

LECTURE #3

By: Aqib Rehman

OUTLINE

Intro to Forms

Create React App

Single Page Apps

Nesting Components

Props

Outputting Lists

Stateless Components

INTRO TO FORMS

A screenshot of the Visual Studio Code interface showing an HTML file named "index.html". The code is a simple React application. It includes a root

element with the id "app", a script block containing a class definition for "App" extending "React.Component", and a render method that creates an

element displaying the state "name" and a form with an input field and a submit button.

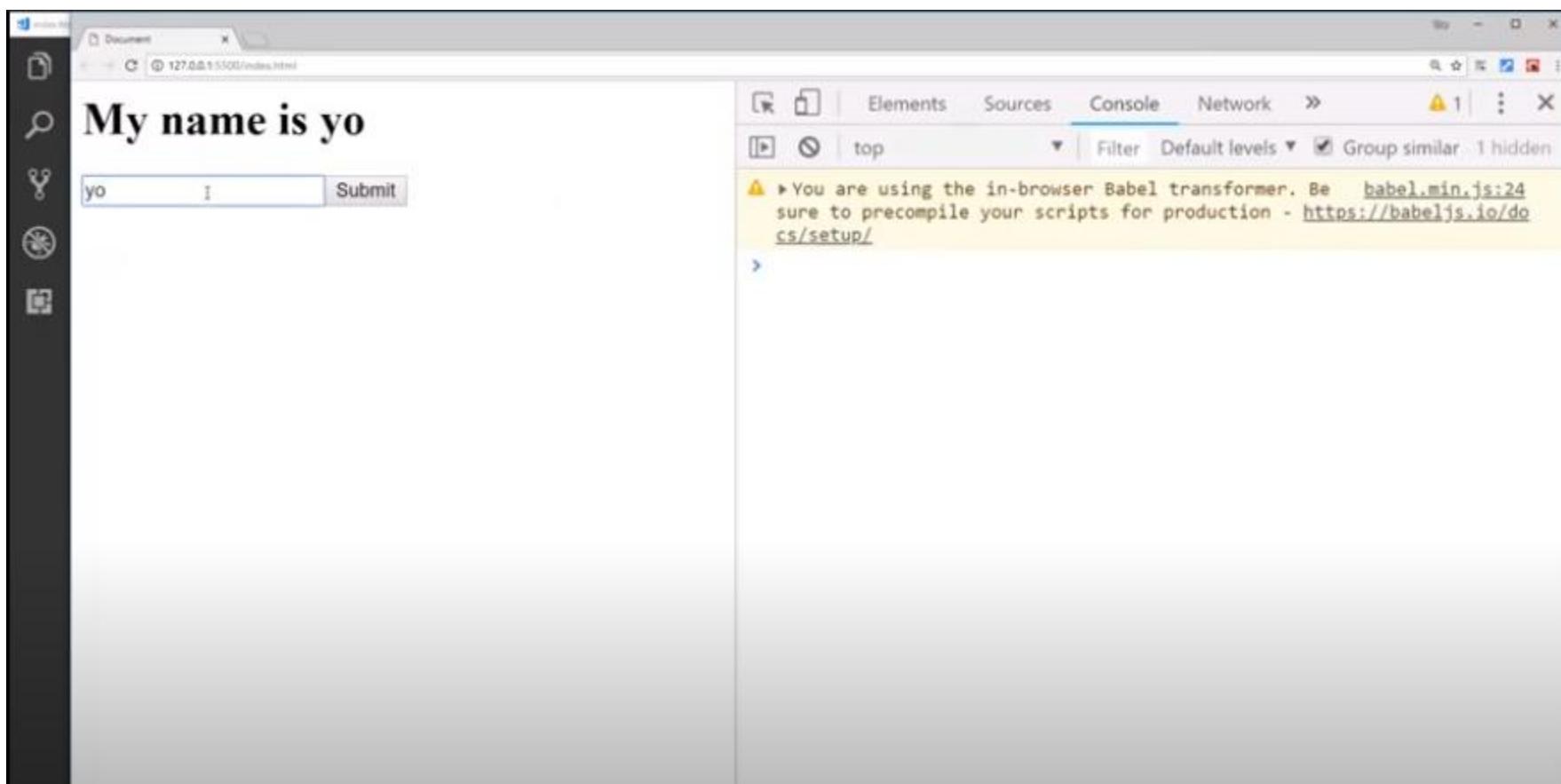
```
12 <div id="app"></div>
13
14
15 <script type="text/babel">
16   class App extends React.Component {
17     state = {
18       name: 'Ryu',
19       age: 30
20     }
21     render() {
22       return (
23         <div className="app-content">
24           <h1>My name is { this.state.name }</h1>
25           <form>
26             <input type="text" onChange={()}/>
27             <button>Submit</button>
28           </form>
29         </div>
30       )
31     }
32   }
33 </script>
```

A screenshot of the Visual Studio Code interface, showing the file `index.html` open. The code is a simple React component named `App`. It includes state management for name and age, and a form to change the name.

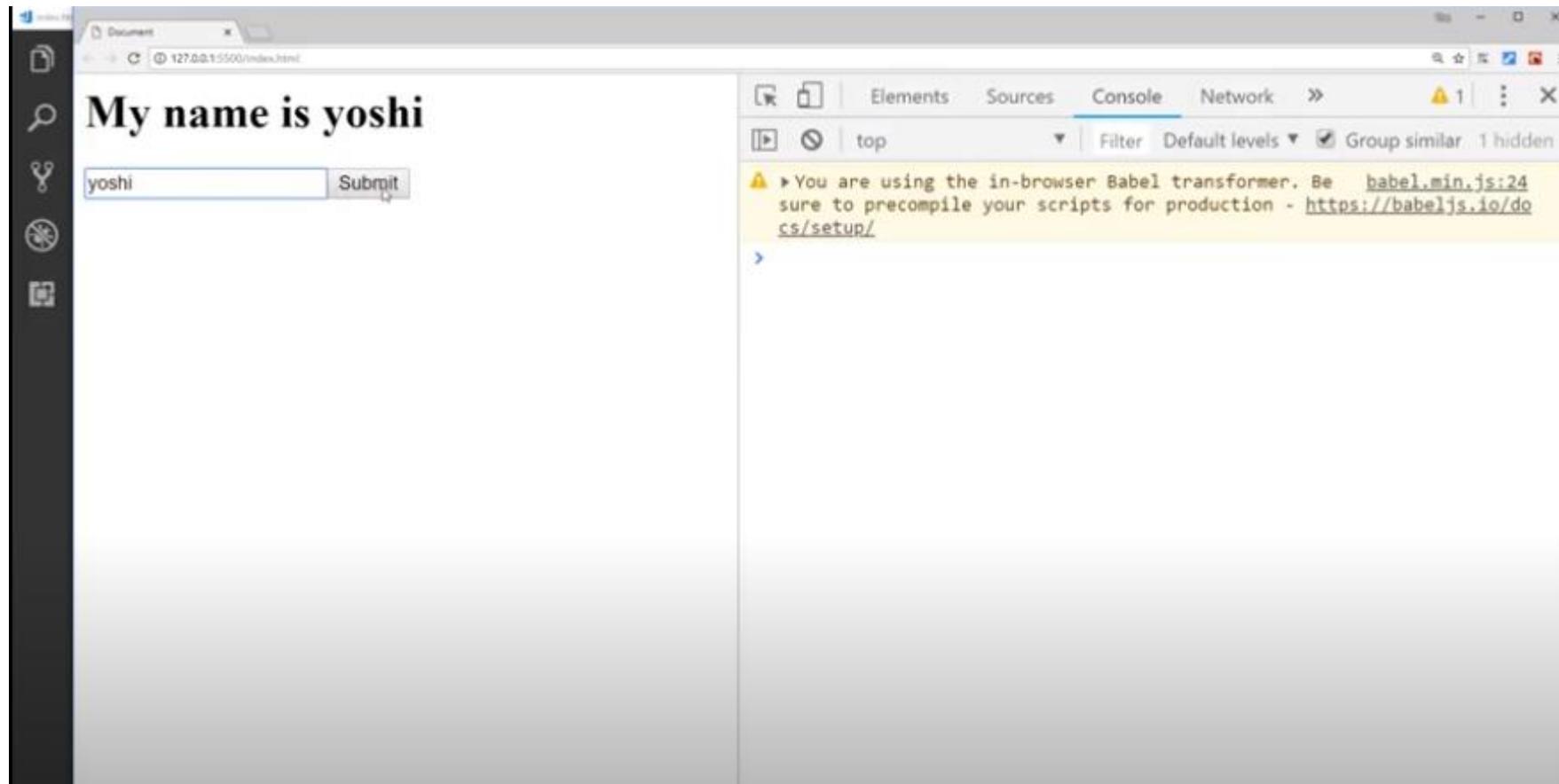
```
12 <div id="app"></div>
13
14
15 <script type="text/babel">
16   class App extends React.Component {
17     state = {
18       name: 'Ryu',
19       age: 30
20     }
21     handleChange = (e) => {
22       this.setState({
23         name: e.target.value
24       })
25     }
26     render() {
27       return (
28         <div className="app-content">
29           <h1>My name is { this.state.name }</h1>
30           <form>
31             <input type="text" onChange={} />
32             <button>Submit</button>
33           </form>
34         </div>
35       )
36     }
37   }
38 </script>
```

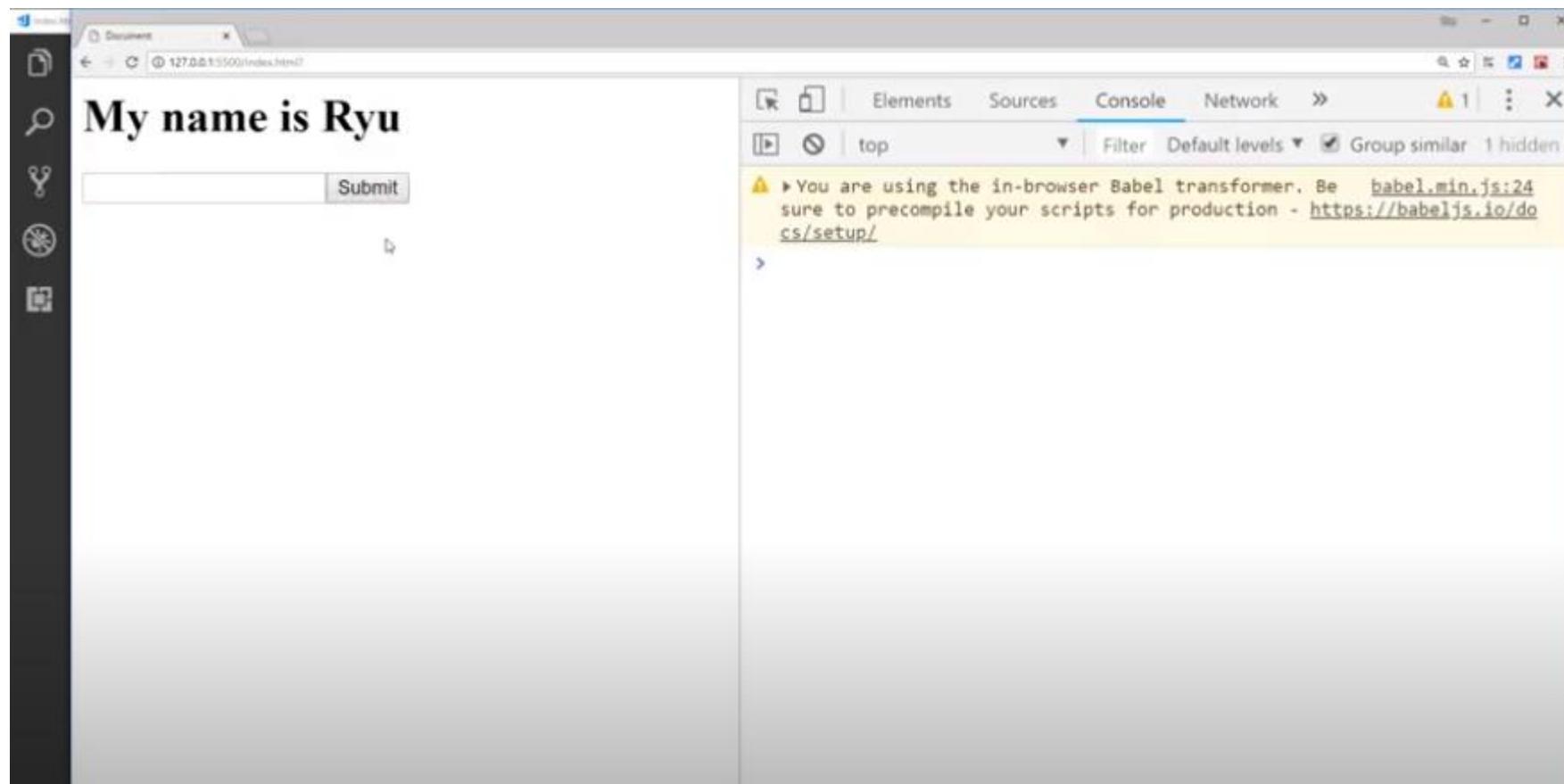
A screenshot of the Visual Studio Code interface, showing a file named "index.html" with the title "index.html - react-reduce-complete". The code is written in JSX and defines a React component named "App". The component has a state with "name" and "age" properties, and a "handleChange" handler for a text input. The "render" method returns a div containing an h1 element with the state name and a form with an input and a button. A code completion dropdown is open over the "handleChange" prop of the input element, showing the "handleChange" suggestion.

```
12 <div id="app"></div>
13
14
15 <script type="text/babel">
16   class App extends React.Component {
17     state = {
18       name: 'Ryu',
19       age: 30
20     }
21     handleChange = (e) => {
22       this.setState({
23         name: e.target.value
24       });
25     }
26     render() {
27       return (
28         <div className="app-content">
29           <h1>My name is { this.state.name }</h1>
30           <form>
31             <input type="text" onChange={this.handleChange} />
32             <button>Submit</button>
33           </form>
34         </div>
```

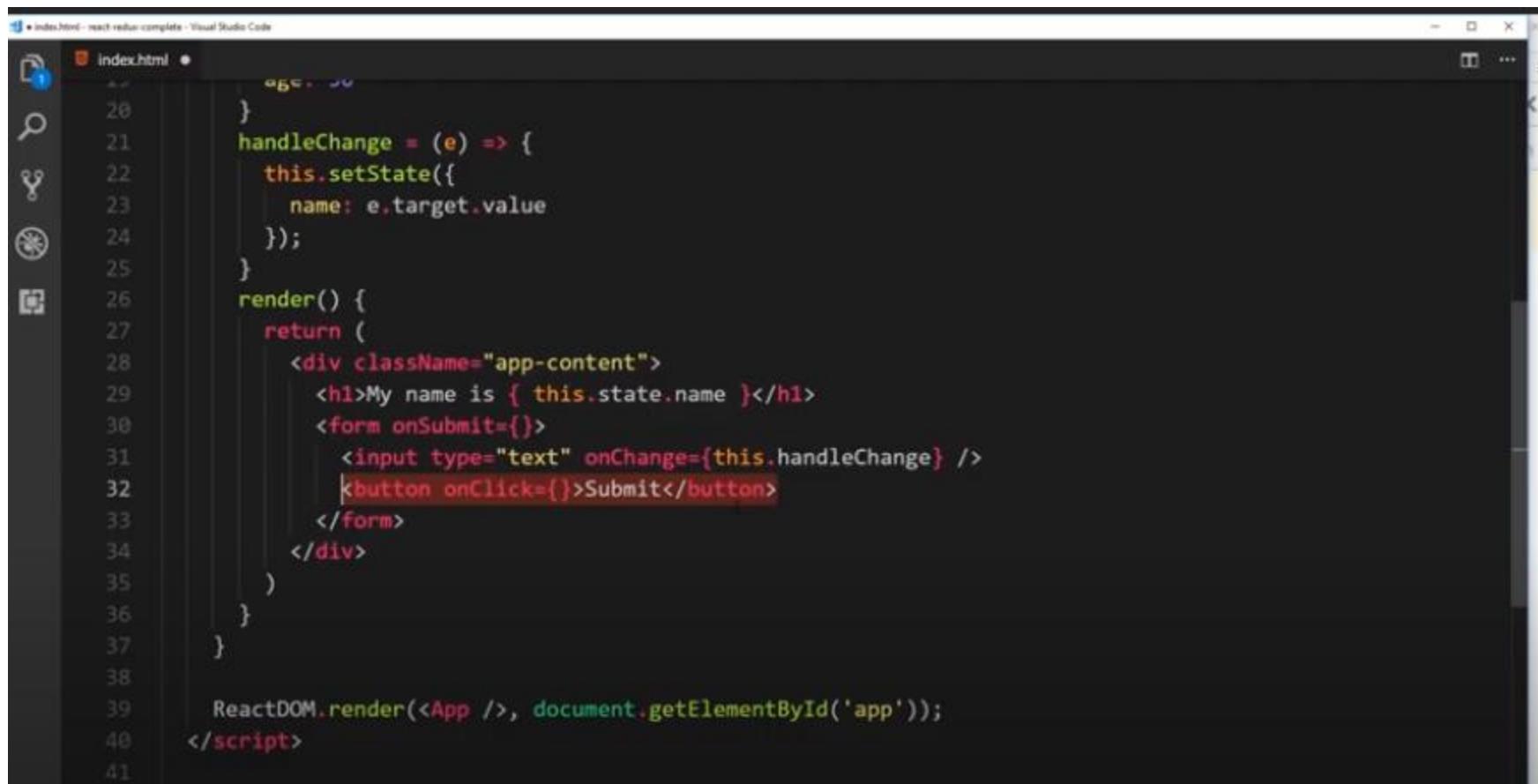


When click on submit page will be refresh default event
of form submit





Only trigger when button will be clicked not on Enter pressed



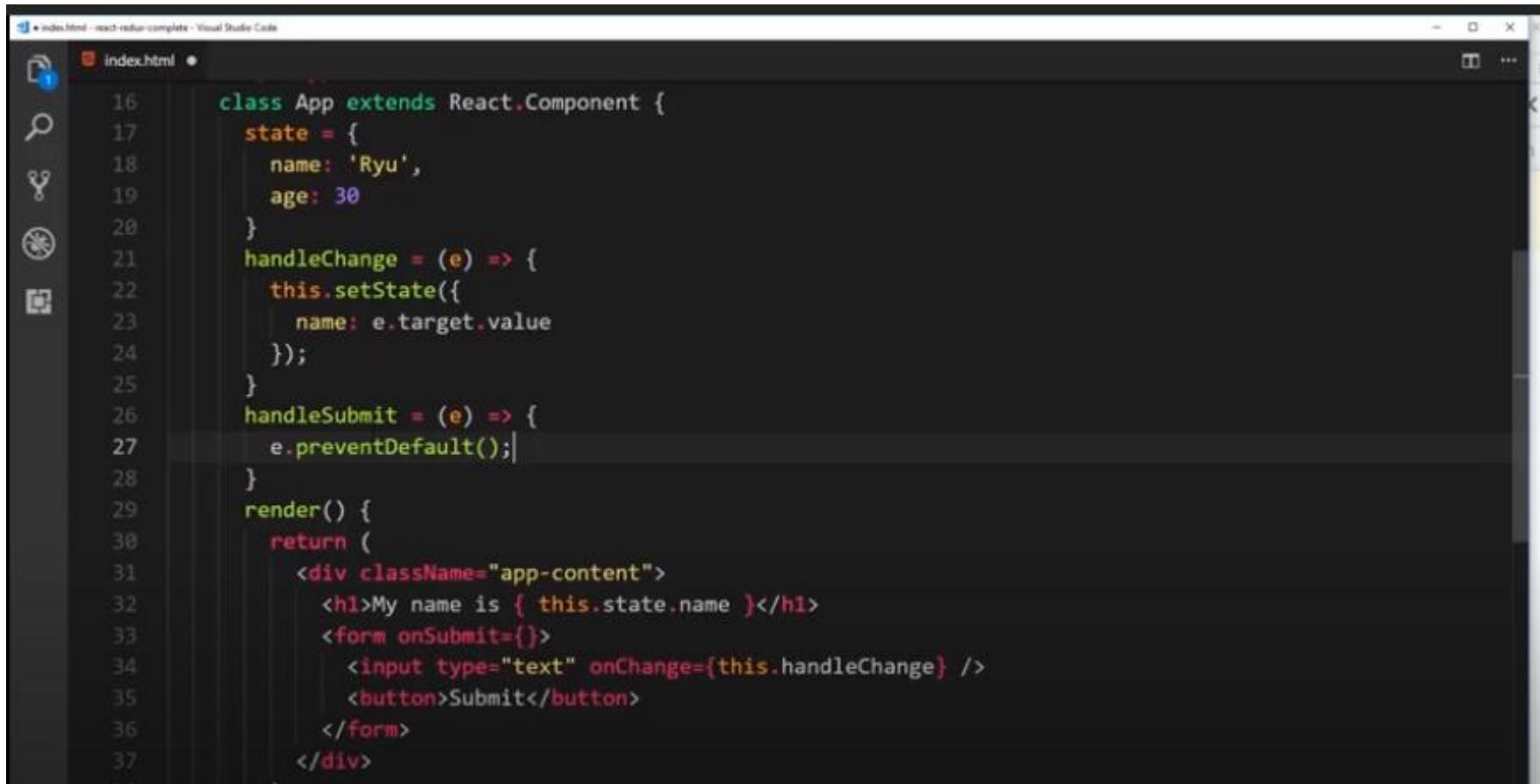
A screenshot of the Visual Studio Code interface showing an HTML file named "index.html". The code is written in JSX (JavaScript with XML-like structures). It defines a class "App" with a constructor that sets up a state "name" and a handler "handleChange". The "render" method creates a user interface with a text input and a button. The text input has an "onChange" event listener pointing to "handleChange". The button has an "onClick" event listener pointing to a function that returns "Submit". Finally, the "ReactDOM.render" function is used to render the "App" component into the element with id "app".

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>React App</title>
5   </head>
6   <body>
7     <div id="app"></div>
8   </body>
9 </html>
10
11 <script>
12   class App extends React.Component {
13     constructor(props) {
14       super(props);
15       this.state = {
16         name: ''
17       };
18       this.handleChange = this.handleChange.bind(this);
19     }
20     handleChange = (e) => {
21       this.setState({
22         name: e.target.value
23       });
24     }
25     render() {
26       return (
27         <div className="app-content">
28           <h1>My name is { this.state.name }</h1>
29           <form onSubmit={this.handleSubmit}>
30             <input type="text" onChange={this.handleChange} />
31             <button onClick={()=>this.handleSubmit}>Submit</button>
32           </form>
33         </div>
34       )
35     }
36   }
37
38   ReactDOM.render(<App />, document.getElementById('app'));
39 </script>
40
41 
```

A screenshot of the Visual Studio Code interface showing an editor window for a file named "index.html". The title bar indicates the file is part of a project named "react-redux-complete". The code in the editor is a React component definition:

```
16  class App extends React.Component {
17    state = {
18      name: 'Ryu',
19      age: 30
20    }
21    handleChange = (e) => {
22      this.setState({
23        name: e.target.value
24      });
25    }
26    render() {
27      return (
28        <div className="app-content">
29          <h1>My name is { this.state.name }</h1>
30          <form onSubmit={this.handleSubmit}>
31            <input type="text" onChange={this.handleChange} />
32            <button>Submit</button>
33          </form>
34        </div>
35      )
36    }
37  }
```

Now page will not be refreshed

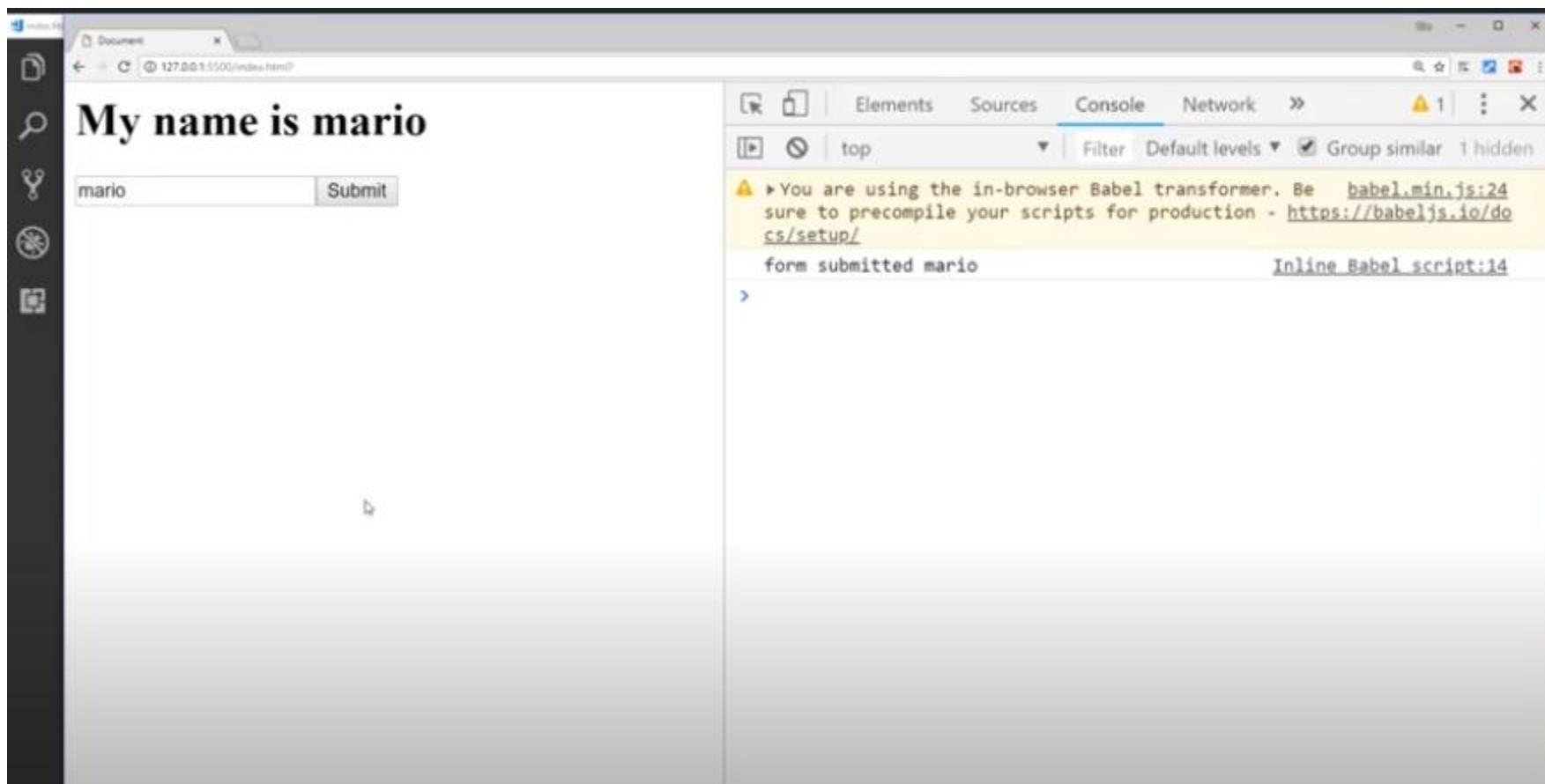


The screenshot shows a Visual Studio Code window with a dark theme. The title bar says "index.html - react-redux-complete - Visual Studio Code". The left sidebar has icons for file, search, and refresh. The main editor area contains the following code:

```
16  class App extends React.Component {
17    state = {
18      name: 'Ryu',
19      age: 30
20    }
21    handleChange = (e) => {
22      this.setState({
23        name: e.target.value
24      });
25    }
26    handleSubmit = (e) => {
27      e.preventDefault();
28    }
29    render() {
30      return (
31        <div className="app-content">
32          <h1>My name is { this.state.name }</h1>
33          <form onSubmit={this.handleSubmit}>
34            <input type="text" onChange={this.handleChange} />
35            <button>Submit</button>
36          </form>
37        </div>
38      );
39    }
40  }
41
42  export default App;
```

A screenshot of the Visual Studio Code interface, showing the file `index.html` open. The code is a simple React component named `App`. It includes state management, event handling for input changes, and a submit handler that logs the submitted name to the console. A code completion dropdown is visible, listing `name` and `className` as suggestions for the current context.

```
16  class App extends React.Component {
17    state = {
18      name: 'Ryu',
19      age: 30
20    }
21    handleChange = (e) => {
22      this.setState({
23        name: e.target.value
24      });
25    }
26    handleSubmit = (e) => {
27      e.preventDefault();
28      console.log('form submitted', this.state.name)
29    }
30    render() {
31      return (
32        <div className="app-content">
33          <h1>My name is { this.state.name }</h1>
34          <form onSubmit={this.handleSubmit}>
35            <input type="text" onChange={this.handleChange} />
36            <button>Submit</button>
37          </form>
38      )
39    }
40  }
41
42  export default App;
```



CREATE REACT APP

Create React App

- A command line tool to create React apps
 - Development server
 - Use ES6 features which are not normally supported
 - Keep our code modular
 - Use build tools to create optimized code

Quick Overview

```
npx create-react-app my-app  
cd my-app  
npm start
```

([npx comes with npm 5.2+ and higher, see instructions for older npm versions](#))

Then open <http://localhost:3000/> to see your app.

When you're ready to deploy to production, create a minified bundle with `npm run build`.

The screenshot shows the official Node.js website (<https://nodejs.org/en/>) displayed in a web browser. The page has a dark header with the Node.js logo and navigation links for HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION. The FOUNDATION link is highlighted with a green background. Below the header, a banner for "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." is visible. A promotional box for "js Interactive" featuring a hexagonal icon, the text "js Interactive", and event details ("join us at jsinteractive October 10-12, 2018 Vancouver, Canada") is present. The main download section features two large buttons: "8.11.3 LTS" (Recommended For Most Users) and "10.7.0 Current" (Latest Features). Below these buttons, smaller links for "Other Downloads", "Download 8.11.3 LTS", "Docs", "Other Downloads", "Changelog", and "API Docs" are shown. A note encourages users to look at the Long Term Support (LTS) schedule, and a call-to-action invites users to sign up for the "Node.js Everywhere" newsletter.

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

js Interactive
join us at jsinteractive
October 10-12, 2018
Vancouver, Canada

Download for Windows (x64)

8.11.3 LTS
Recommended For Most Users

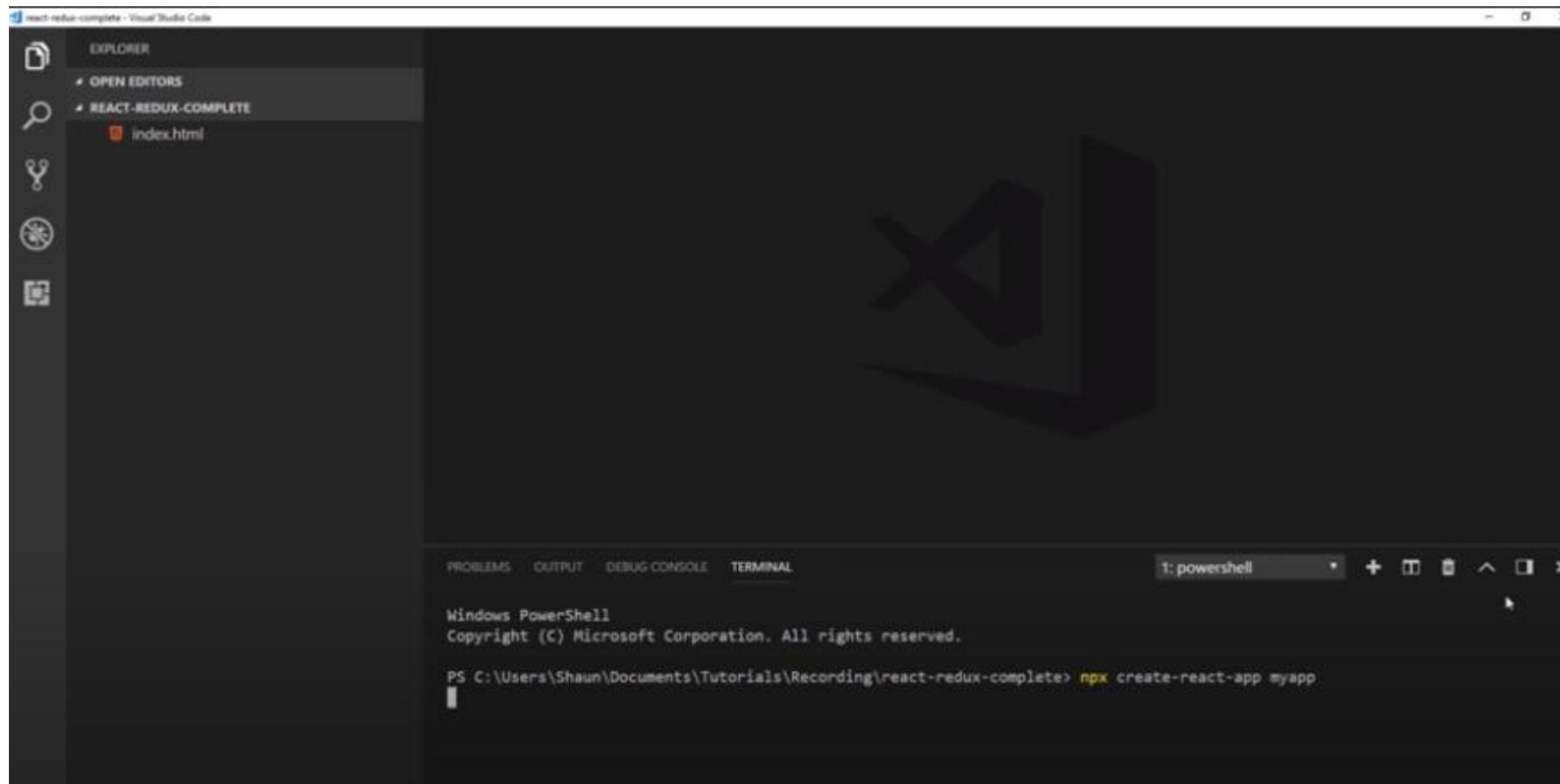
10.7.0 Current
Latest Features

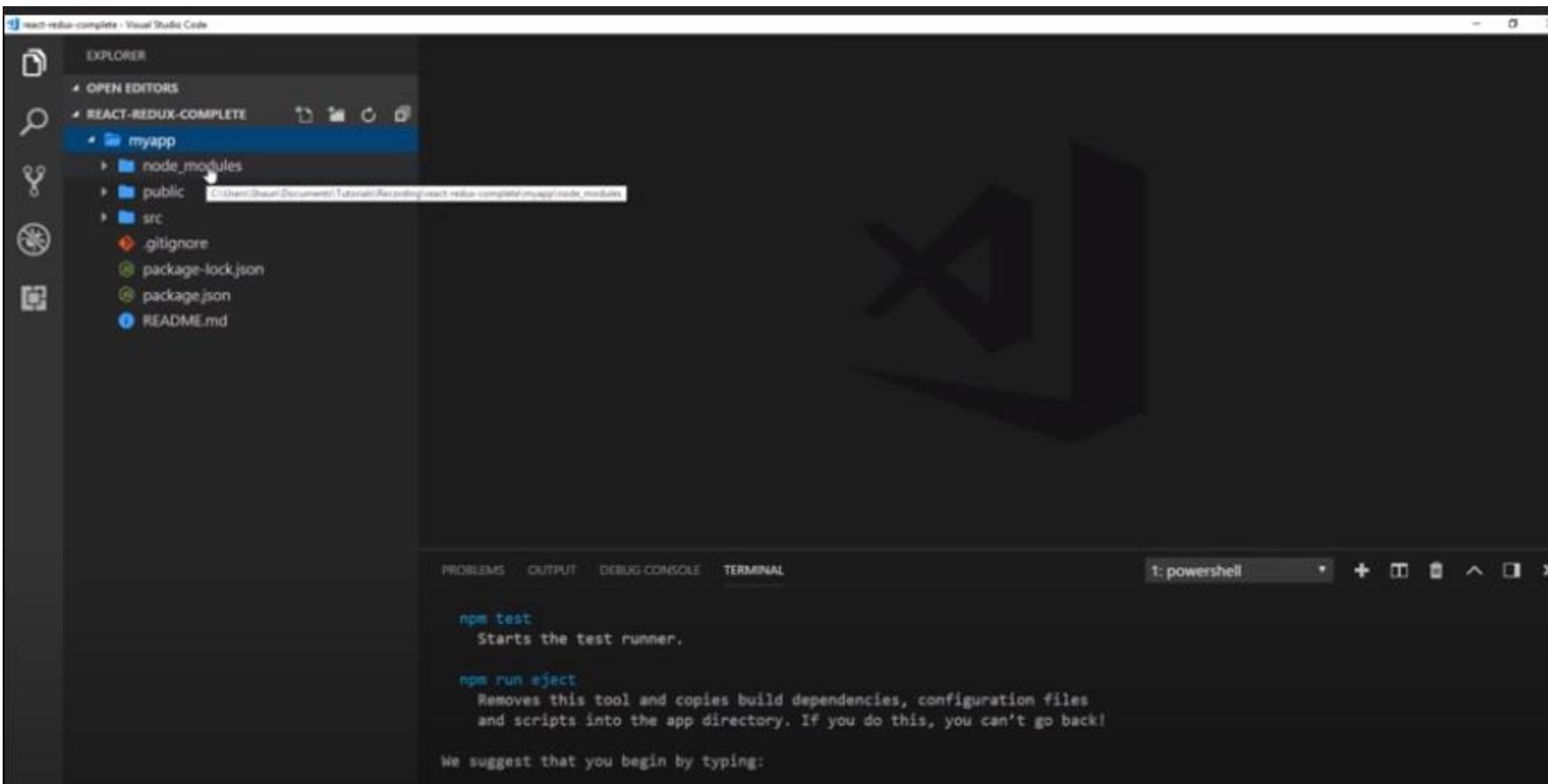
Other Downloads | Download 8.11.3 LTS | Docs | Other Downloads | Changelog | API Docs

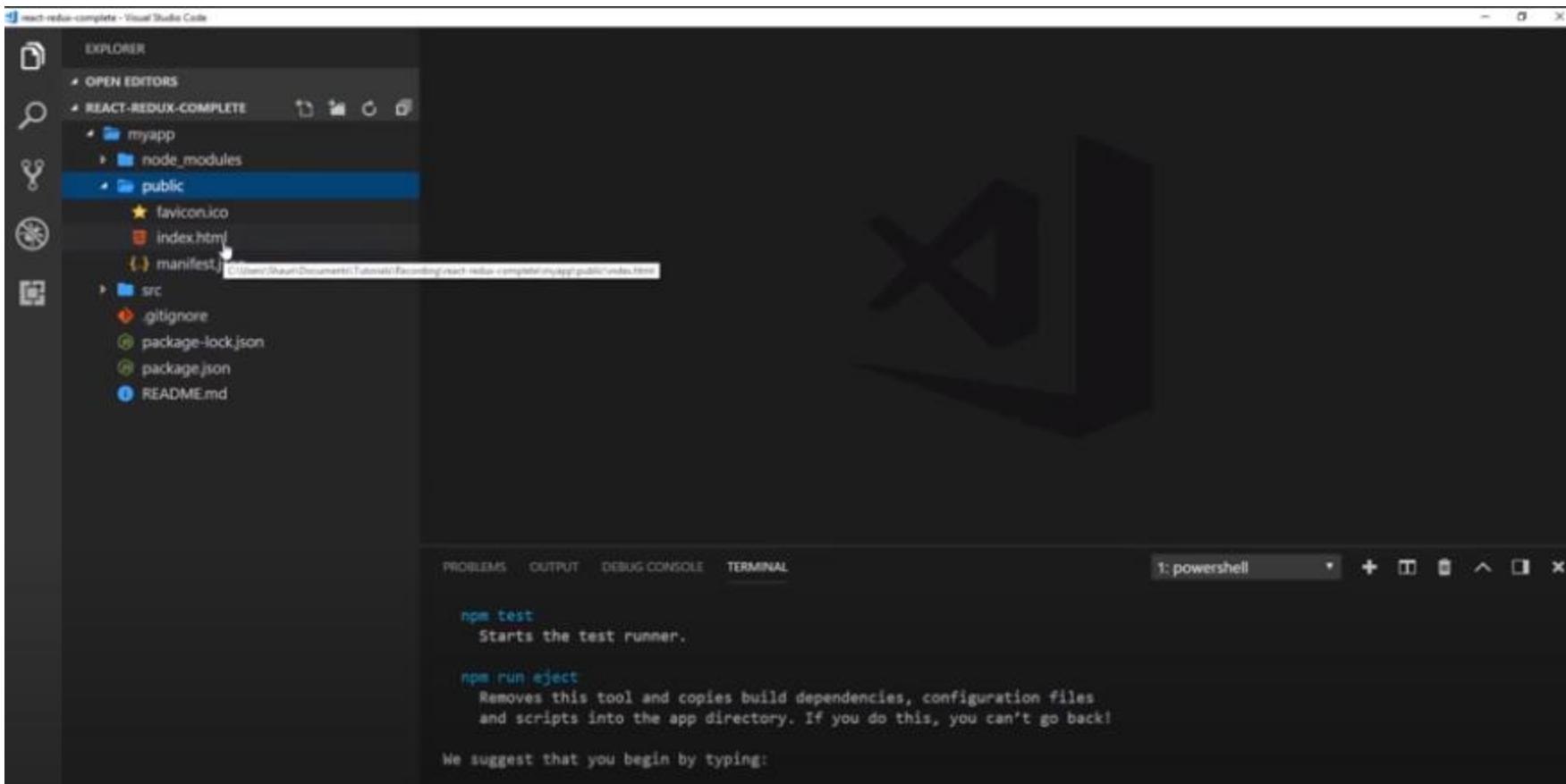
Or have a look at the Long Term Support (LTS) schedule.

Sign up for Node.js Everywhere, the official Node.js Weekly Newsletter.

CREATING APP WITH COMMAND







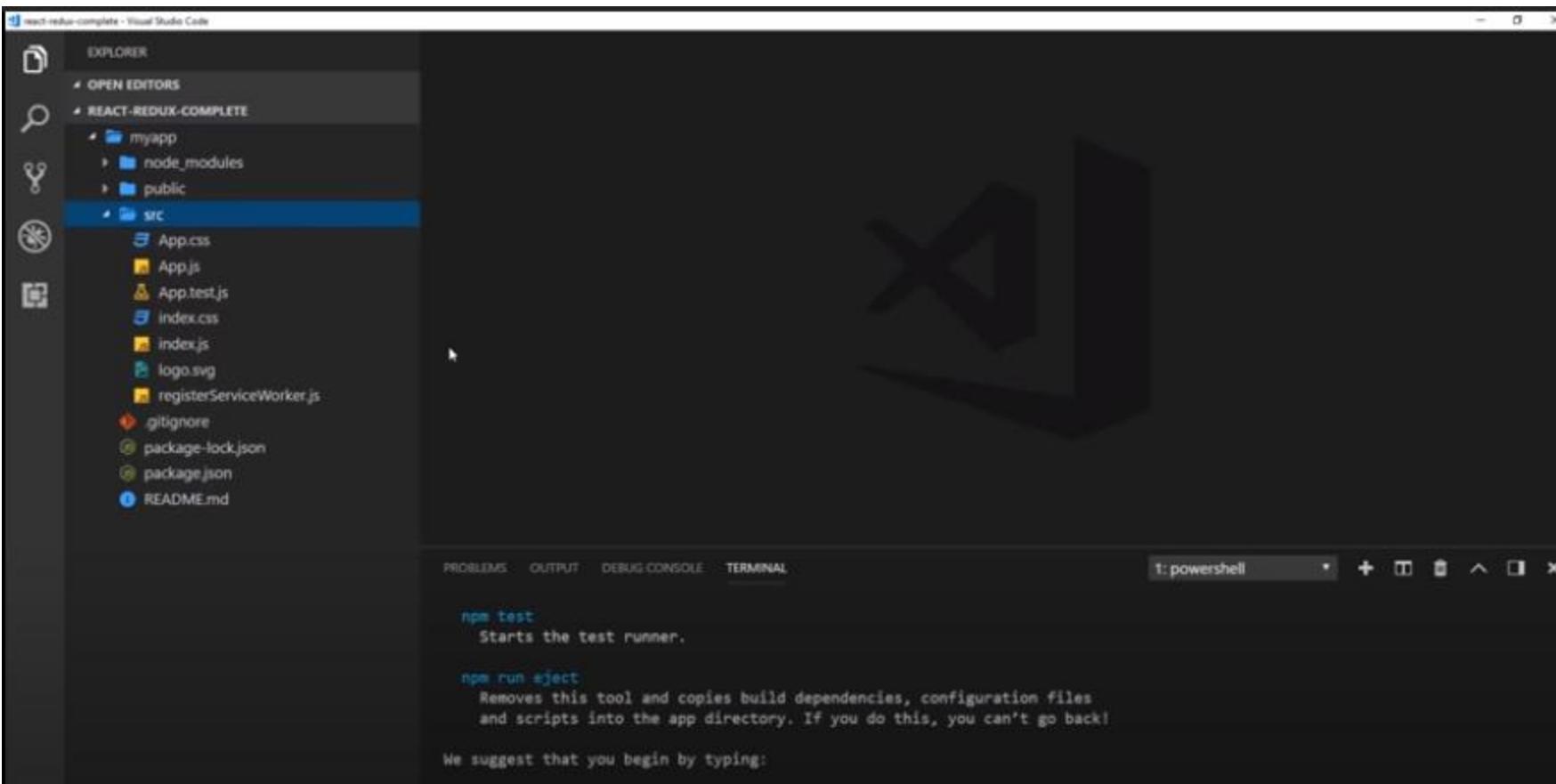
The screenshot shows a Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "REACT-REDUX-COMPLETE". The "myapp" folder contains "node_modules", "public", and "src" subfolders, along with ".gitignore", "package-lock.json", and "package.json".
- Editor View:** Displays the "package.json" file content. The file specifies dependencies for "react-dom" and "react-scripts", and defines scripts for "start", "build", "test", and "eject".

```
    "react-dom": "^16.4.1",
    "react-scripts": "1.1.4"

  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
}
```
- Terminal View:** Shows npm commands and their descriptions:
 - `npm test`: Starts the test runner.
 - `npm run eject`: Removes this tool and copies build dependencies, configuration files and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:



The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "REACT-REDUX-COMPLETE". The "index.html" file is selected and highlighted in blue.
- Editor View:** Displays the content of the "index.html" file. The code includes a title, a noscript block for JavaScript enablement, and a root div for component injection.

```
<title>React App</title>
</head>
<body>
  <noscript>
    You need to enable JavaScript to run this app.
  </noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.
  -->
  You can add webfonts, meta tags, or analytics to this file.

```

- Terminal View:** Shows a Windows PowerShell window with the following output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All
rights reserved.

PS C:\Users\Shaun\Documents\Tutorials\Recording\react-redux-complete>
```

Basically this index.html is our one page app. All the components will be injected dynamically in this root div

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "REACT-REDUX-COMPLETE". The "src" folder contains "App.js", "App.css", "index.css", "index.js", "logo.svg", and "registerServiceWorker.js".
- Code Editor:** The "App.js" file is open, displaying the following code:

```
1 import React, { Component } from 'react';
2 import logo from './logo.svg';
3 import './App.css';

4
5 class App extends Component {
6   render() {
7     return (
8       <div className="App">
9         <header className="App-header">
10           <img src={logo} className="App-logo" alt="logo" />
11           <h1 className="App-title">Welcome to React</h1>
12         </header>
13         <p className="App-intro">
14           To get started, edit <code>src/App.js</code> and save to reload.
15         </p>

```
- Terminal:** The terminal window shows a PowerShell session with the following output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All
rights reserved.

PS C:\Users\Shaun\Documents\Tutorials\Recording\react-redux-complete>
```

First component React created for us with name App.

Its same which we saw earlier like a class based component with render method having a jsx template code, returning a div.

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "REACT-REDUX-COMPLETE". The "src" folder contains "App.css", "App.js", "App.test.js", "index.css", "index.js", "logo.svg", and "registerServiceWorker.js".
- Editor View:** The "App.js" file is open. The code is as follows:

```
8 <div className="App">
9   <header className="App-header">
10    <img src={logo} className="App-logo" alt="logo" />
11    <h1 className="App-title">Welcome to React</h1>
12  </header>
13  <p className="App-intro">
14    To get started, edit <code>src/App.js</code> and save to reload.
15  </p>
16 </div>
17 );
18 }
19 <export default App;>
20
21 </export>
22
```

The code includes a header with a logo and a title, followed by an introductory paragraph. The component is wrapped in a div and exported at the end.

Bottom Status Bar: Shows "Windows PowerShell" and "Copyright (C) Microsoft Corporation. All rights reserved."

At the end we are exporting that component so that we can render it to the DOM.

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer Panel:** Shows the project structure under "REACT-REDUX-COMPLETE".
 - myapp:** Contains node_modules, public (with favicon.ico, index.html, manifest.json), and src (with App.css, App.js, App.test.js, index.css, index.js, logo.svg, registerServiceWorker.js).
 - REACT-REDUX-COMPLETE:** Contains myapp/src (with App.js and index.js).
- Editor:** The "index.js" file is open, showing the following code:

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import registerServiceWorker from './registerServiceWorker';
6
7 ReactDOM.render(<App />, document.getElementById('root'));
8 registerServiceWorker();
```
- Terminal:** A PowerShell terminal window is open at the bottom, showing the command prompt and the current working directory.

Importing that component in index.js and rendering it to the root element of index.html
So basically here we are inject our component with ReactDOM.render method

How to run this app

The screenshot shows a Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "REACT-REDUX-COMPLETE". The "myapp" folder contains "node_modules", "public" (with favicon.ico, index.html, manifest.json), and "src" (with App.css, App.js, App.test.js, index.css, index.js, logo.svg, registerServiceWorker.js, .gitignore, package-lock.json, package.json, and README.md). "App.js" is currently selected.
- Code Editor:** Displays the content of App.js:

```
1 import React, { Component } from 'react';
2 import logo from './logo.svg';
3 import './App.css';

4
5 class App extends Component {
6   render() {
7     return (
8       <div className="App">
9         <header className="App-header">
10           <img src={logo} className="App-logo" alt="logo" />
11           <h1 className="App-title">Welcome to React</h1>
12         </header>
13         <p className="App-intro">
14           To get started, edit <code>src/App.js</code> and save to reload.
15         </p>

```
- Terminal:** Shows a Windows PowerShell window with the following output:

```
PS C:\Users\Shaun\Documents\Tutorials\Recording\react-redux-complete> cd myapp
PS C:\Users\Shaun\Documents\Tutorials\Recording\react-redux-complete\myapp> npm start
```

Firstly with cd command will go into the directory

Then will start a local server

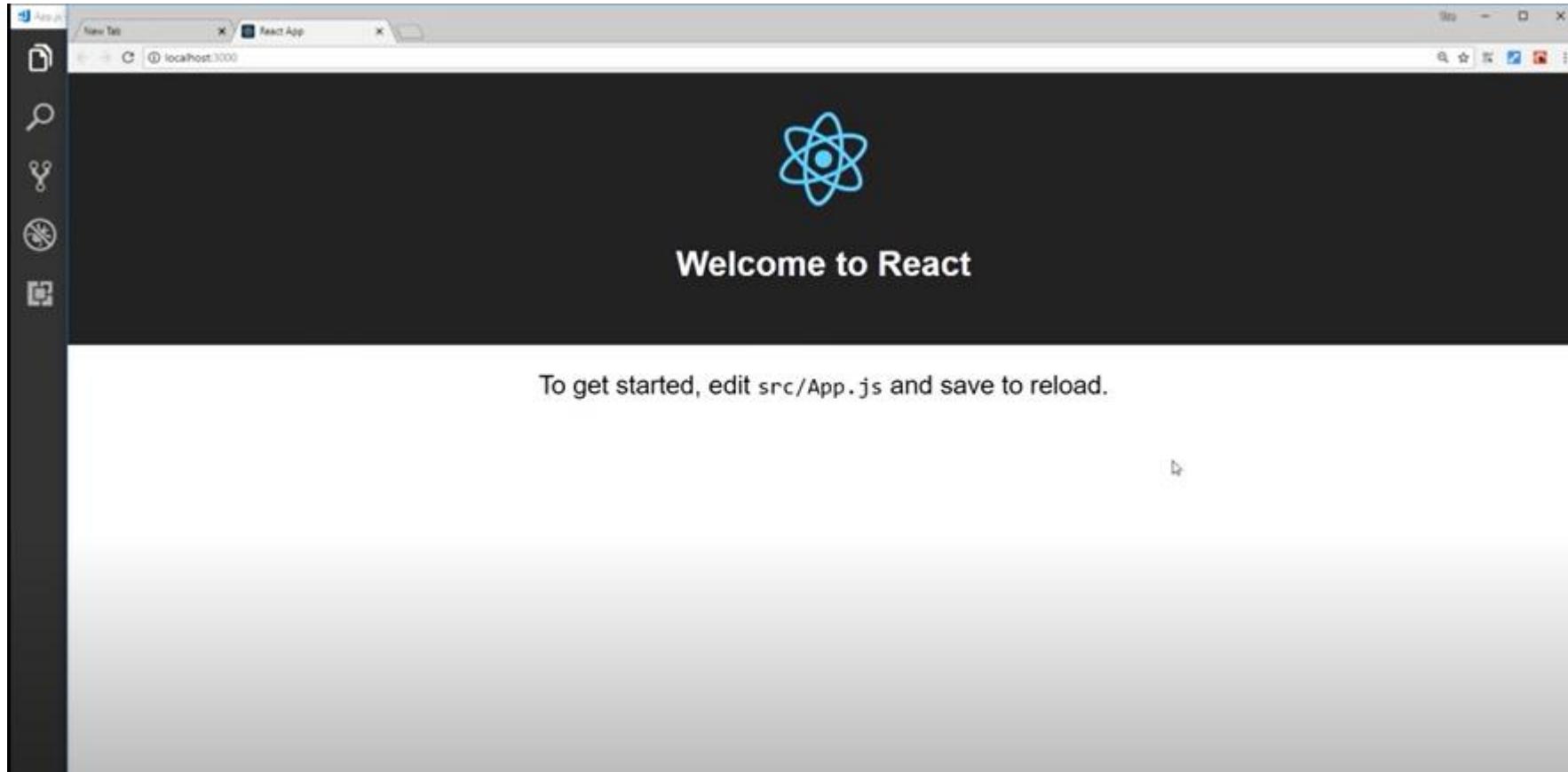
The screenshot shows a Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists project files: 'myapp' (containing 'node_modules', 'public' (with 'favicon.ico', 'index.html', 'manifest.json'), and 'src' (containing 'App.css', 'App.js', 'App.test.js', 'index.css', 'index.js', 'logo.svg', 'registerServiceWorker.js', '.gitignore', 'package-lock.json', 'package.json', and 'README.md')). The 'REACT-REDUX-COMPLETE' extension is also listed. The main editor area displays the 'App.js' file content:

```
1 import React, { Component } from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4
5 class App extends Component {
6   render() {
7     return (
8       <div className="App">
9         <header className="App-header">
10           <img src={logo} className="App-logo" alt="logo" />
11           <h1 className="App-title">Welcome to React</h1>
12         </header>
13         <p className="App-intro">
14           To get started, edit <code>src/App.js</code> and save to reload.
15         </p>

```

The status bar at the bottom shows 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The terminal tab shows the command '1: node' and a message: 'Compiled successfully!'. Below this, it says 'You can now view myapp in the browser.' with a tooltip 'Ctrl + click to follow link'. It also provides local and network URLs: 'Local: http://localhost:3000/' and 'On Your Network: http://192.168.1.10:3000/'. A note at the bottom states: 'Note that the development build is not optimized. To create a production build, use `npm run build`'.

It will shows the ip address where that app is running



It will also be opened in browser automatically

```
App.js - react-redux-complete - Visual Studio Code
```

EXPLORER APPS

OPEN EDITORS App.js myapp/src

REACT-REDUX-COMPLETE

myapp

- node_modules
- public
 - favicon.ico
 - index.html
 - manifest.json
- src
 - App.css
 - App.js
 - App.test.js
 - index.css
 - index.js
 - logo.svg
 - registerServiceWorker.js
- .gitignore
- package-lock.json
- package.json
- README.md

```
1 import React, { Component } from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4
5 class App extends Component {
6   render() {
7     return (
8       <div className="App">
9         <header className="App-header">
10           <img src={logo} className="App-logo" alt="logo" />
11           <h1 className="App-title">Welcome to React</h1>
12         </header>
13         <p className="App-intro">
14           To get started, edit <code>src/App.js</code> and save to reload.
15         </p>

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Compiled successfully!

You can now view myapp in the browser.

Local: http://localhost:3000/
On Your Network: http://192.168.1.73:3000/

Note that the development build is not optimized.
To create a production build, use [npx react-build](#)

1: node

All the text on the browser is coming from here: App component

```
1 .App {  
2   text-align: center;  
3 }  
4  
5 .App-logo {  
6   animation: App-logo-spin infinite 20s linear;  
7   height: 80px;  
8 }  
9  
10 .App-header {  
11   background-color: #222;  
12   height: 150px;  
13   padding: 20px;  
14   color: white;  
15 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Compiled successfully!

You can now view myapp in the browser.

Local: http://localhost:3000/
On Your Network: http://192.168.1.73:3000/

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

Lets delete the css file. We do not going to show that default text on browser like its logo

The screenshot shows a Visual Studio Code interface with the following details:

- EXPLORER:** Shows the project structure:
 - OPEN EDITORS: App.js
 - REACT-REDUX-COMPLETE
 - myapp
 - node_modules
 - public
 - favicon.ico
 - index.html
 - manifest.json
 - src
 - App.js
 - App.test.js
 - index.css
 - index.js
 - logo.svg
 - registerServiceWorker.js
 - .gitignore
 - package-lock.json
 - package.json
 - README.md- EDITOR:** App.js code:

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
    );
  }
}

export default App;
```
- TERMINAL:** Shows the output of a command:

```
Failed to compile.

./src/App.css
Module build failed: Error: ENOENT: no such file or directory, open 'C:\Users\Shaun\Documents\Tutorials\Recording\react-redux-complete\myapp\src\App.css'
```

Also delete the reference of the previous css file

The screenshot shows a Visual Studio Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure with files like `App.js`, `App.test.js`, `index.css`, etc., under the `myapp` folder.
- EDITOR**: The `App.test.js` file is open, containing a Jest test for the `App` component.
- TERMINAL**: Shows the output of a build command, indicating a failure due to a missing file (`./src/App.css`).

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import App from './App';

4
5 it('renders without crashing', () => {
6   const div = document.createElement('div');
7   ReactDOM.render(<App />, div);
8   ReactDOM.unmountComponentAtNode(div);
9 });
10

Failed to compile.

./src/App.css
Module build failed: Error: ENOENT: no such file or directory, open 'C:\Users\Shaun\Documents\Tutorials\Recording\react-redux-complete\myapp\src\App.css'
```

Also delete the test file. No need at this time

The screenshot shows a Visual Studio Code interface with the following details:

- Explorer:** Shows the project structure under "REACT-REDUX-COMPLETE".
 - myapp:** Contains node_modules, public (with favicon.ico, index.html), and src (with App.js, index.css, index.js, logo.svg, registerServiceWorker.js).
 - index.css:** Selected file in the Explorer.
- Editor:** Displays the contents of index.css:

```
1 body {  
2   margin: 0;  
3   padding: 0;  
4   font-family: sans-serif;  
5 }  
6
```
- Terminal:** Shows the output of a build command:

```
Failed to compile.  
  
./src/App.css  
Module build failed: Error: ENOENT: no such file or directory, open 'C:\Users\Shaun\Documents\Tutorials\Recording\react-redux-complete\myapp\src\App.css'
```

So now basically we have
App.js file which is our component.

Index.css which is a global css file of our project we can also add more css in this file.

Index.js file where we render the component to the DOM.

Logo.svg delete this logo too we have no further need.

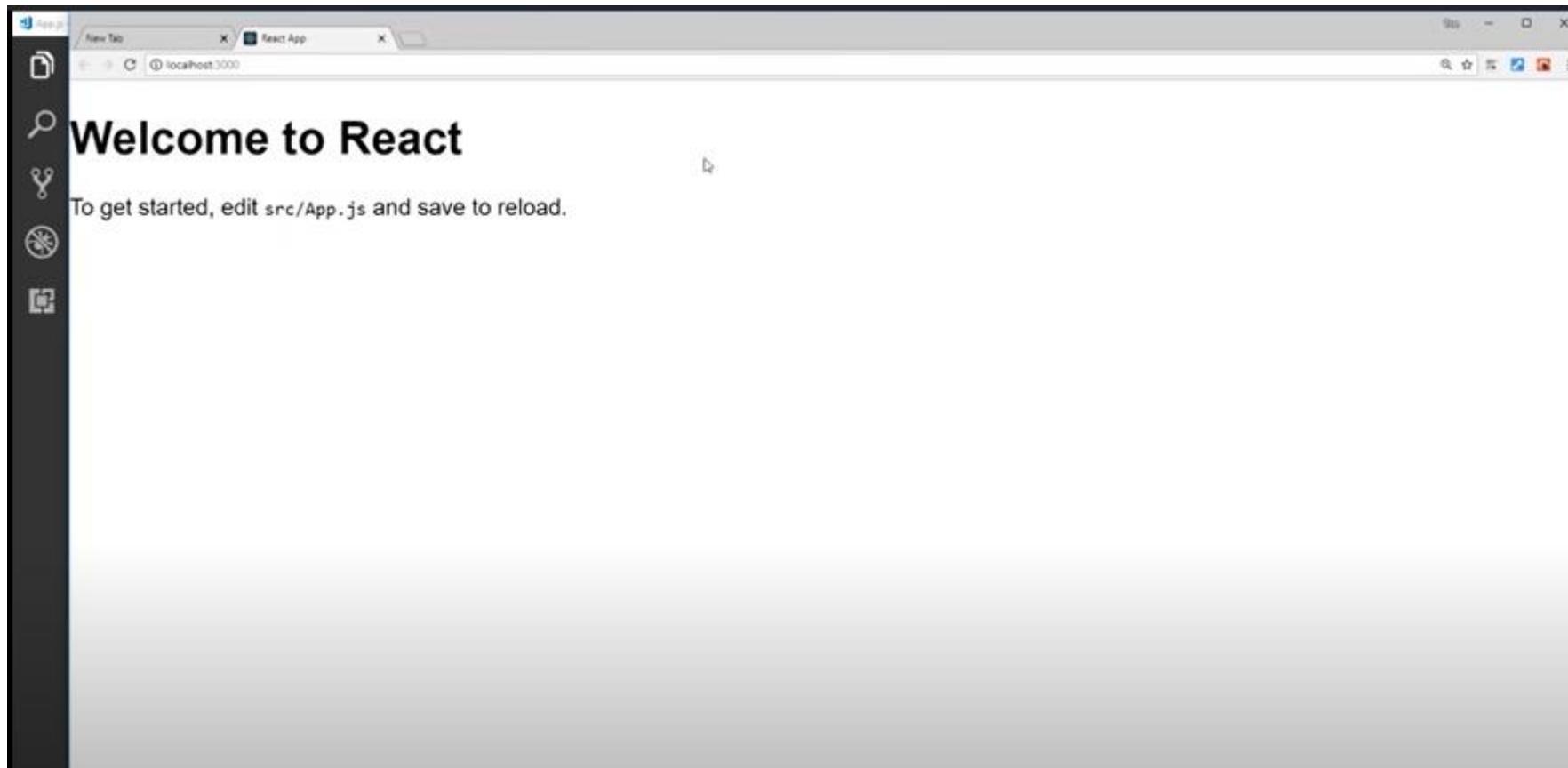
registerServiceWorker.js file which manage cache for best user experience leave it...

The screenshot shows a Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "REACT-REDUX-COMPLETE". The "src" folder contains "App.js", "index.css", "index.js", and "registerServiceWorker.js".
- Code Editor:** Displays the content of "App.js". The code imports React and a logo from a file named "logo.svg". It defines a class "App" that extends "Component" and has a "render" method. The render method creates a "div" with class "App", which contains a "header" with class "App-header", an "img" tag with class "App-logo", and a "h1" with class "App-title" containing the text "Welcome to React". Below the header, there is a "p" tag with class "App-intro" containing the text "To get started, edit <code>src/App.js</code> and save to reload.".
- Terminal:** Shows the output of a build command. It starts with "Failed to compile." followed by the command "./src/App.css". Then it shows an error message: "Module build failed: Error: ENOENT: no such file or directory, open 'C:\Users\Shaun\Documents\Tutorials\Recording\react-redux-complete\myapp\src\App.css'".

Delete the import logo line we have no more this logo and also delete the image tag where we are putting logo.

Now in the browser we will have:



Simple Right?

Delete the rest too

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** App.js - react-redux-complete - Visual Studio Code
- Explorer:** Shows the project structure:
 - myapp
 - node_modules
 - public
 - favicon.ico
 - index.html
 - manifest.json
 - src
 - App.js
 - index.css
 - index.js
 - registerServiceWorker.js
 - .gitignore
 - package-lock.json
 - package.json
 - README.md
- Editor:** The App.js file is open, showing the following code:

```
1 import React, { Component } from 'react';
2
3 class App extends Component {
4   render() {
5     return (
6       <div className="App">
7         <header className="App-header">
8           <h1 className="App-title">Welcome to React</h1>
9         </header>
10        <p className="App-intro">
11          To get started, edit <code>src/App.js</code> and save to reload.
12        </p>
13      </div>
14    );
15  }
}
```

A red rectangular highlight covers the entire content of the editor area.
- Terminal:** Shows the command "1: node" followed by a series of small icons.
- Bottom Status Bar:** Displays the message "Compiled successfully!"
- Bottom Output Panel:** Displays the local and network URLs for the application: "Local: http://localhost:3000/" and "On Your Network: http://192.168.1.73:3000/". It also includes a note: "Note that the development build is not optimized. To create a production build, use `npm run build`".

Add simple template

The screenshot shows a Visual Studio Code interface with the following details:

- EXPLORER:** Shows the project structure:
 - OPEN EDITORS: App.js (myapp/src)
 - REACT-REDUX-COMPLETE: myapp, node_modules, public (favicon.ico, index.html), manifest.json, src (App.js, index.css, index.js, registerServiceWorker.js), .gitignore, package-lock.json, package.json, README.md
- EDITOR:** App.js code:

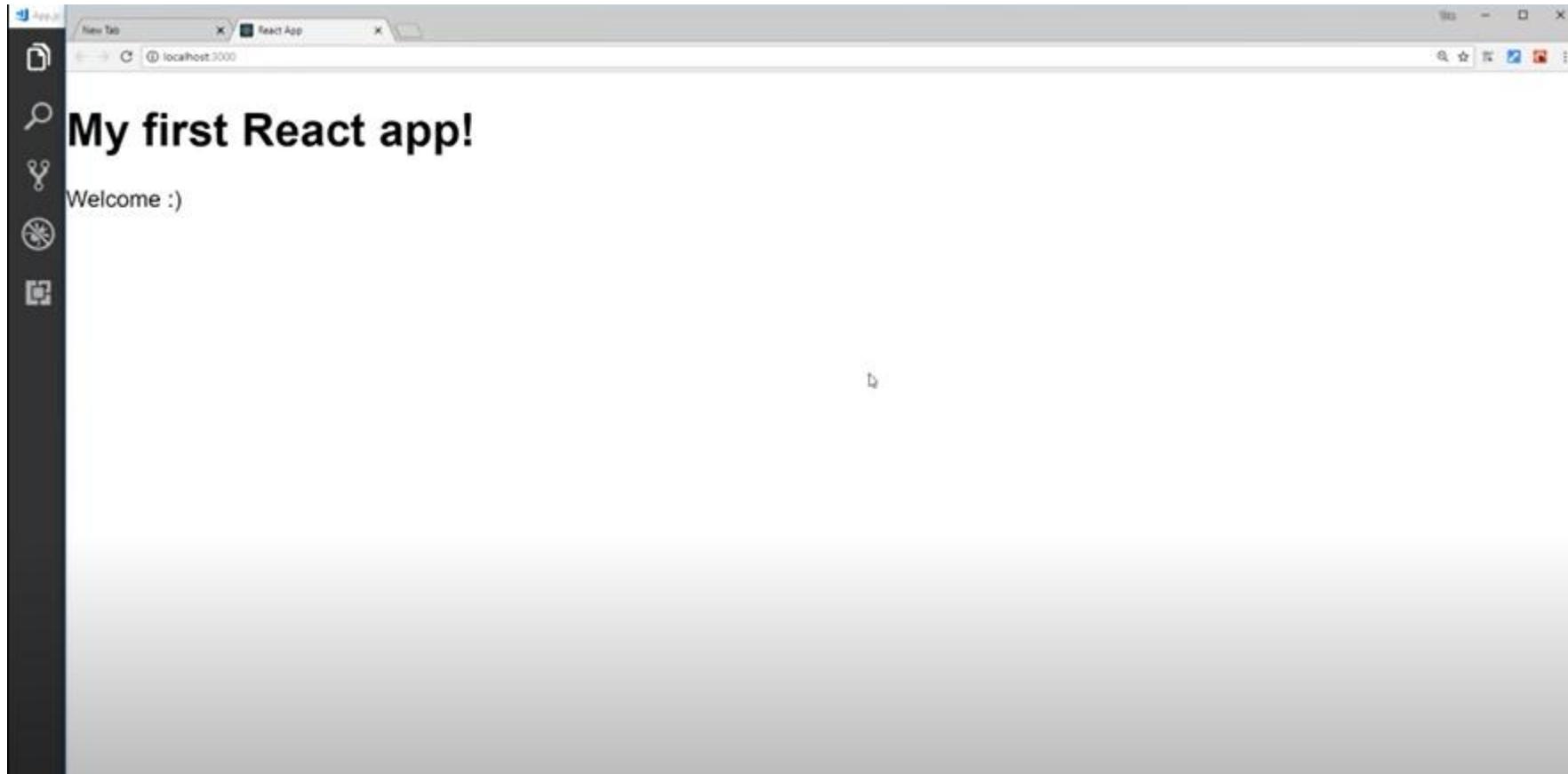
```
1 import React, { Component } from 'react';
2
3 class App extends Component {
4   render() {
5     return (
6       <div className="App">
7         <h1>My first React app!</h1>
8         <p>Welcome :)</p>
9       </div>
10    );
11  }
12}
13
14 export default App;
```
- TERMINAL:** Output of the build process:

```
Compiled successfully!

You can now view myapp in the browser.

Local:          http://localhost:3000/
On Your Network: http://192.168.1.73:3000/

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



When you hit Ctrl +S key it will automatically updated on the browser

SINGLE PAGE APPS

Single Page Apps

- React apps are typically SPA's**
- Only ever one HTML page served to the browser**
- React then controls what a user sees on that page**

Multi Page Apps

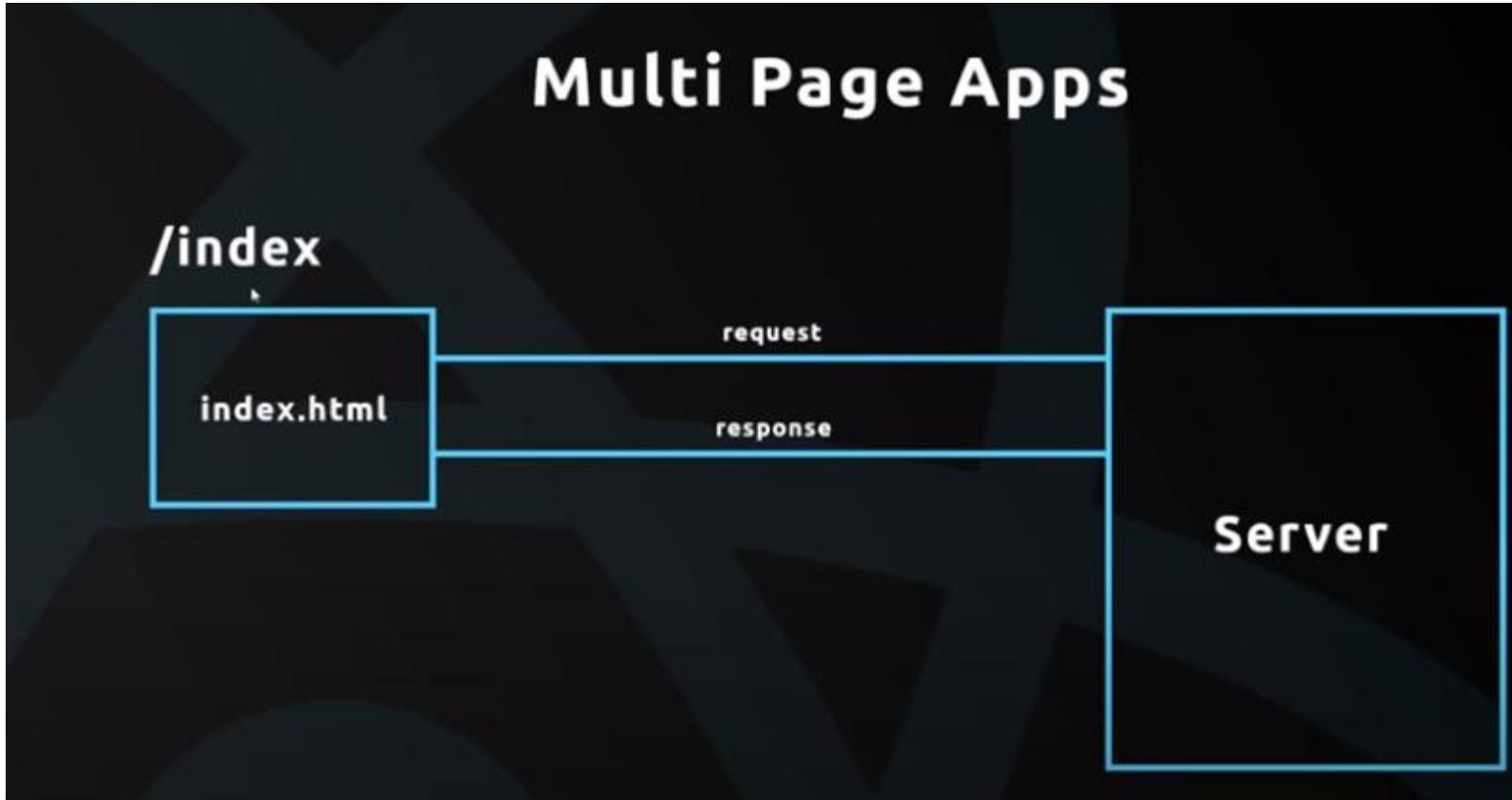
/index

index.html

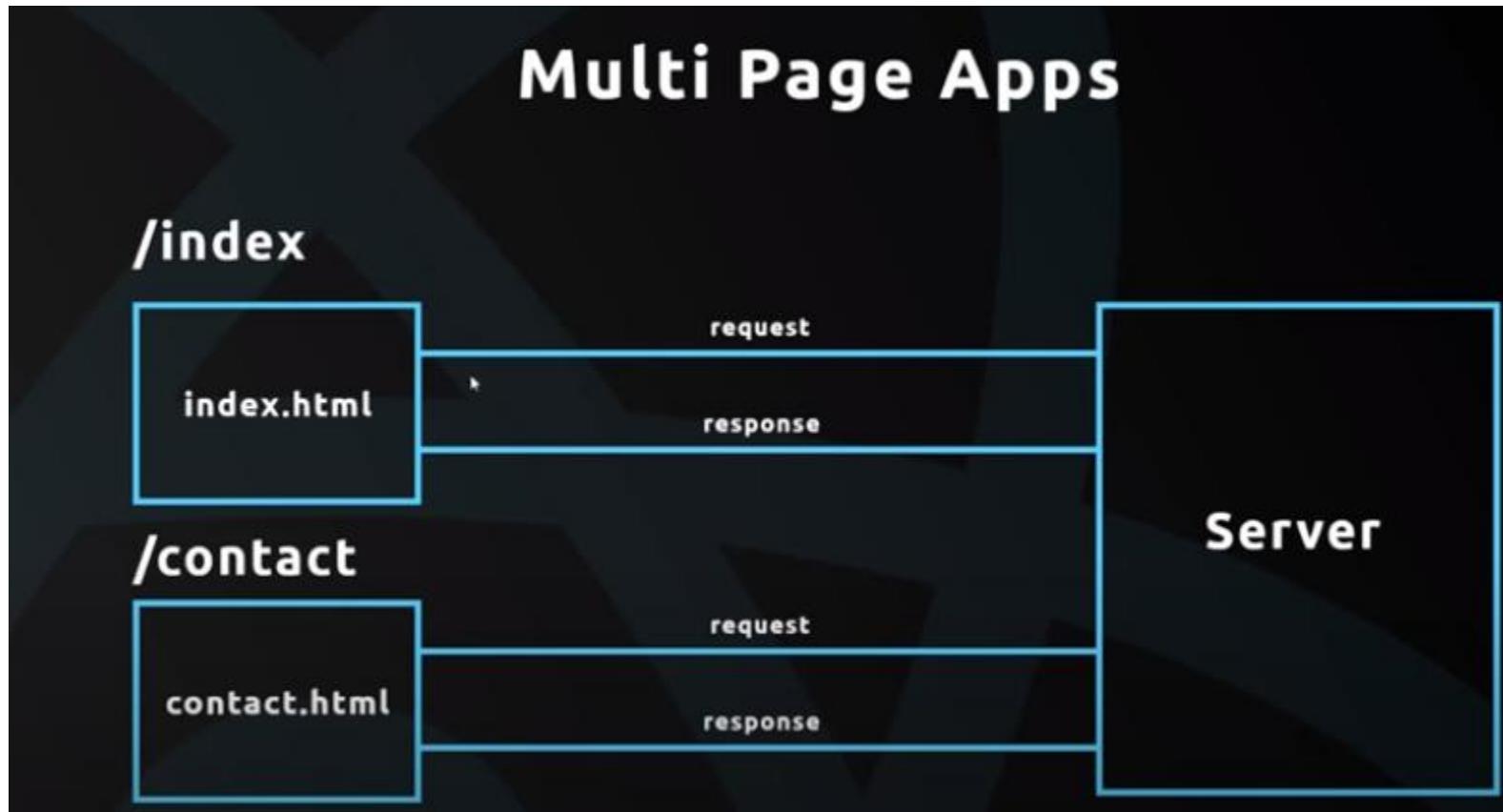
request

response

Server



Multi Page Apps



Single Page Apps

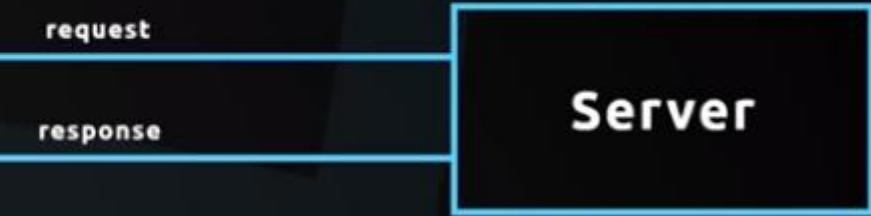
/index



request

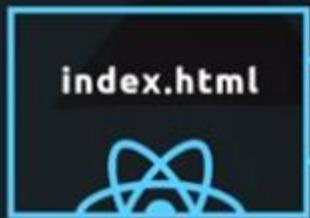
response

Server



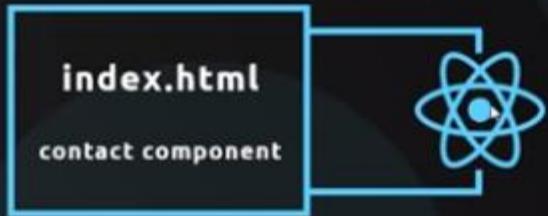
Single Page Apps

/index



Server

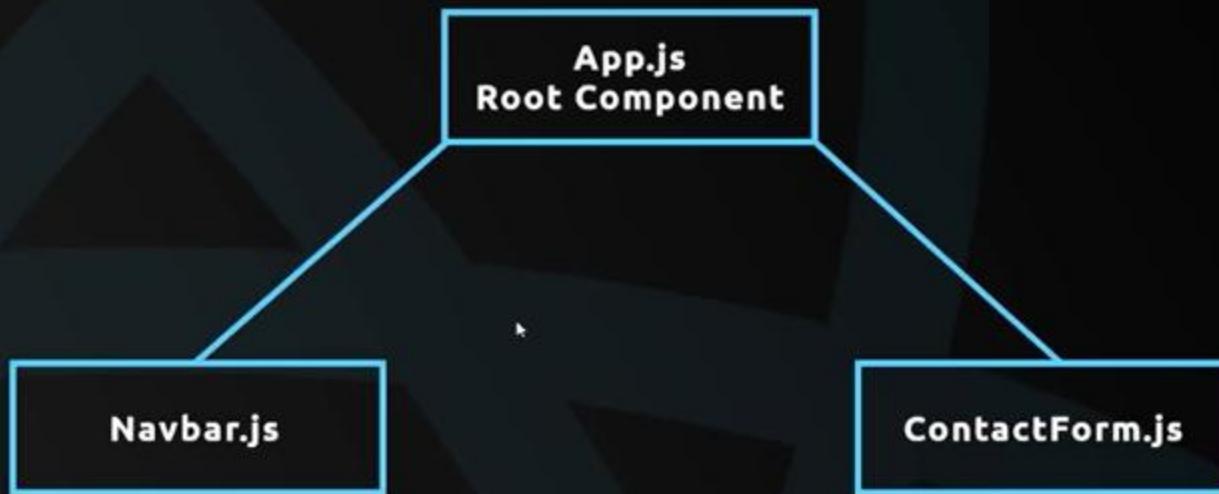
/contact



NESTING COMPONENTS

In React root component is App.js we can nesting more components in it

Nesting Components



The screenshot shows a Visual Studio Code interface with a dark theme. The left sidebar displays the project structure:

- OPEN EDITORS**: Shows `App.js` and `Ninjas.js`.
- REACT-REDUX-COMPLETE**: Shows the project root with subfolders `myapp`, `node_modules`, and `public` containing files like `favicon.ico`, `index.html`, and `manifest.json`. It also shows the `src` folder containing `App.js`, `index.css`, `index.js`, `Ninjas.js`, and `registerServiceWorker.js`, along with files `.gitignore`, `package-lock.json`, `package.json`, and `README.md`.

The main editor area shows the code for `Ninjas.js`:

```
1 import React, { Component } from 'react';
2
3 class Ninjas extends Component{
4   render(){
5     return(
6       <div className="ninja">
7         <div>Name: Ryu</div>
8         <div>Age: 30</div>
9         <div>Belt: Black</div>
10      </div>
11    )
12  }
13 }
14
15 export default Ninjas;
```

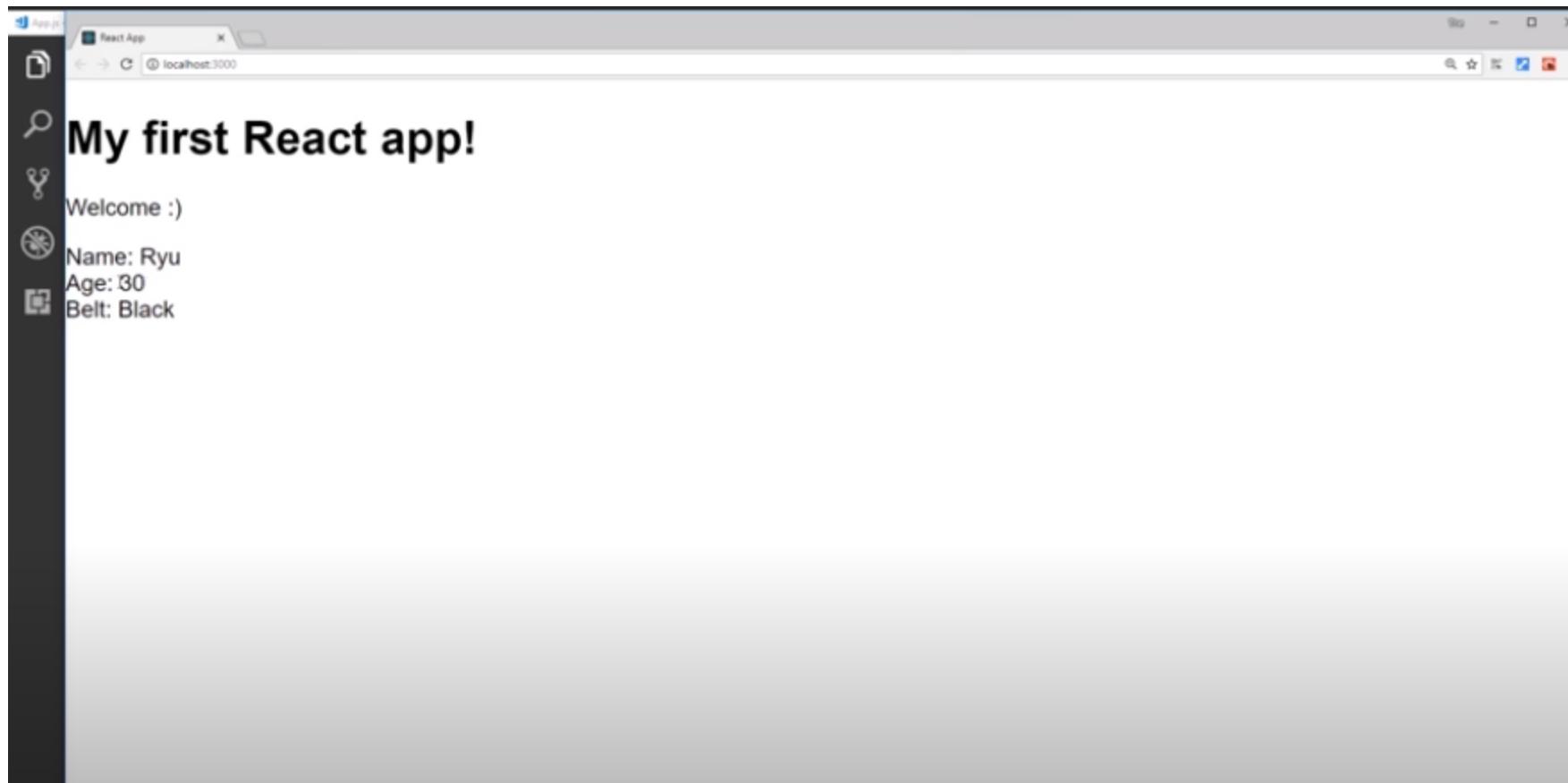
Creating another component with name `Ninjas` in `src` directory

The screenshot shows a Visual Studio Code interface with the following details:

- Title Bar:** App.js - react-redux-complete - Visual Studio Code
- Explorer View:** Shows the project structure:
 - OPEN EDITORS: App.js myappsrc, Ninjas.js myappsrc
 - REACT-REDUX-COMPLETE: myapp (node_modules, public: favicon.ico, index.html, manifest.json), src (App.js, index.css, index.js, Ninjas.js, registerServiceWorker.js), .gitignore, package-lock.json, package.json, README.md
- Editor View:** The App.js file is open, displaying the following code:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3
4 class App extends Component {
5   render() {
6     return (
7       <div className="App">
8         <h1>My first React app!</h1>
9         <p>Welcome :)</p>
10        <Ninjas />
11      </div>
12    );
13  }
14}
15
16 export default App;
17
```

Injecting component



PROPS

The screenshot shows the Visual Studio Code interface with the following details:

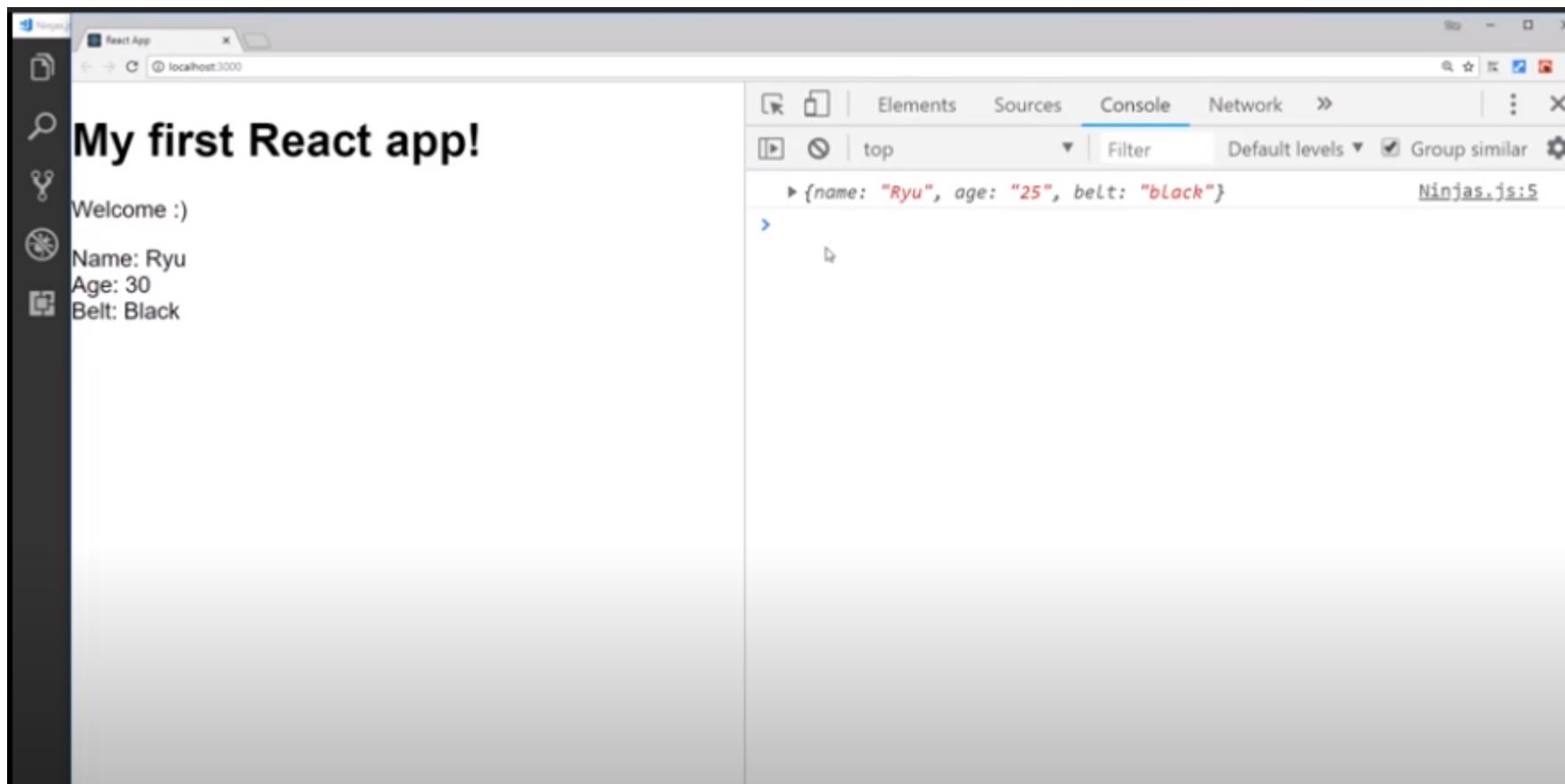
- Title Bar:** App.js - react-redux-complete - Visual Studio Code
- Sidebar:** Shows the project structure with "App.js" selected.
- Code Editor:** Displays the following code in App.js:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3
4 class App extends Component {
5   render() {
6     return (
7       <div className="App">
8         <h1>My first React app!</h1>
9         <p>Welcome :)</p>
10        <Ninjas name="Ryu" age="25" belt="black"/>
11      </div>
12    );
13  }
14}
15
16 export default App;
17
```

Passing data to component (Parent to child)

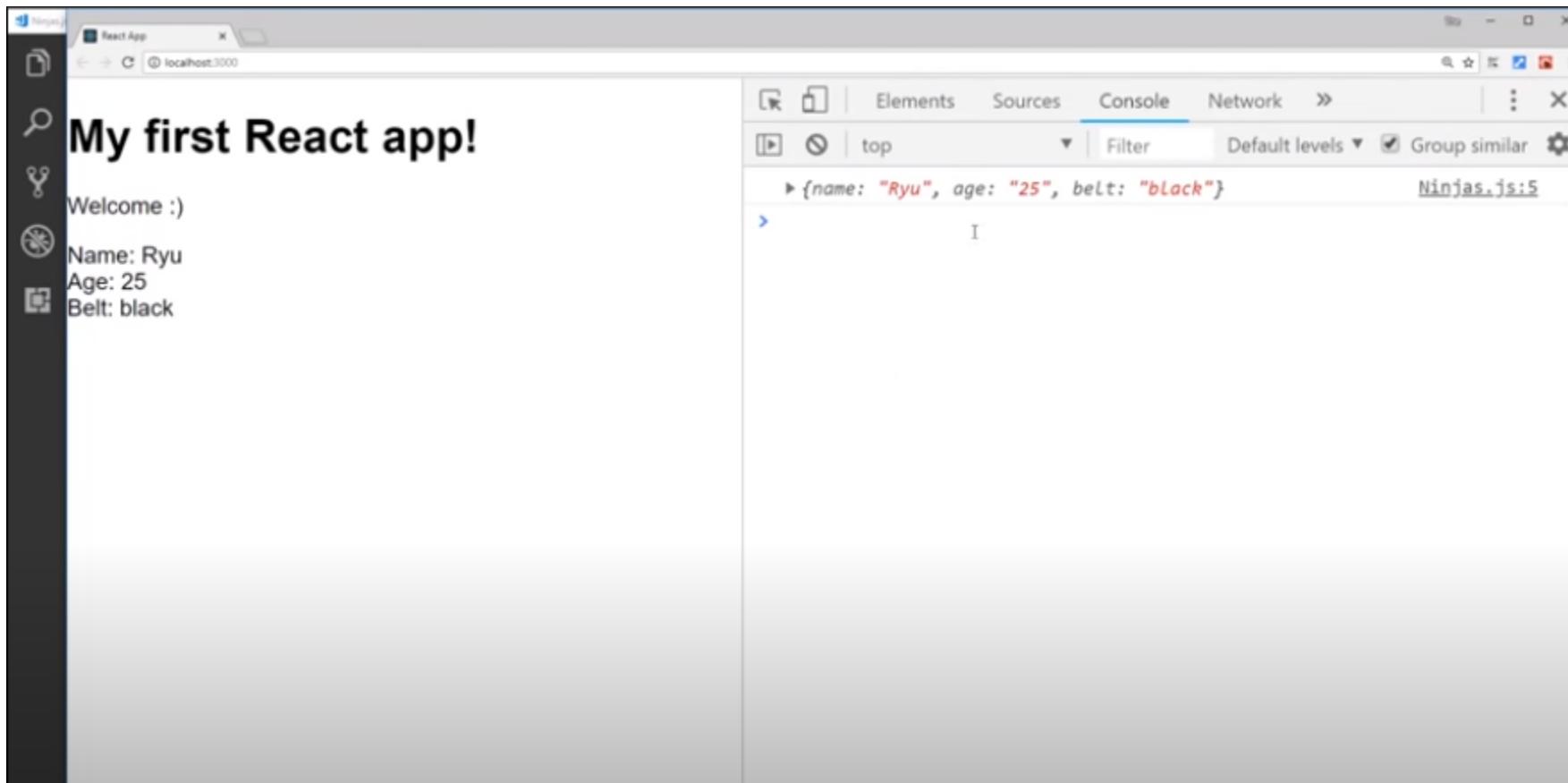
A screenshot of the Visual Studio Code interface showing a file named 'Ninjas.js'. The code is a simple React component that logs its props to the console and returns a div with three child divs containing 'Name: Ryu', 'Age: 30', and 'Belt: Black' respectively. The code is written in a dark-themed editor.

```
1 import React, { Component } from 'react';
2
3 class Ninjas extends Component{
4   render(){
5     console.log(this.props);
6     return(
7       <div className="ninja">
8         <div>Name: Ryu</div>
9         <div>Age: 30</div>
10        <div>Belt: Black</div>
11      </div>
12    )
13  }
14}
15
16 export default Ninjas
```



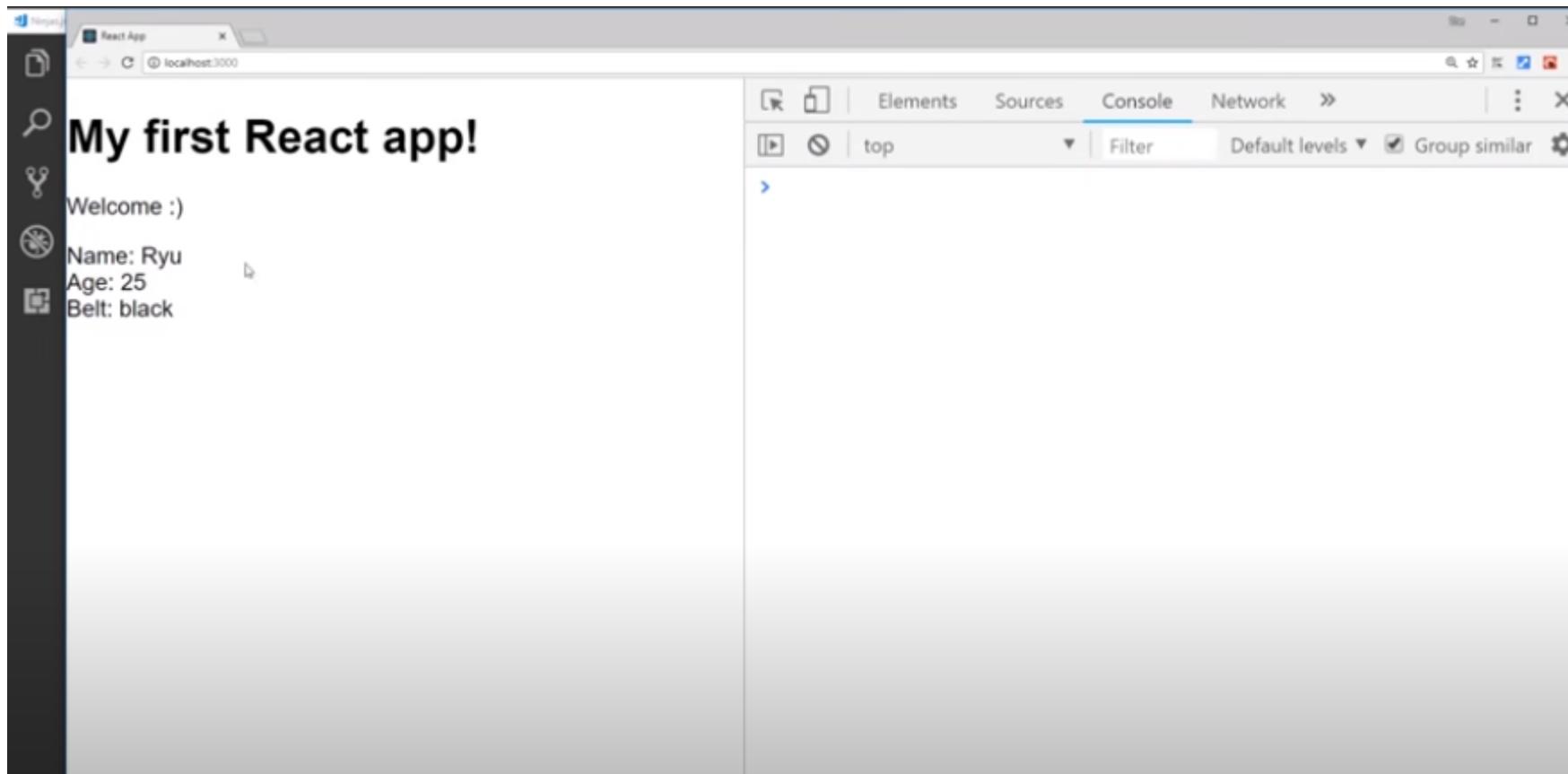
A screenshot of the Visual Studio Code interface showing a file named 'Ninjas.js'. The code is a simple React component that logs its props to the console and displays three pieces of information: name, age, and belt. The code uses ES6 syntax, including a class-based component and template literals.

```
1 import React, { Component } from 'react';
2
3 class Ninjas extends Component{
4   render(){
5     console.log(this.props);
6     return(
7       <div className="ninja">
8         <div>Name: { this.props.name }</div>
9         <div>Age: { this.props.age }</div>
10        <div>Belt: { this.props.belt }</div>
11      </div>
12    )
13  }
14}
15
16 export default Ninjas
```

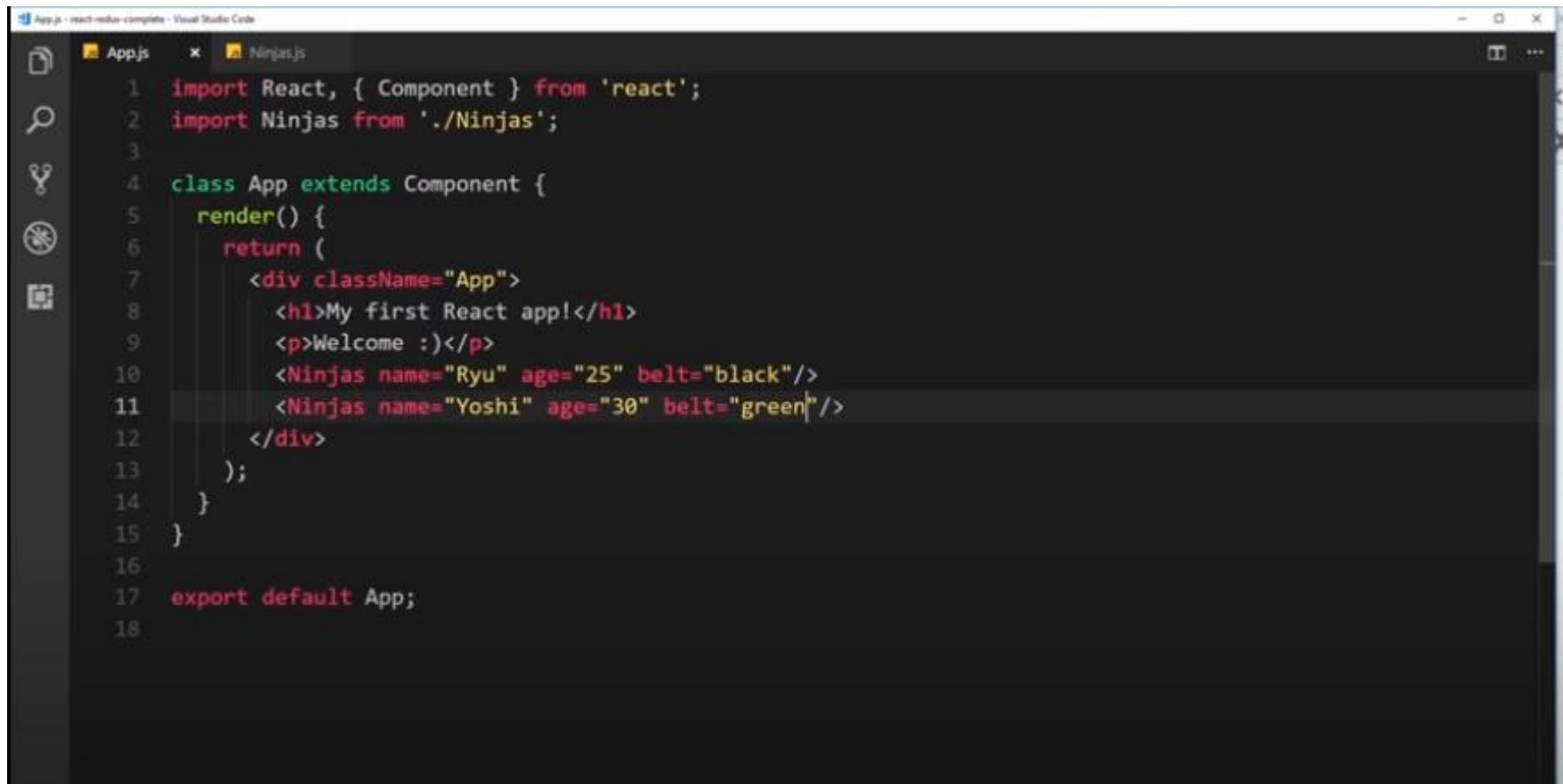


A screenshot of the Visual Studio Code interface showing a file named 'Ninjas.js'. The code is a simple React component that takes props and logs them to the console. It then creates a div with the class 'ninja' containing three child divs with the text 'Name: { name }', 'Age: { age }', and 'Belt: { belt }' respectively.

```
1 import React, { Component } from 'react';
2
3 class Ninjas extends Component{
4   render(){
5     // console.log(this.props);
6     const { name, age, belt } = this.props;
7     return(
8       <div className="ninja">
9         <div>Name: { name }</div>
10        <div>Age: { age }</div>
11        <div>Belt: { belt }</div>
12      </div>
13    )
14  }
15}
16
17 export default Ninjas
```



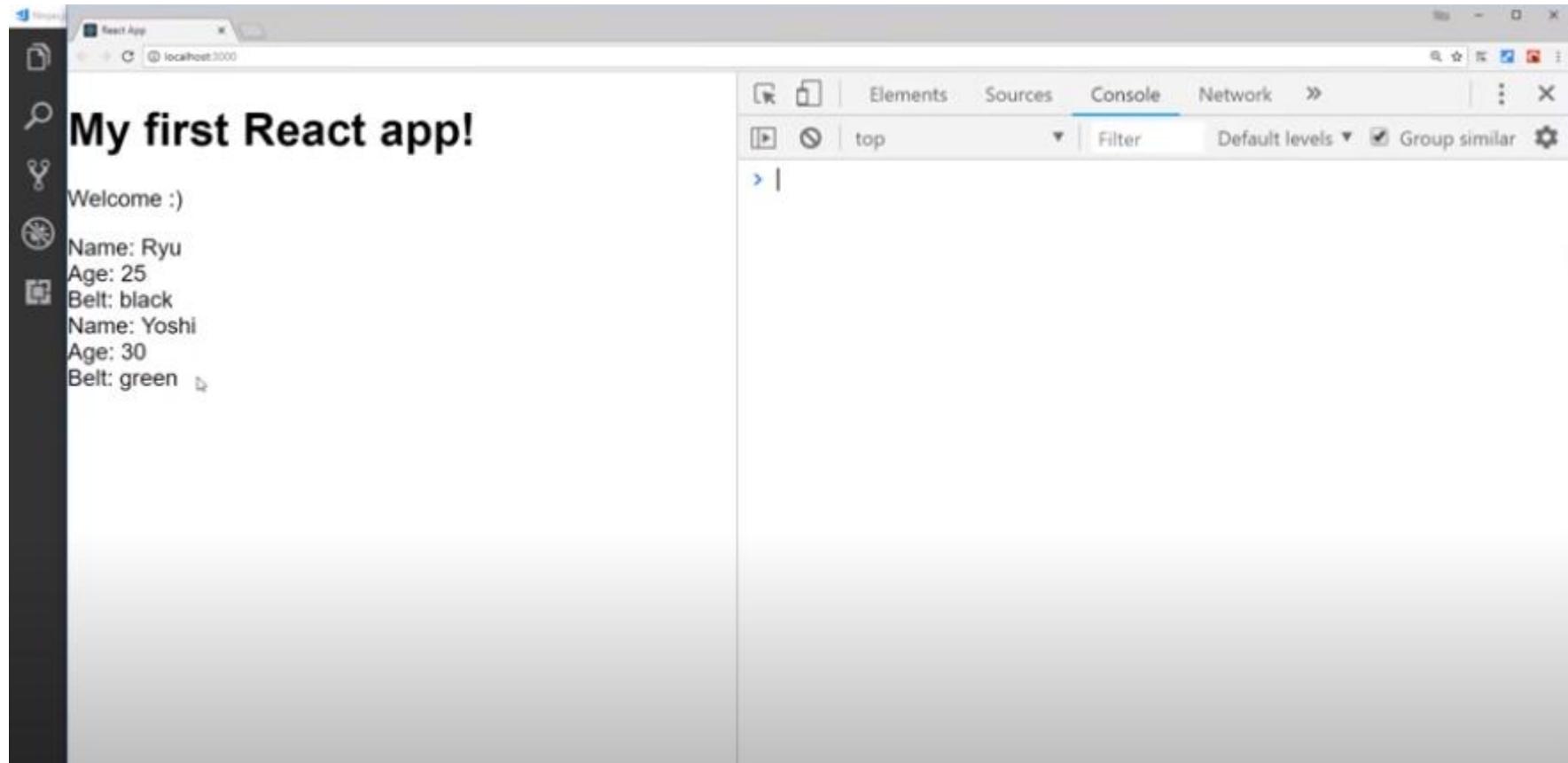
Still same output



The screenshot shows the Visual Studio Code interface with the title bar "App.js - react-movie-complete - Visual Studio Code". The left sidebar contains icons for file operations like Open, Save, Find, and Refresh. The main editor area displays the following code:

```
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3
4 class App extends Component {
5   render() {
6     return (
7       <div className="App">
8         <h1>My first React app!</h1>
9         <p>Welcome :)</p>
10        <Ninjas name="Ryu" age="25" belt="black"/>
11        <Ninjas name="Yoshi" age="30" belt="green"/>
12      </div>
13    );
14  }
15}
16
17 export default App;
18
```

Use same component with diff values: dynamic behavior



What it looks on browser

OUTPUTTING LISTS

A screenshot of the Visual Studio Code interface showing the file `App.js`. The code defines a class-based component named `App` that imports `React` and `Ninjas` from external modules. It initializes state with an array of three ninjas, each having a name, age, belt color, and ID. The `render` method returns a `div` containing an `h1` and a `p` element, followed by a `Ninjas` component.

```
App.js  ●  Ninja.js
1 import React, { Component } from 'react';
2 import Ninjas from './Ninjas';
3
4 class App extends Component {
5   state = {
6     ninjas : [
7       { name: 'Ryu', age: 30, belt: 'black', id: 1 },
8       { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
9       { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
10    ]
11  }
12  render() {
13    return (
14      <div className="App">
15        <h1>My first React app!</h1>
16        <p>Welcome :)</p>
17        <Ninjas />
18      </div>
19    );
20  }
21}
22
```

```
App.js • 1 App.js • 2 Ninjas.js
3
4  class App extends Component {
5    state = {
6      ninjas: [
7        { name: 'Ryu', age: 30, belt: 'black', id: 1 },
8        { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
9        { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
10      ]
11    }
12    render() {
13      return (
14        <div className="App">
15          <h1>My first React app!</h1>
16          <p>Welcome :)</p>
17          <Ninjas ninjas={this.state.ninjas} />
18        </div>
19      );
20    }
21  }
22
23  export default App;
24
```

A screenshot of the Visual Studio Code interface, showing a file named 'Ninjas.js' in the center editor area. The code is a simple React component:

```
1 import React, { Component } from 'react';
2
3 class Ninjas extends Component{
4   render(){
5     const { ninjas } = this.props;
6     return(
7       <div className="ninja">
8         <div>Name: { name }</div>
9         <div>Age: { age }</div>
10        <div>Belt: { belt }</div>
11      </div>
12    )
13  }
14}
15
16 export default Ninjas
```

The code uses ES6 syntax, including imports, classes, and arrow functions. The 'ninjas' prop is destructured within the render method. The component returns a single 'div' element with a 'ninja' class, which contains three child 'div' elements with dynamic content based on the 'name', 'age', and 'belt' props.

A screenshot of the Visual Studio Code interface showing a file named 'Ninjas.js'. The code is a class-based React component named 'Ninjas' that takes an array of ninjas as props. It maps over the array to render each ninja's name, age, and belt color in separate

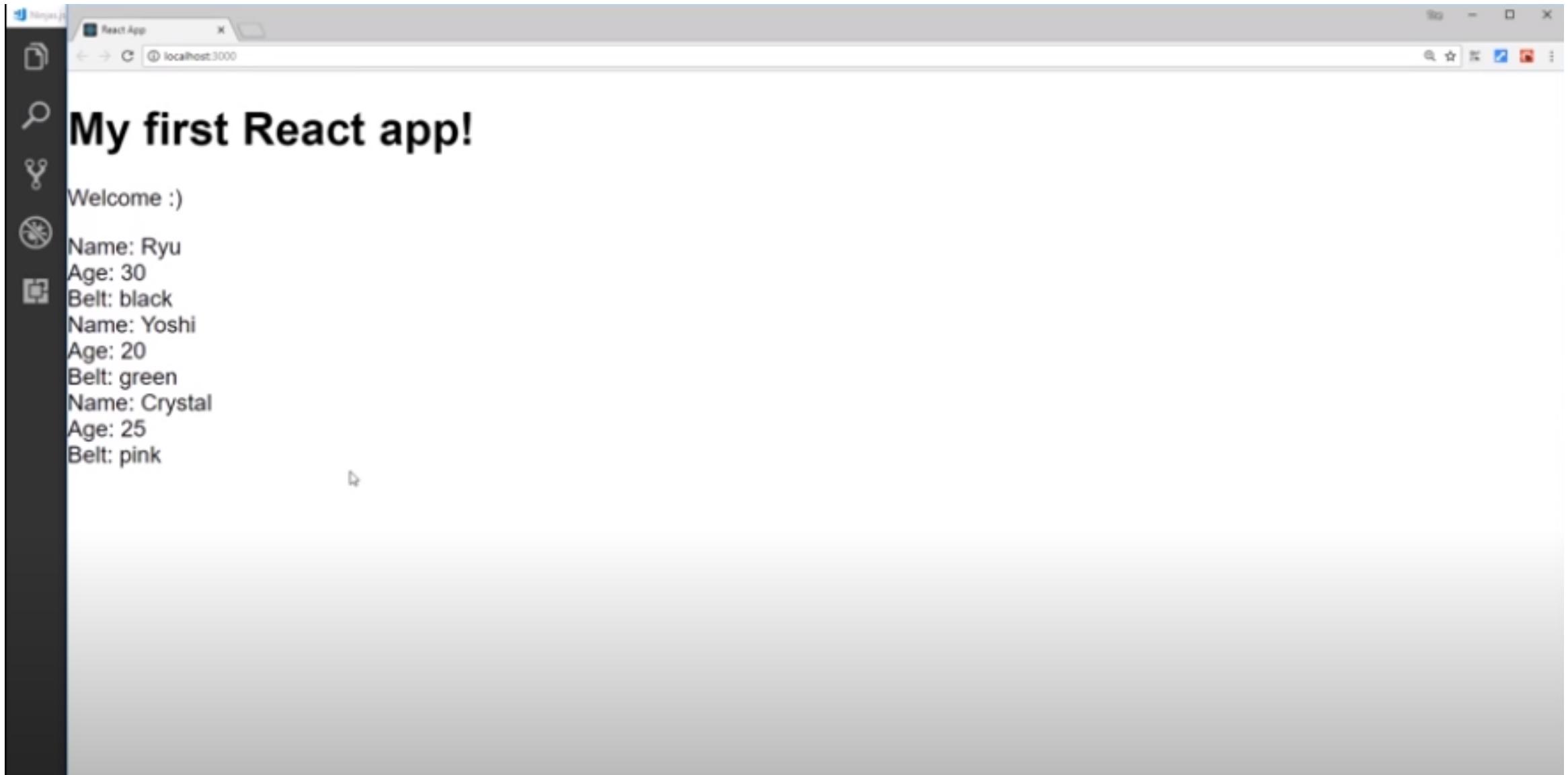
elements.

```
1 import React, { Component } from 'react';
2
3 class Ninjas extends Component{
4     render(){
5         const { ninjas } = this.props;
6         const ninjaList = ninjas.map(ninja => {
7             return (
8                 <div className="ninja">
9                     <div>Name: { name }</div>
10                    <div>Age: { age }</div>
11                    <div>Belt: { belt }</div>
12                </div>
13            )
14        })
15        return(
16            )
17    }
18 }
19 }
20
21 export default Ninjas
```

A screenshot of the Visual Studio Code interface, showing a file named `Ninjas.js` with code completion enabled. The code defines a class-based React component named `Ninjas`. The `render` method maps over the `ninjas` prop, creating a list of ninjas. The code completion feature is active, with a red box highlighting the `ninja.name` part of the first `<div>` element. The code is as follows:

```
1 import React, { Component } from 'react';
2
3 class Ninjas extends Component{
4   render(){
5     const { ninjas } = this.props;
6     const ninjaList = ninjas.map(ninja => {
7       return (
8         <div className="ninja">
9           <div>Name: { ninja.name }</div>
10          <div>Age: { ninja.age }</div>
11          <div>Belt: { ninja.belt }</div>
12        </div>
13      )
14    })
15    return(
16      )
17    }
18  }
19 }
20
21 export default Ninjas
```

```
App.js    Ninja.js •  
1 import React, { Component } from 'react';  
2  
3 class Ninjas extends Component{  
4   render(){  
5     const { ninjas } = this.props;  
6     const ninjaList = ninjas.map(ninja => {  
7       return (  
8         <div className="ninja">  
9           <div>Name: { ninja.name }</div>  
10          <div>Age: { ninja.age }</div>  
11          <div>Belt: { ninja.belt }</div>  
12        </div>  
13      )  
14    })  
15    return(  
16      <div className="ninja-list">  
17        { ninjaList }  
18      </div>  
19    )  
20  }  
21}  
22
```



My first React app!

Welcome :)

Name: Ryu

Age: 30

Belt: black

Name: Yoshi

Age: 20

Belt: green

Name: Crystal

Age: 25

Belt: pink

A screenshot of a web browser window displaying a React application. The application has a title "My first React app!" and a list of ninjas with their details: Name, Age, and Belt. The browser's developer tools are open, specifically the "Console" tab, which shows a warning message about the "key" prop.

My first React app!

Welcome :)

Name: Ryu
Age: 30
Belt: black

Name: Yoshi
Age: 20
Belt: green

Name: Crystal
Age: 25
Belt: pink

Elements Sources Console Network > 1 ⓘ

top Filter Default levels Group similar

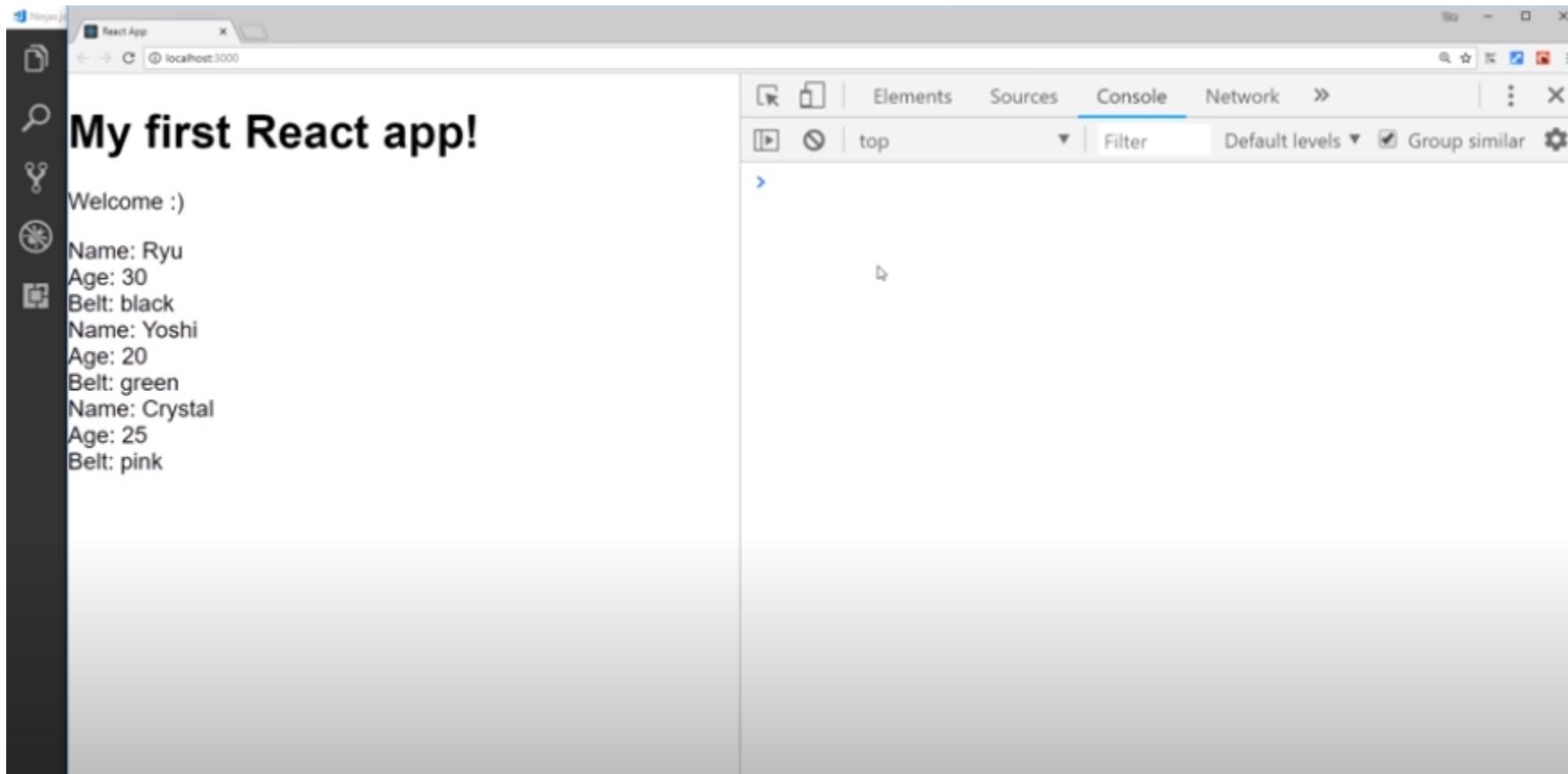
Warning: Each child in an array or iterator should have a unique "key" prop. [index.js:2178](#)

Check the render method of 'Ninjas'. See <https://fb.me/react-warning-keys> for more information.

in div (at Ninjas.js:8)
in Ninjas (at App.js:17)
in div (at App.js:14)
in App (at index.js:7)

The screenshot shows a Visual Studio Code interface with two tabs open: "App.js" and "Ninjas.js". The "Ninjas.js" tab is currently active, displaying the following code:

```
1 import React, { Component } from 'react';
2
3 class Ninjas extends Component{
4   render(){
5     const { ninjas } = this.props;
6     const ninjaList = ninjas.map(ninja => {
7       return (
8         <div className="ninja" key={ninja.id}>
9           <div>Name: { ninja.name }</div>
10          <div>Age: { ninja.age }</div>
11          <div>Belt: { ninja.belt }</div>
12        </div>
13      )
14    })
15    return(
16      <div className="ninja-list">
17        { ninjaList }
18      </div>
19    )
20  }
21}
22
```



STATELESS COMPONENTS

Container vs UI Components

Container Components

- Contain state
- Contain lifecycle hooks
- Not concerned with UI
- Use classes to create

UI Components

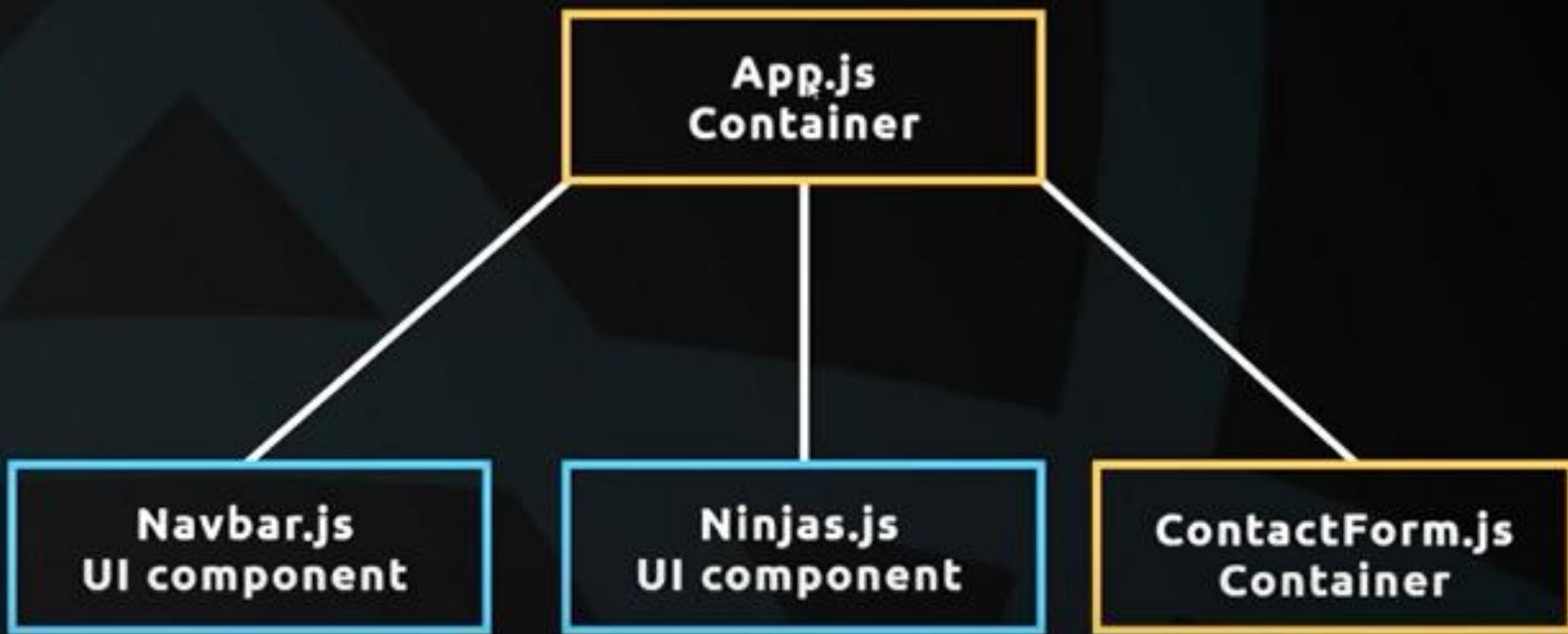
- Don't contain state
- Receive data from props
- Only concerned with UI
- Use functions to create

Container components are not concerned with UI or look of app

Container Comp.=Statefull Comp.=Class based Comp.

UI Comp.=Stateless Comp.=Functional Comp.

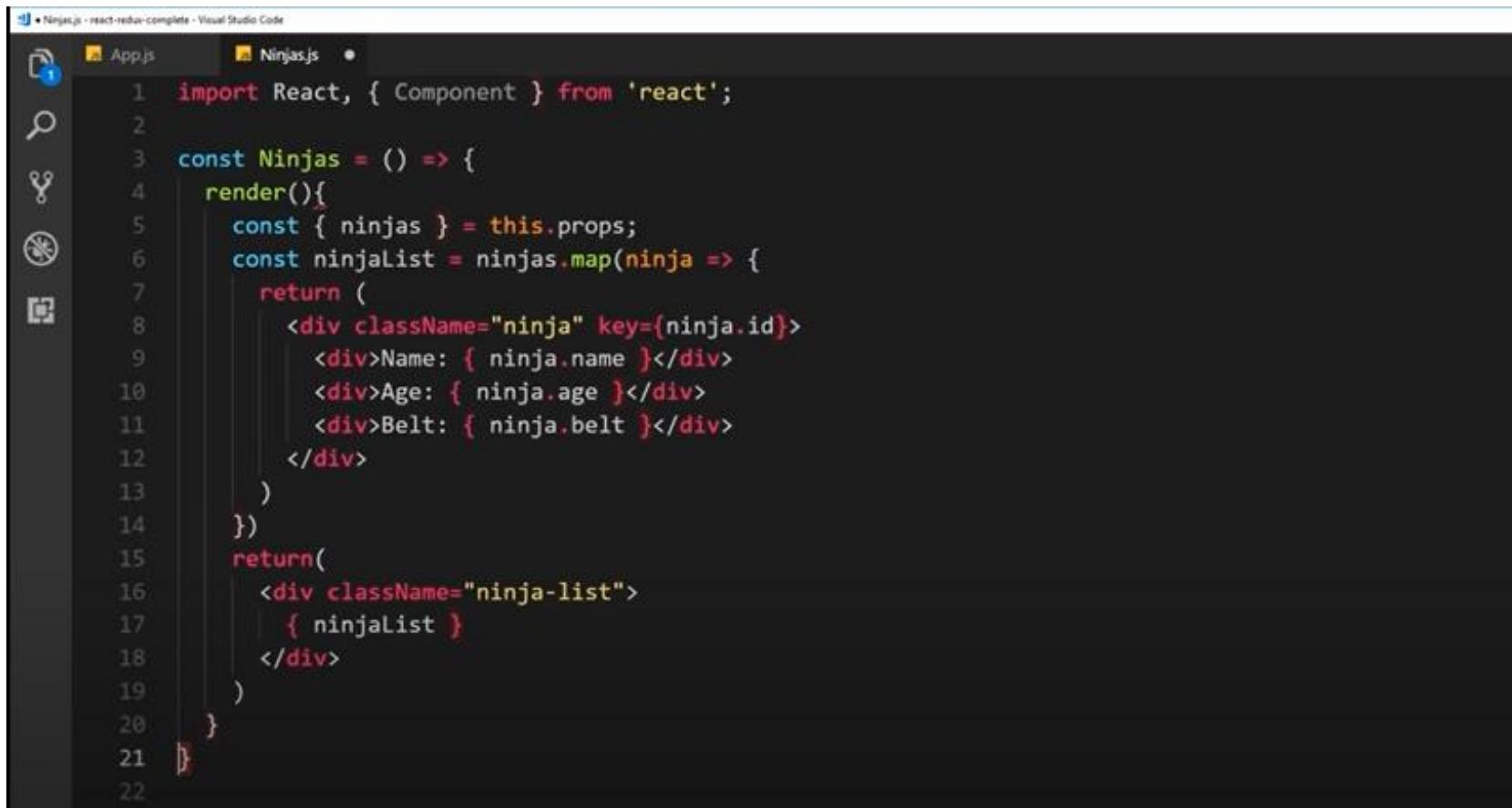
Container vs UI Components



The image shows a screenshot of a code editor interface. On the left, there is a sidebar with several icons: a file, a search magnifying glass, a circular arrow, a square with a circle, and a double square. Next to these icons are the file names: "App.js" and "Ninjas.js".

The main area displays the content of the "Ninjas.js" file. The code is written in JavaScript and uses React syntax. It defines a class named "Ninjas" which extends "Component". The "render" method creates a list of ninjas by mapping over the "ninja" prop and rendering each one as a "ninja" div. Each "ninja" div contains "Name", "Age", and "Belt" information.

```
1 import React, { Component } from 'react';
2
3 class Ninjas extends Component{
4   render(){
5     const { ninjas } = this.props;
6     const ninjaList = ninjas.map(ninja => {
7       return (
8         <div className="ninja" key={ninja.id}>
9           <div>Name: { ninja.name }</div>
10          <div>Age: { ninja.age }</div>
11          <div>Belt: { ninja.belt }</div>
12        </div>
13      )
14    })
15    return(
16      <div className="ninja-list">
17        { ninjaList }
18      </div>
19    )
20  }
21}
22
23 export default Ninjas
```



A screenshot of the Visual Studio Code interface. The title bar shows the path: 'Ninjas.js - react-redux-complete - Visual Studio Code'. The left sidebar has icons for file operations, search, and other settings. There are two tabs open: 'App.js' and 'Ninjas.js'. The 'App.js' tab is active and displays the following code:

```
1 import React, { Component } from 'react';
2
3 const Ninjas = () => {
4   render(){
5     const { ninjas } = this.props;
6     const ninjaList = ninjas.map(ninja => {
7       return (
8         <div className="ninja" key={ninja.id}>
9           <div>Name: { ninja.name }</div>
10          <div>Age: { ninja.age }</div>
11          <div>Belt: { ninja.belt }</div>
12        </div>
13      )
14    })
15    return(
16      <div className="ninja-list">
17        { ninjaList }
18      </div>
19    )
20  }
21}
22
```

A screenshot of the Visual Studio Code interface showing the `App.js` file open. The code defines a `Ninjas` component that maps over an array of `ninjas` to render individual `ninja` components.

```
1 import React, { Component } from 'react';
2
3 const Ninjas = () => {
4   | render(){
5     const { ninjas } = this.props;
6     const ninjaList = ninjas.map(ninja => {
7       return (
8         <div className="ninja" key={ninja.id}>
9           <div>Name: { ninja.name }</div>
10          <div>Age: { ninja.age }</div>
11          <div>Belt: { ninja.belt }</div>
12        </div>
13      )
14    })
15    return(
16      <div className="ninja-list">
17        { ninjaList }
18      </div>
19    )
20  }
21}
22}
```

A screenshot of the Visual Studio Code interface. The title bar shows the file path: 'Ninjas.js - react-redux-complete'. The left sidebar has icons for file, search, file history, and refresh. There are two tabs open: 'App.js' and 'Ninjas.js'. The 'Ninjas.js' tab is active and displays the following code:

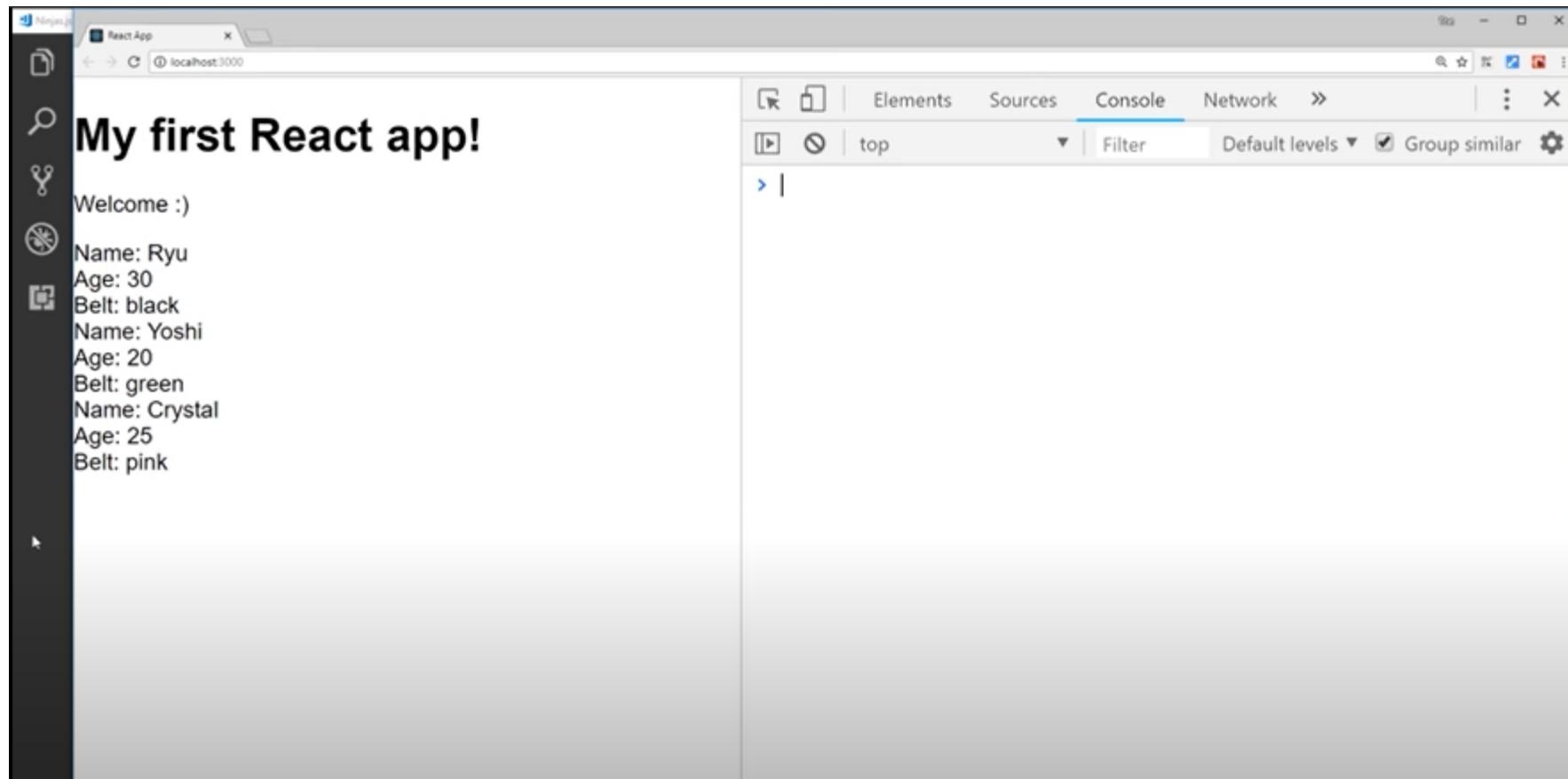
```
1 import React, { Component } from 'react';
2
3 const Ninjas = () => {
4   const { ninjas } = this.props;
5   const ninjaList = ninjas.map(ninja => {
6     return (
7       <div className="ninja" key={ninja.id}>
8         <div>Name: { ninja.name }</div>
9         <div>Age: { ninja.age }</div>
10        <div>Belt: { ninja.belt }</div>
11      </div>
12    )
13  })
14  return(
15    <div className="ninja-list">
16      { ninjaList }
17    </div>
18  )
19}
20
21 export default Ninjas
```

```
1 import React from 'react';
2
3 const Ninjas = (props) => {
4   const { ninjas } = props;
5   const ninjaList = ninjas.map(ninja => {
6     return (
7       <div className="ninja" key={ninja.id}>
8         <div>Name: { ninja.name }</div>
9         <div>Age: { ninja.age }</div>
10        <div>Belt: { ninja.belt }</div>
11      </div>
12    )
13  })
14  return(
15    <div className="ninja-list">
16      { ninjaList }
17    </div>
18  )
19}
20
21 export default Ninjas
```

A screenshot of the Visual Studio Code interface. The title bar shows 'Ninjas.js - react-reduce-complete - Visual Studio Code'. The left sidebar has icons for file, search, and other functions. There are two tabs open: 'App.js' and 'Ninjas.js'. The 'App.js' tab is active and displays the following code:

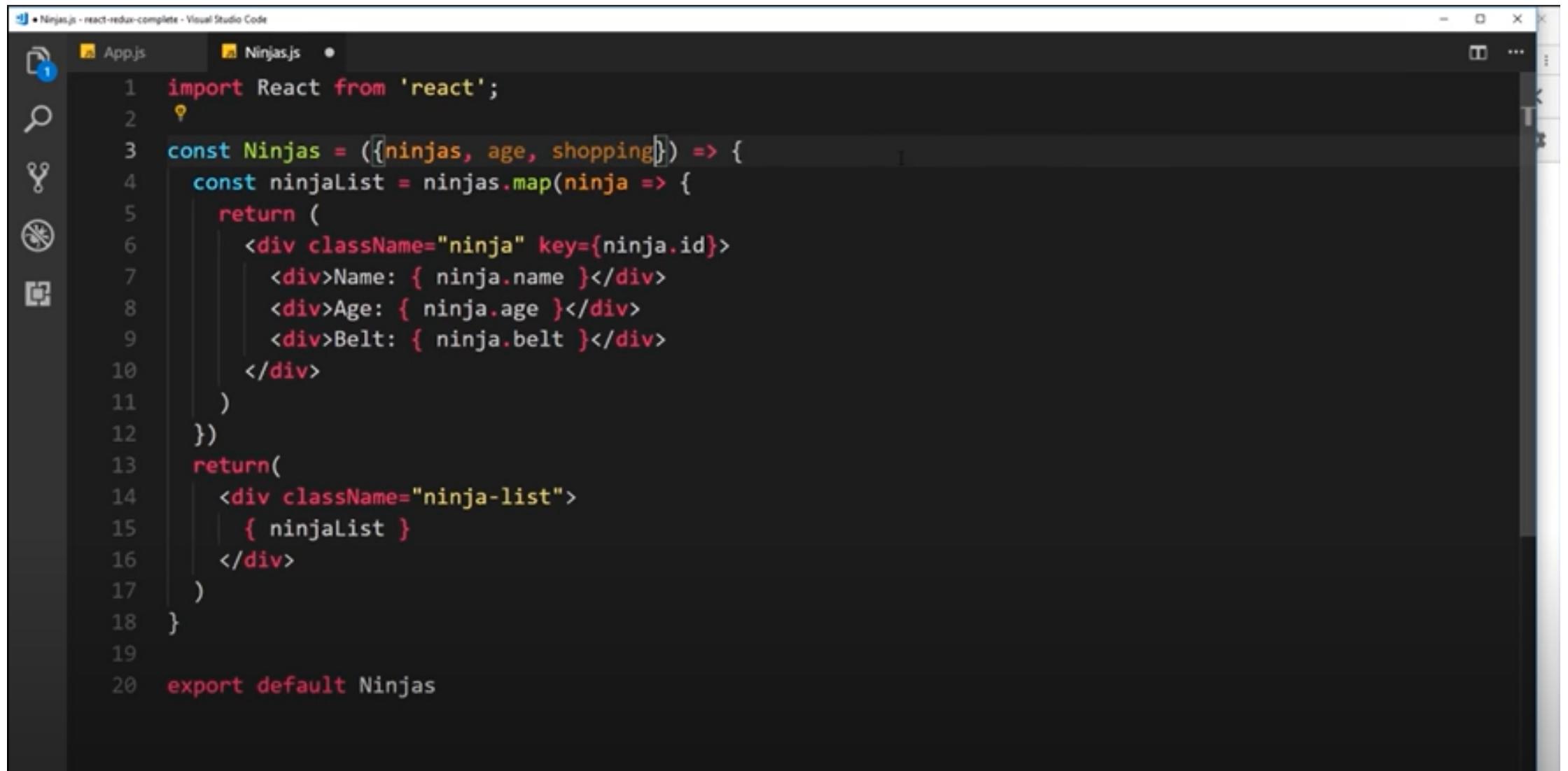
```
1 import React from 'react';
2
3 const Ninjas = (props) => {
4   const { ninjas } = props;
5   const ninjaList = ninjas.map(ninja => {
6     return (
7       <div className="ninja" key={ninja.id}>
8         <div>Name: { ninja.name }</div>
9         <div>Age: { ninja.age }</div>
10        <div>Belt: { ninja.belt }</div>
11      </div>
12    )
13  })
14  return(
15    <div className="ninja-list">
16      { ninjaList }
17    </div>
18  )
19}
20
21 export default Ninjas
```

Final One



The screenshot shows a code editor window in Visual Studio Code with the title bar "Ninjas.js - react-redux-complete - Visual Studio Code". The left sidebar has icons for file operations, search, and other tools. The main editor area displays the following code:

```
1 import React from 'react';
2
3 const Ninjas = ({nинjas}) => {
4     const ninjaList = ninjas.map(ninja => {
5         return (
6             <div className="ninja" key={ninja.id}>
7                 <div>Name: { ninja.name }</div>
8                 <div>Age: { ninja.age }</div>
9                 <div>Belt: { ninja.belt }</div>
10            </div>
11        )
12    })
13    return(
14        <div className="ninja-list">
15            { ninjaList }
16        </div>
17    )
18 }
19
20 export default Ninjas
```



```
1 import React from 'react';
2
3 const Ninjas = ({ninja}) => {
4   const ninjaList = ninja.map(ninja => {
5     return (
6       <div className="ninja" key={ninja.id}>
7         <div>Name: {ninja.name}</div>
8         <div>Age: {ninja.age}</div>
9         <div>Belt: {ninja.belt}</div>
10      </div>
11    )
12  })
13  return(
14    <div className="ninja-list">
15      {ninjaList}
16    </div>
17  )
18}
19
20 export default Ninjas
```

If have many

```
1 import React from 'react';
2 
3 const Ninjas = (props) => {
4   const ninjaList = ninjas.map(ninja => {
5     return (
6       <div className="ninja" key={ninja.id}>
7         <div>Name: { ninja.name }</div>
8         <div>Age: { ninja.age }</div>
9         <div>Belt: { ninja.belt }</div>
10      </div>
11    )
12  })
13  return(
14    <div className="ninja-list">
15      { ninjaList }
16    </div>
17  )
18}
19
20 export default Ninjas
```

Also write this