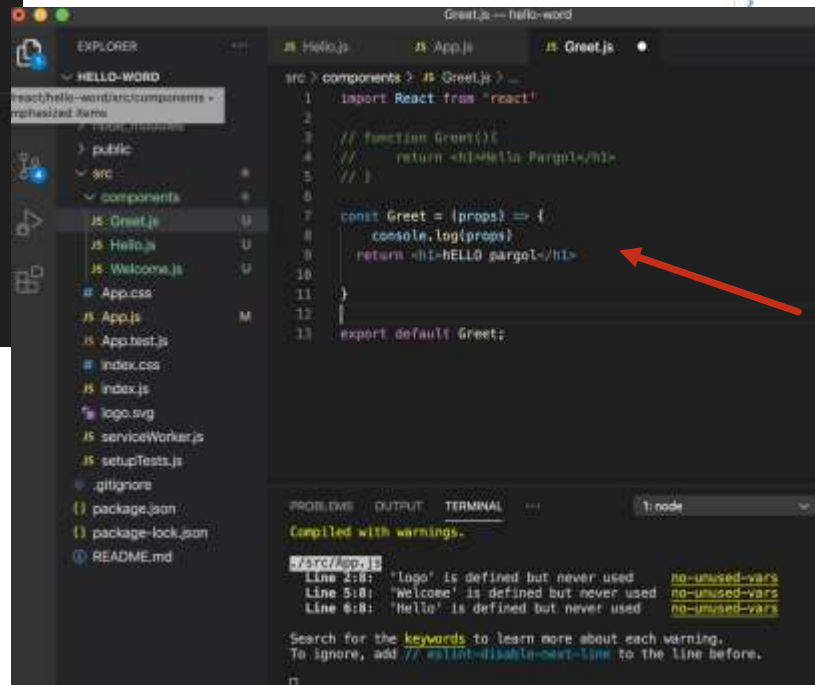
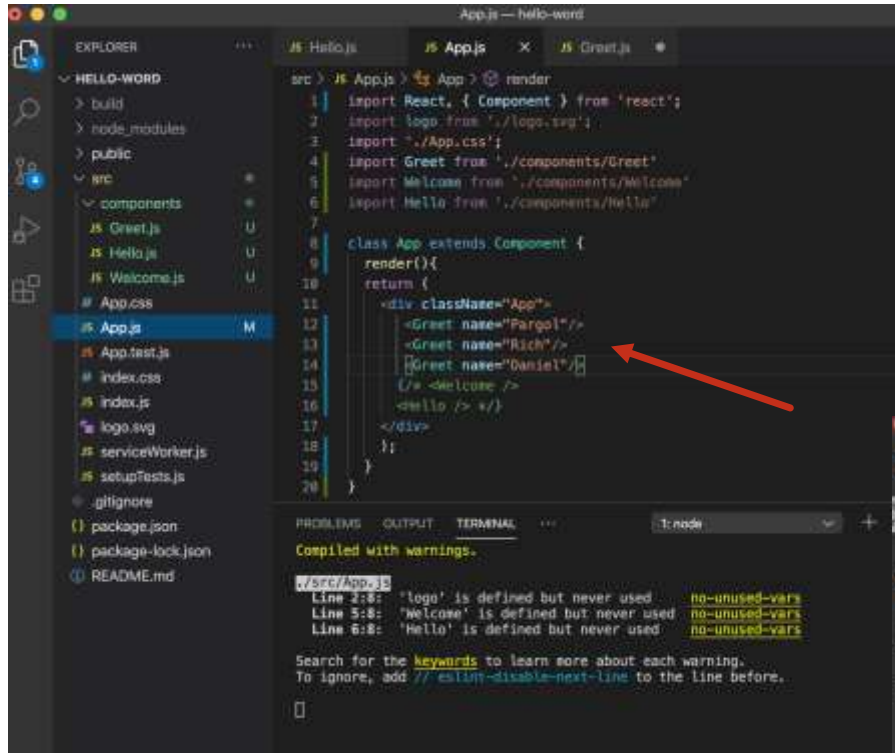
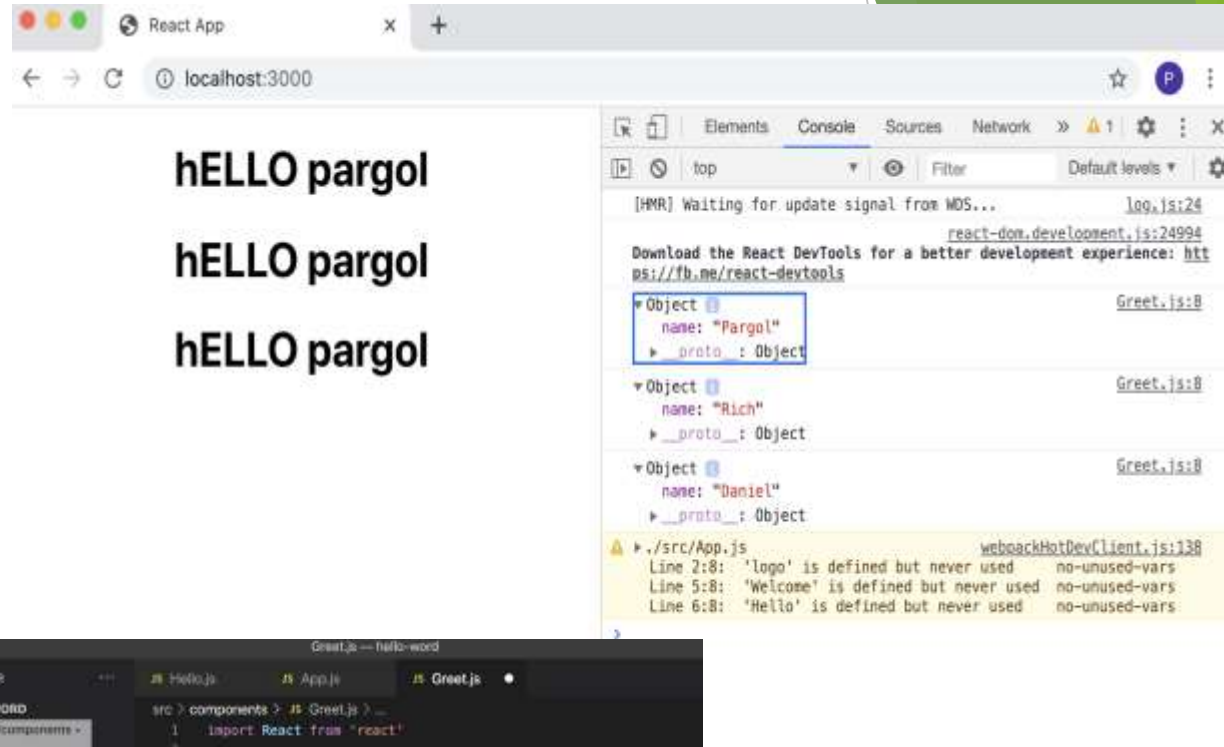
Create and return a new [React element](#) of the given type. The type argument can be either a tag name string (such as 'div' or 'span'), a [React component](#) type (a class or a function), or a [React fragment](#) type.

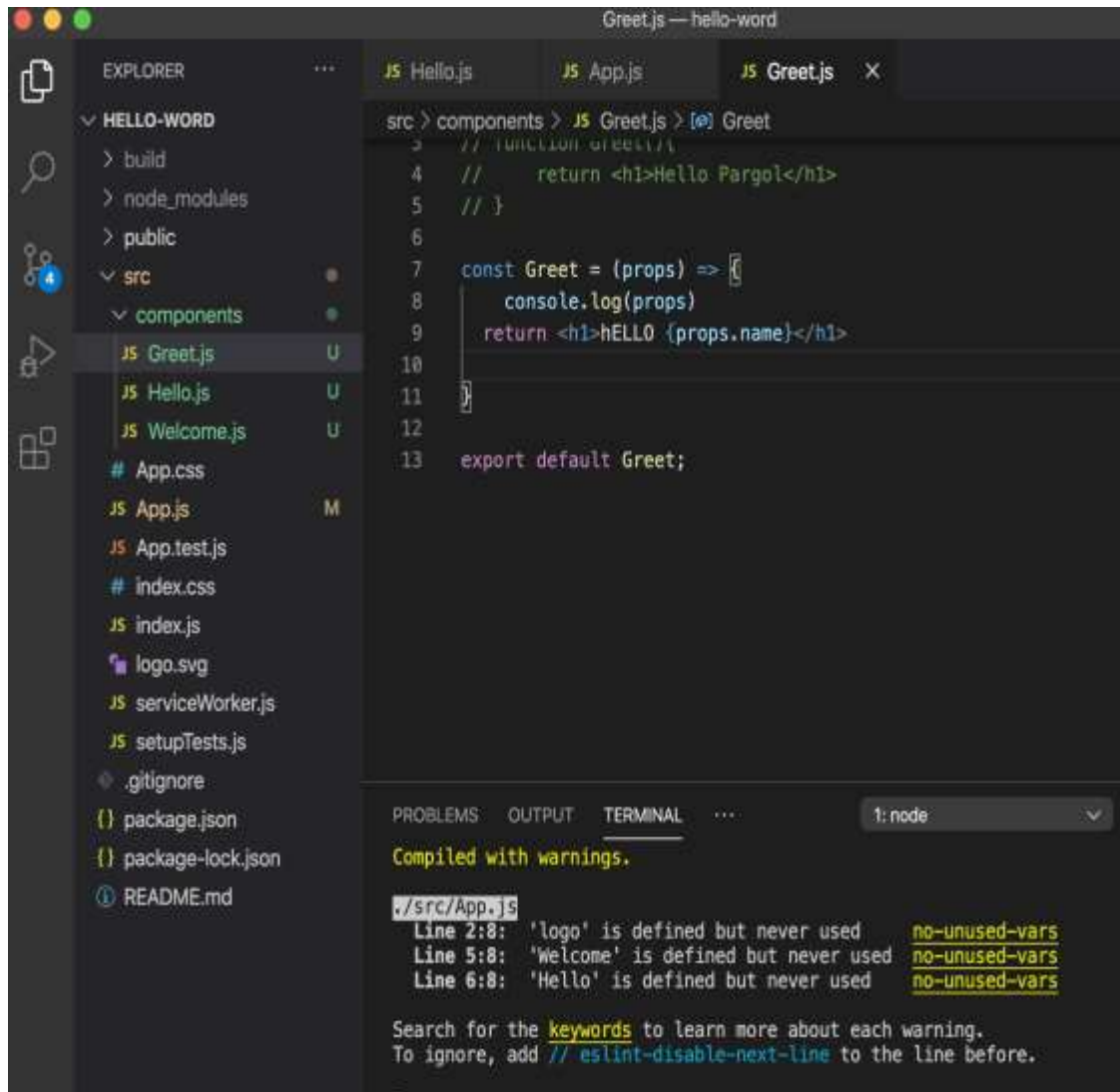
Code written with [JSX](#) will be converted to use `React.createElement()`. You will not typically invoke `React.createElement()` directly if you are using JSX. See [React Without JSX](#) to learn more.

```
React.createElement(
  type,
  [props],
  [...children]
)
```

# Props:

Make components Dynamics





The image shows a VS Code editor window with a project named 'hello-word'. The Explorer sidebar on the left shows the file structure: 'HELLO-WORD' (root) contains 'build', 'node\_modules', 'public', and 'src'. The 'src' directory contains 'components', 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', 'logo.svg', 'serviceWorker.js', 'setupTests.js', '.gitignore', 'package.json', 'package-lock.json', and 'README.md'. The 'components' directory contains 'Greet.js', 'Hello.js', and 'Welcome.js'. The 'Greet.js' file is open in the editor, showing the following code:

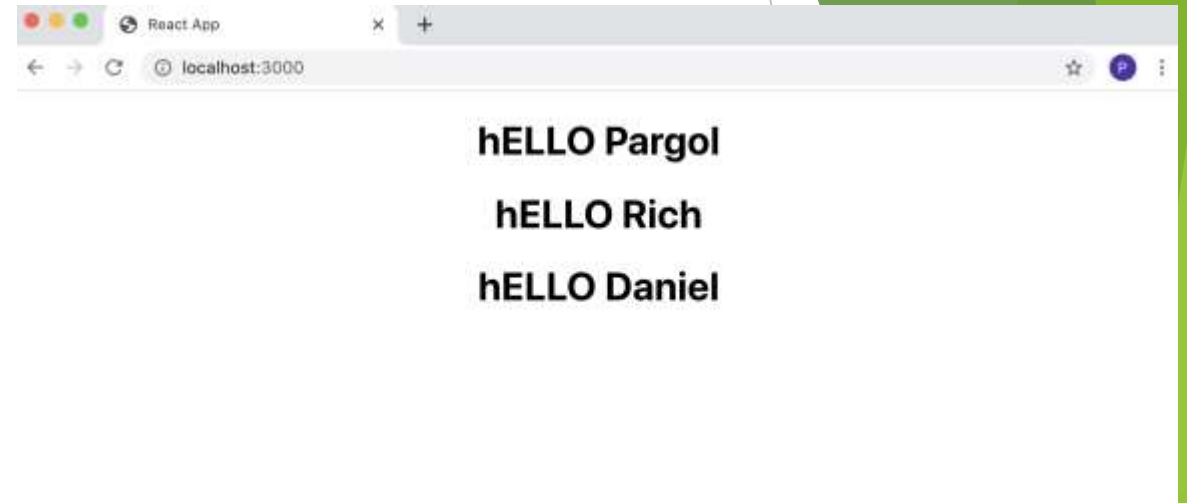
```
1 // function greet()
2 //
3 //
4 //   return <h1>Hello Pargol</h1>
5 // }
6
7 const Greet = (props) => {
8   console.log(props)
9   return <h1>hELLO {props.name}</h1>
10 }
11
12
13 export default Greet;
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
1: node
Compiled with warnings.

./src/App.js
Line 2:8: 'logo' is defined but never used  no-unused-vars
Line 5:8: 'Welcome' is defined but never used  no-unused-vars
Line 6:8: 'Hello' is defined but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

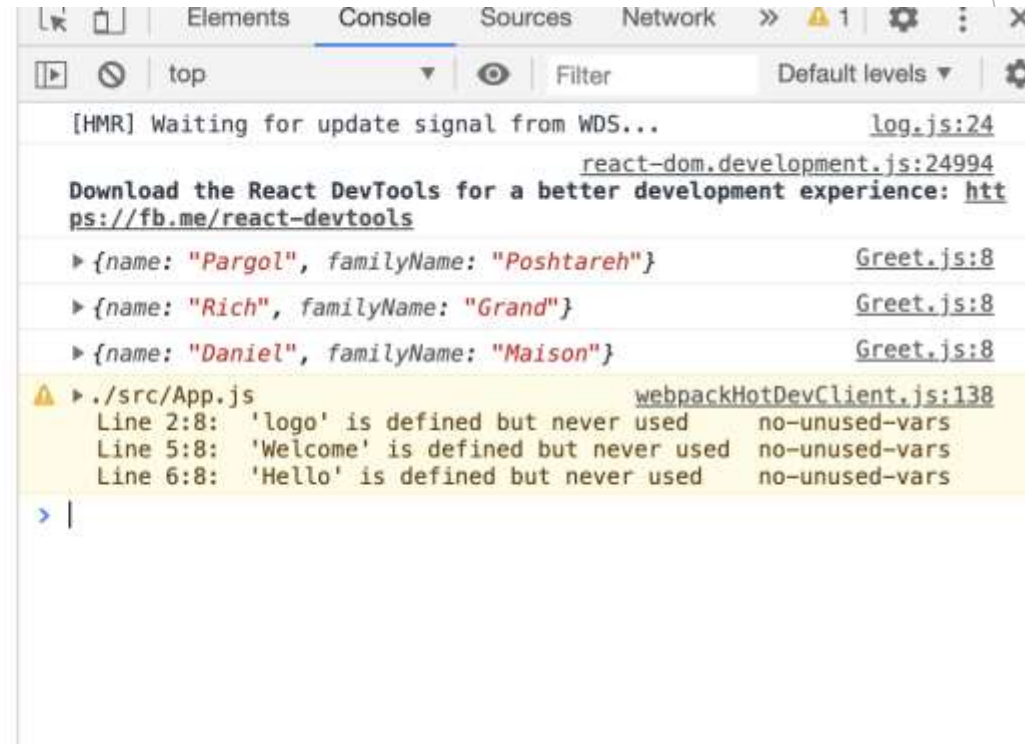


Exercise :

**hELLO Pargol Poshtareh**

**hELLO Rich Grand**

**hELLO Daniel Maison**



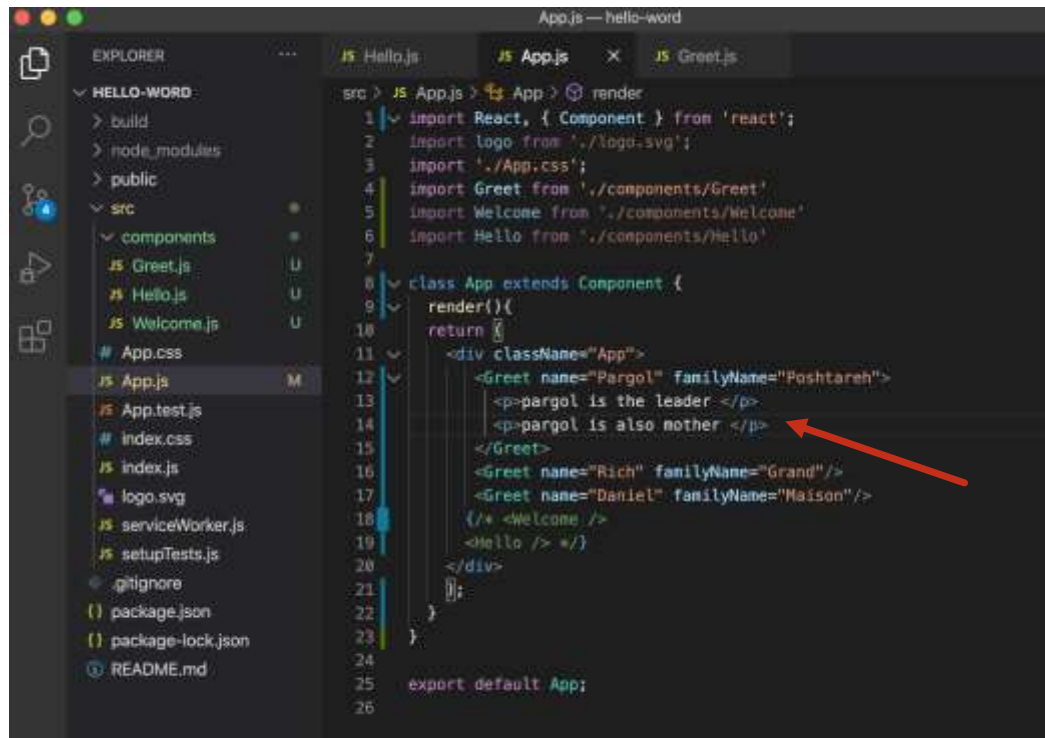
The screenshot shows the Chrome DevTools Console with the following content:

- Tab: Console
- Filter: top
- Messages:
  - [HMR] Waiting for update signal from WDS... [log.js:24](#)
  - Download the React DevTools for a better development experience: <https://fb.me/react-devtools> [react-dom.development.js:24994](#)
  - ▶ {name: "Pargol", familyName: "Poshtareh"} [Greet.js:8](#)
  - ▶ {name: "Rich", familyName: "Grand"} [Greet.js:8](#)
  - ▶ {name: "Daniel", familyName: "Maison"} [Greet.js:8](#)
- Errors (highlighted in yellow):
  - ▶ ./src/App.js [webpackHotDevClient.js:138](#)
    - Line 2:8: 'logo' is defined but never used no-unused-vars
    - Line 5:8: 'Welcome' is defined but never used no-unused-vars
    - Line 6:8: 'Hello' is defined but never used no-unused-vars
- Bottom: > |

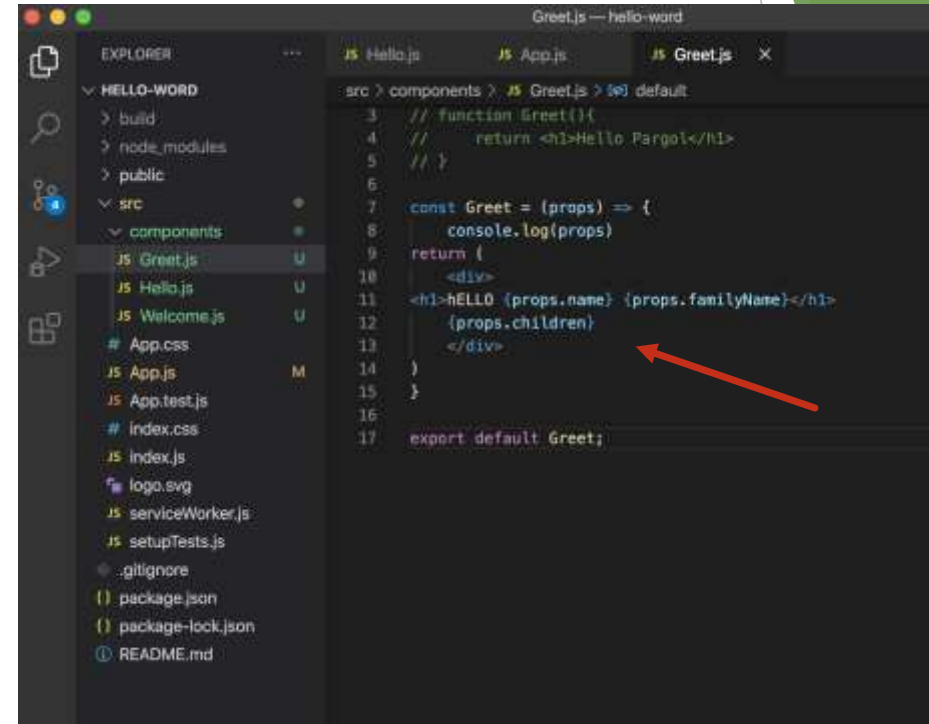


## Now Children of components :

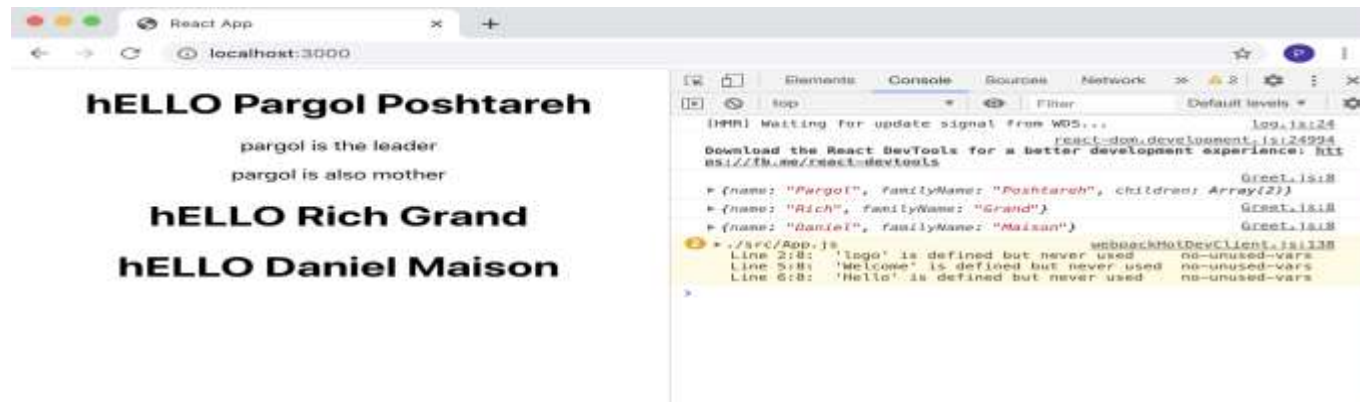
For sure we need <div>



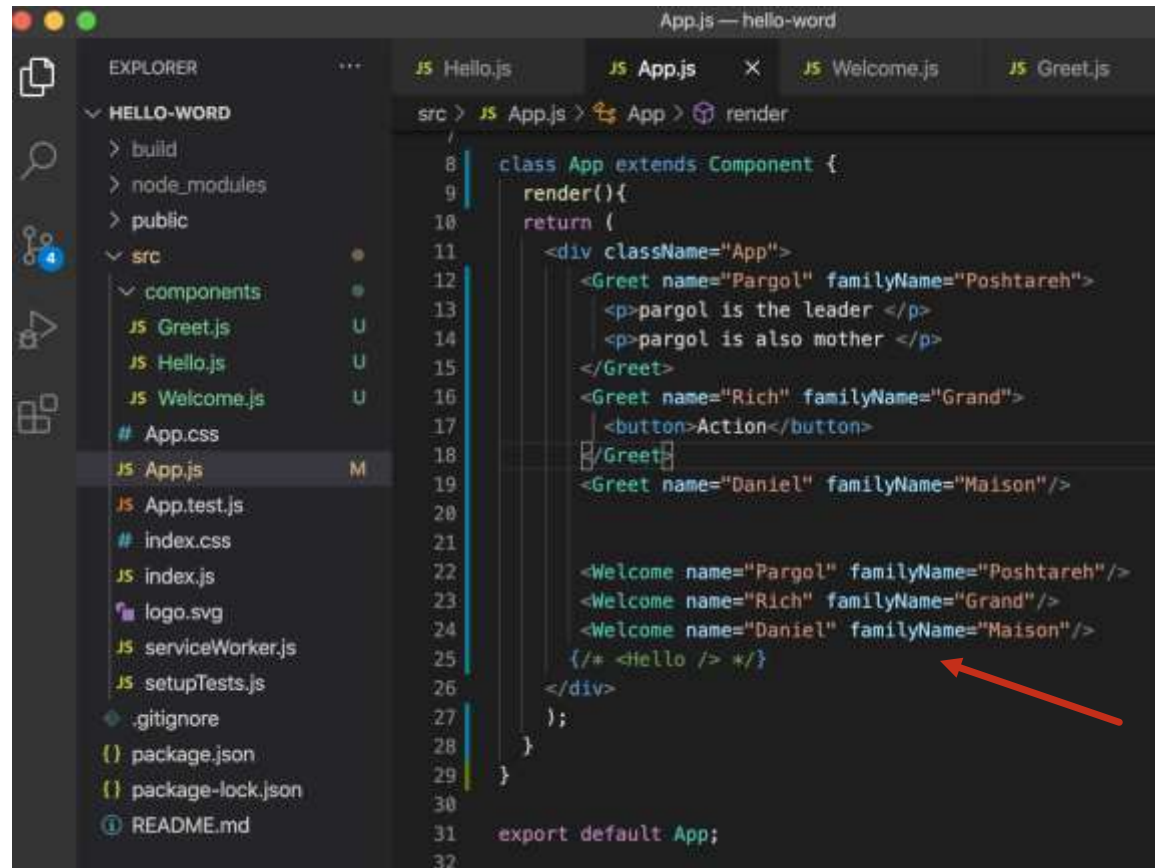
```
src > JS App.js > App > render
1 import React, { Component } from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4 import Greet from './components/Greet'
5 import Welcome from './components/Welcome'
6 import Hello from './components/Hello'
7
8 class App extends Component {
9   render() {
10     return (
11       <div className="App">
12         <Greet name="Pargol" familyName="Poshtareh">
13           <p>pargol is the leader </p>
14           <p>pargol is also mother </p>
15         </Greet>
16         <Greet name="Rich" familyName="Grand"/>
17         <Greet name="Daniel" familyName="Maison"/>
18         { /* <Welcome /> */ }
19         <Hello />
20       </div>
21     );
22   }
23 }
24
25 export default App;
```



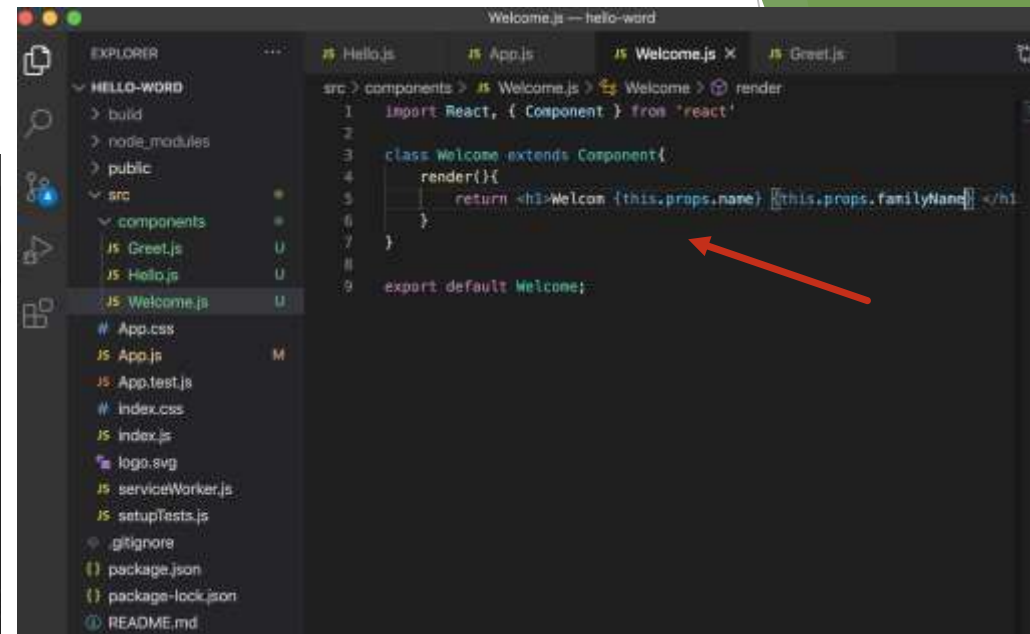
```
Greet.js — hello-word
src > components > JS Greet.js > default
3 // function Greet(){
4 //   return <h1>Hello Pargol</h1>
5 // }
6
7 const Greet = (props) => {
8   console.log(props)
9   return (
10     <div>
11       <h1>HELLO {props.name} {props.familyName}</h1>
12       {props.children}
13     </div>
14   )
15 }
16
17 export default Greet;
```



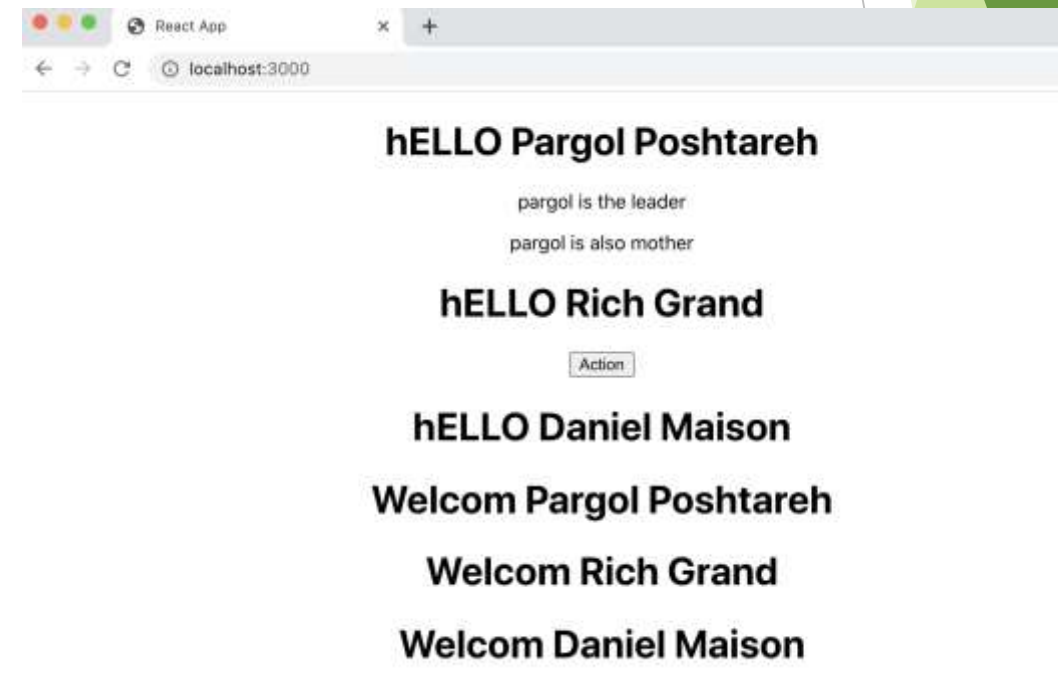
## How about Class Components:



```
8 class App extends Component {
9   render(){
10    return (
11      <div className="App">
12        <Greet name="Pargol" familyName="Poshtareh">
13          <p>pargol is the leader </p>
14          <p>pargol is also mother </p>
15        </Greet>
16        <Greet name="Rich" familyName="Grand">
17          <button>Action</button>
18        </Greet>
19        <Greet name="Daniel" familyName="Maison"/>
20
21        <Welcome name="Pargol" familyName="Poshtareh"/>
22        <Welcome name="Rich" familyName="Grand"/>
23        <Welcome name="Daniel" familyName="Maison"/>
24        { /* <Hello /> */ }
25      </div>
26    );
27  }
28 }
29
30
31 export default App;
```



```
1 import React, { Component } from 'react'
2
3 class Welcome extends Component{
4   render(){
5     return <h1>Welcom {this.props.name} {this.props.familyName} </h1>
6   }
7 }
8
9 export default Welcome;
```



You need this.prop

# *Props VS state:*

## props

props get passed to the component

Function parameters

props are immutable

props – Functional Components

this.props – Class Components

## state

state is managed within the component

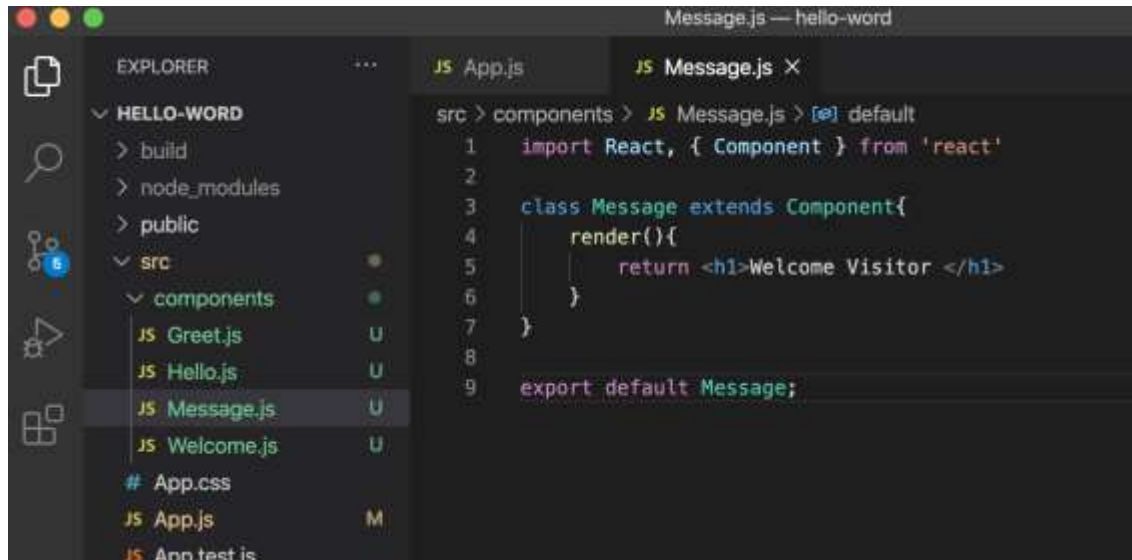
Variables declared in the function body

state can be changed

useState Hook – Functional Components

this.state – Class Components

**Example** Create new component and named Message.js



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'HELLO-WORD' with a 'src' directory containing a 'components' subdirectory. The 'components' directory contains four files: 'Greet.js', 'Hello.js', 'Message.js', and 'Welcome.js'. The 'Message.js' file is selected. The code editor shows the following code:

```
src > components > JS Message.js > [e] default
1  import React, { Component } from 'react'
2
3  class Message extends Component{
4      render(){
5          return <h1>Welcome Visitor </h1>
6      }
7  }
8
9  export default Message;
```

Imagine you want to put a button which if you click the message will change.

In this scenario you can't do with Prop, so state will involve but how?

We need to create the state inside constructor and initial it



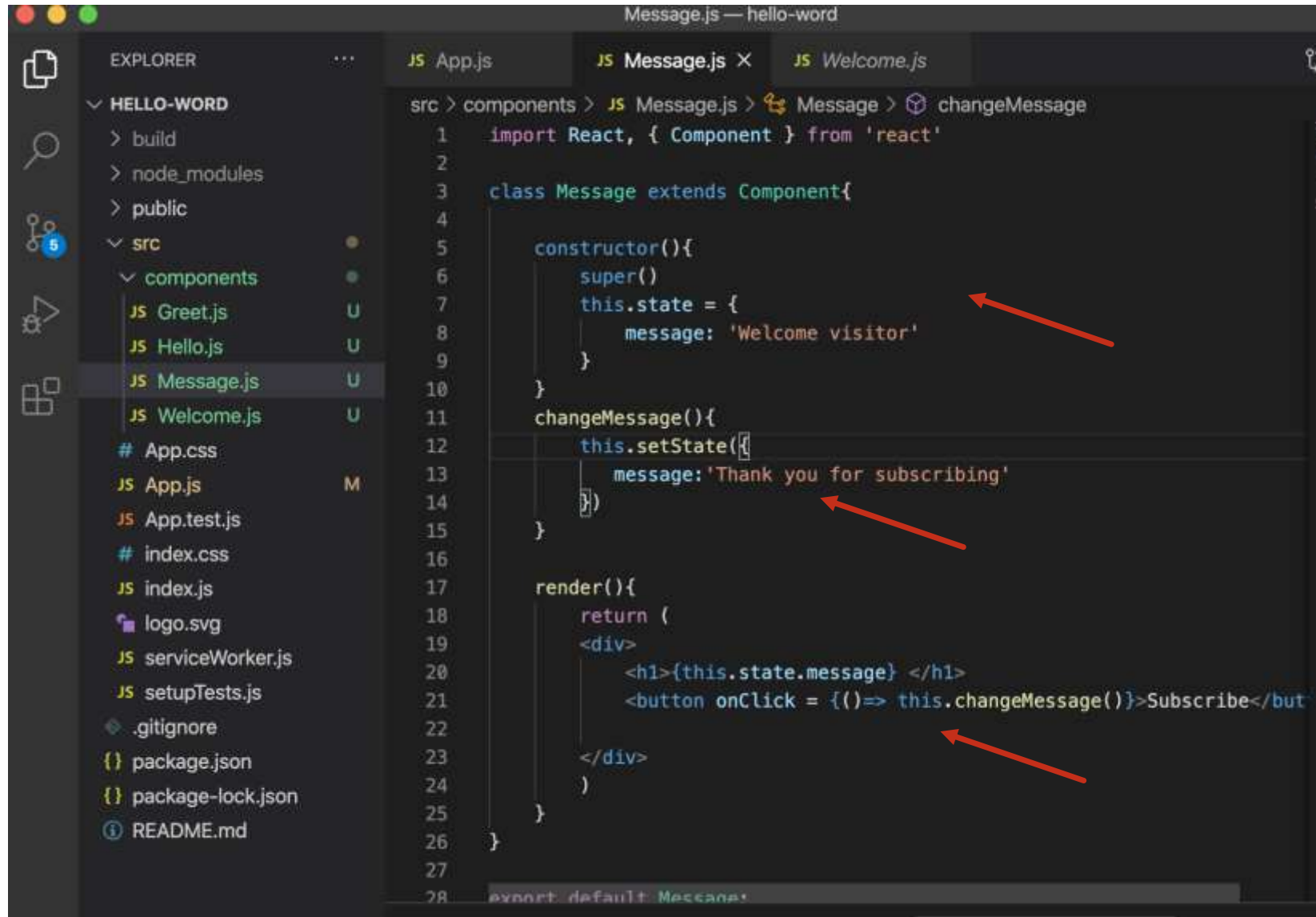
Message.js — hello-word

EXPLORER

- HELLO-WORD
  - build
  - node\_modules
  - public
  - src
    - components
      - Greet.js
      - Hello.js
      - Message.js
      - Welcome.js

src > components > JS Message.js > Message > changeMessage

```
1  import React, { Component } from 'react'
2
3  class Message extends Component{
4
5      constructor(){
6          super()
7          this.state = {
8              message: 'Welcome visitor'
9          }
10     }
11     changeMessage(){
12         this.setState({
13             message: 'Thank you for subscribing'
14         })
15     }
16
17     render(){
18         return (
19             <div>
20                 <h1>{this.state.message} </h1>
21                 <button onClick = {()=> this.changeMessage()}>Subscribe</but
22             </div>
23         )
24     }
25 }
26
27
28 export default Message;
```



Finish