

LECTURE #2

By: Aqib Rehman

OUTLINE

React Dev Tools

DOM Events

Changing State (and 'this')

REACT DEV TOOLS

To Interact with components

chrome web store

Search the store

Extensions Themes

CATEGORIES All

FEATURES Runs Offline By Google Free Available for Works with

RATINGS ★★★★☆ (1020) Developer Tools 1,112,882 users

OVERVIEW REVIEWS SUPPORT RELATED

React Developer Tools
offered by Facebook

ADDED TO CHROME

Compatible with your device

Adds React debugging tools to the Chrome Developer Tools.

React Developer Tools is a Chrome DevTools extension for the open-source React JavaScript library. It allows you to inspect the React component hierarchies in the Chrome Developer Tools.

You will get a new tab called React in your Chrome DevTools. This shows you the root React components that were rendered on the page, as well as the subcomponents that they ended up rendering.

Website Report Abuse

Additional Information
Version: 3.2.4
Updated: July 16, 2018
Size: 314KB
Language: English

View all

Privacy Policy

RELATED

react-detector Refined GitHub Pix to pix (Pixel perfect by Ymatuhin) React Performance Devtool

<https://chromewebstore.google.com/detail/react-developer-tools/fmkadmapgofadopljbfkapdkoienih?hl=en&pli=1>

React Developer Tools - Document React - A JavaScript lib... 127.0.0.1:5500/index.html

Hey, ninjas

My name is: Crystal and I am 35

Elements Sources Console Network

```
<html lang="en">
  <head>...</head>
  ... <body> == $0
    <div id="app">...</div>
    <script type="text/babel">...</script>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```

Styles

:hover .cls +

```
element.style
{
}

user agent sty...
body {
  display: block;
  margin: 8px;
}
```

grid 8

order -

padding -

- 625.333 × 650.667 -

-

-

8

React Developer Tools Document React - A JavaScript libri 127.0.0.1:5500/index.html

Hey, ninjas

My name is: Crystal and I am 35

Elements Sources Console Network Performance Memory

```
<html lang="en">
  <head>...</head>
  ... ▼<body> == $0
    ▶ <div id="app">...</div>
    ▶ <script type="text/babel">...</script>
      <!-- Code injected by live-server -->
    ▶ <script type="text/javascript">...</script>
  </body>
</html>
```

Application Security Audits React .cls + element.style { } user agent sty... body { display: block; margin: 8px; } min 8 order - padding - 434.667 x 650.667 - - - - 8

React Developer Tools - Document React - A JavaScript lib... 127.0.0.1:5500/index.html

Hey, ninjas

My name is: Crystal and I am 35

App | 434.66668701171875px x 76.10417175292969px

Elements Sources Console Network Performance React » ⚠ 1 : X

Highlight Updates Highlight Search

Search (text or /regex/)

<App> == \$r

<div className="app-content">

<h1>Hey, ninjas</h1>

<p>

"My name is: "

"Crystal"

" and I am "

"35"

</p>

</div>

</App>

Props
Empty object

State
age: 35
name: "Crystal"

The screenshot shows the React Developer Tools interface integrated into a browser window. The title bar indicates the tab is "React - A JavaScript file". The main content area displays the application's UI with the text "Hey, ninjas" and "My name is: Crystal and I am 35". On the right side, the React tab is selected, showing the component tree: <App>...</App>. The state object is shown as an empty object. The props object contains the state object with properties: age: 35 and name: "Crystal".

Hey, ninjas

My name is: Crystal and I am 35

React Developer Tools Document React - A JavaScript file 127.0.0.1:5500/index.html

Elements Sources Console Network Performance React ⚠ 1

Highlight Updates Highlight Search

Search (text or /regex/)

<App>...</App> *** \$r

Props
Empty object

State
age: 35
name: "Crystal" I

The screenshot shows the React Developer Tools interface integrated into a browser window. The title bar includes tabs for "React Developer Tools" and "Document". The address bar shows the URL "127.0.0.1:5500/index.html". The main content area displays the text "Hey, ninjas" and "My name is: Ryu and I am 35". The React tab is selected, showing a component tree with the root node being "<App>...</App> == \$r". To the right, the "Props" section shows an "Empty object". The "State" section shows two properties: "age: 35" and "name: "Ryu"".

Hey, ninjas

My name is: Ryu and I am 35

React Developer Tools - Document

React - A JavaScript library

127.0.0.1:5500/index.html

Elements Sources Console Network Performance React > ⚠ 1 ⋮ X

Highlight Updates Highlight Search

Search (text or /regex/)

<App>...</App> == \$r

Props
Empty object

State
age: 35
name: "Ryu"

Its not for end user

He/she will not interact with it like this

It is just for you to play around this

The screenshot shows the React Developer Tools interface. On the left, the code editor displays the `TodoApp` component's code. The right side shows the component tree and its state.

Code Editor (TodoApp):

```
class TodoApp extends React.Component { render() { return ( <div> <h1>TODO</h1> <TodoList items={this.state.items} /> <form onSubmit={this.handleSubmit}> <Label htmlFor="new-todo"> What needs to be done? </Label> <input type="text" id="new-todo" value={this.state.newTodo} onChange={this.handleChange} /> <button type="submit">Add #2</button> </form> </div> ); } }
```

RESULT:

TODOS

- hey there

What needs to be done?

Add #2

React DevTools:

- Elements
- Sources
- Console
- Network
- Performance
- React
- More

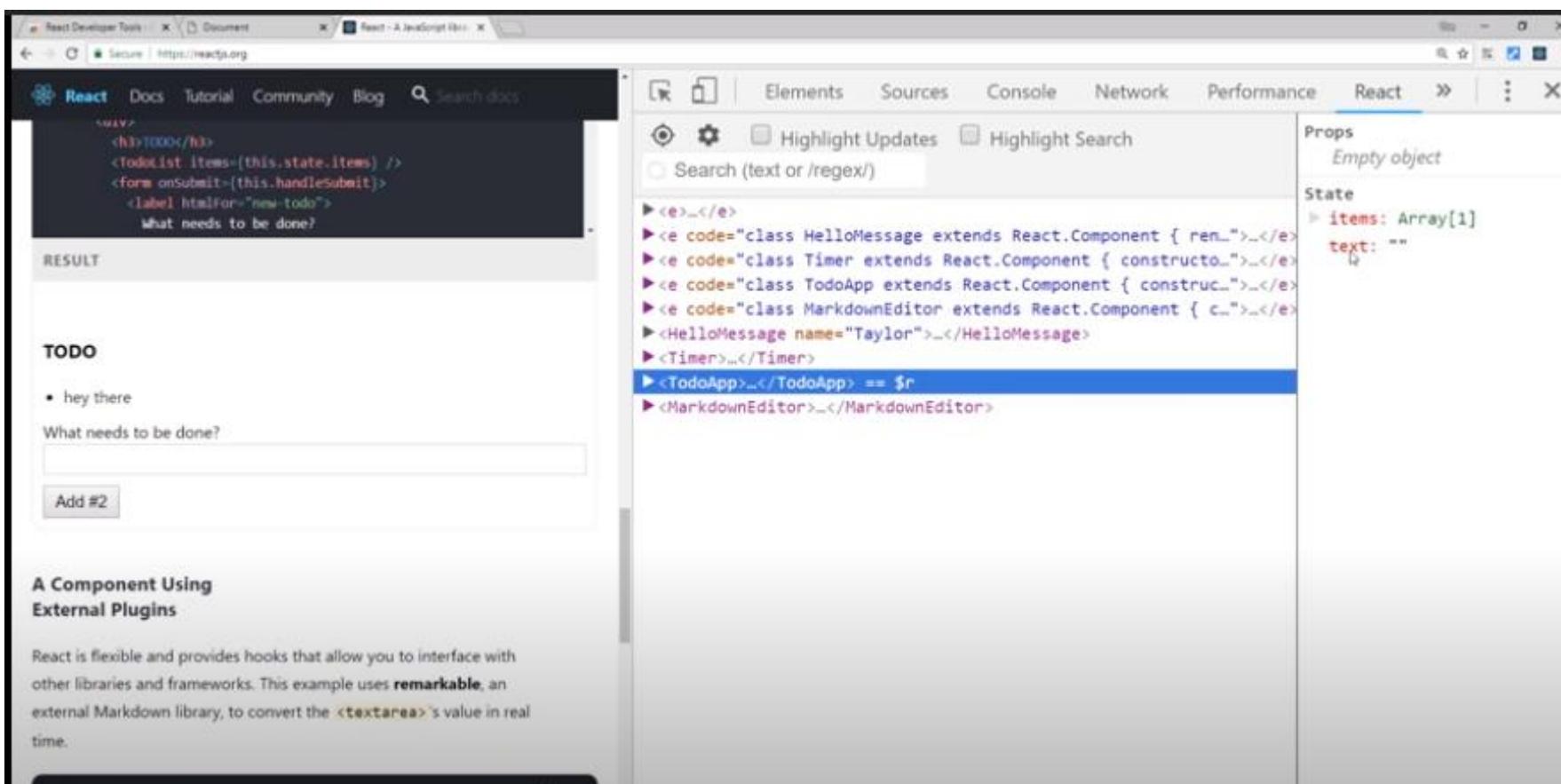
Props: Empty object

State:

```
items: Array[1]
  0: {}
    id: 1533114363390
    text: "hey there"
  text: ""
```

Production version of react
Sign color is black and blue

<https://legacy.reactjs.org/>



The screenshot shows the React Developer Tools interface over a browser window displaying a React application. The browser tab is titled "React - A JavaScript View".

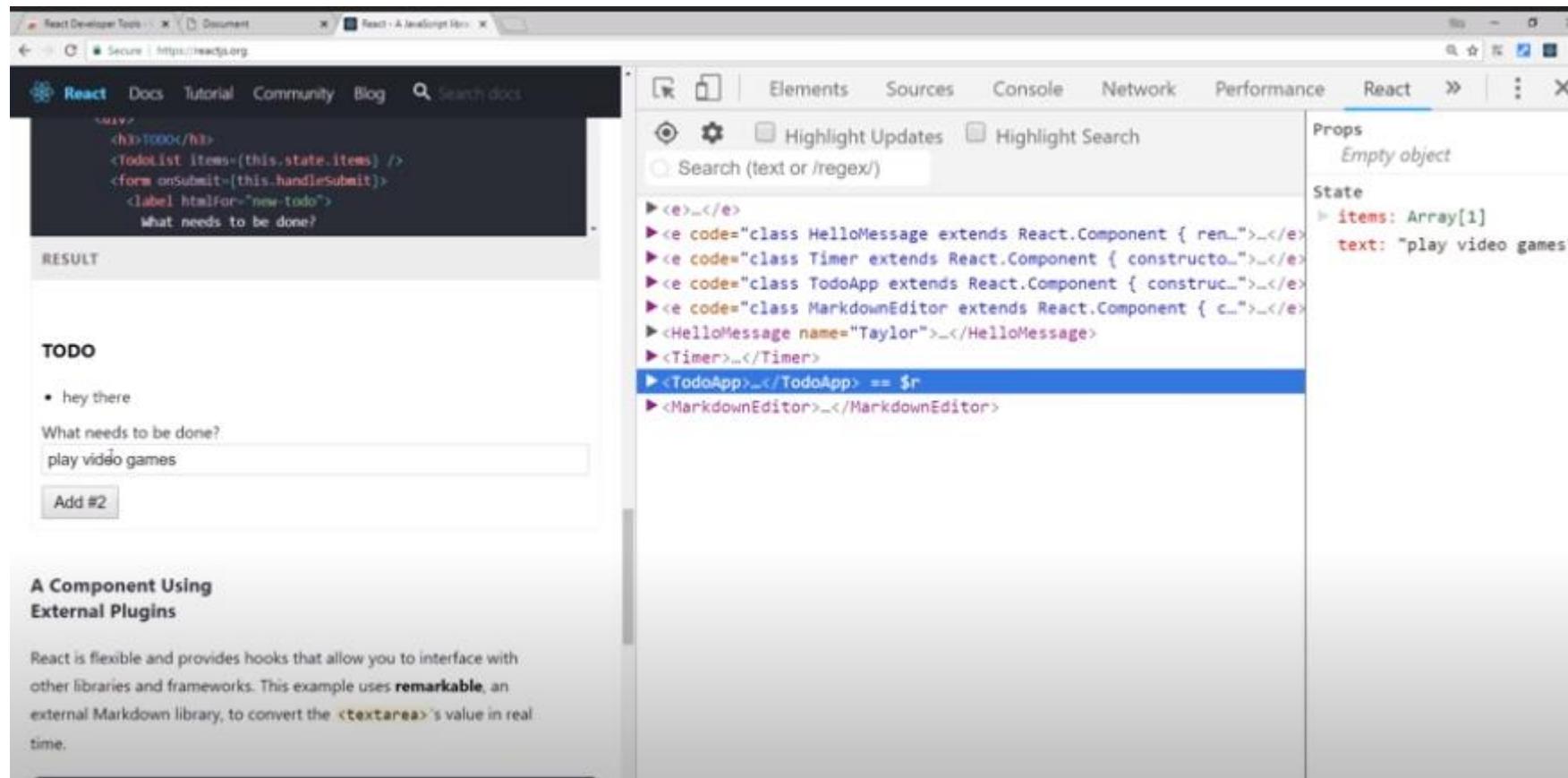
The React application displays a "TODO" list with one item: "hey there". Below the list is a form with a text input containing "buy m" and a button labeled "Add #2". To the left of the browser window, a sidebar section titled "A Component Using External Plugins" explains that the application uses the `remarkable` library to convert Markdown input.

The React DevTools sidebar has the "React" tab selected. It shows the component tree on the left:

```
> <e>..</e>
▶ <e code="class HelloMessage extends React.Component { ren...">..</e>
▶ <e code="class Timer extends React.Component { constructo...">..</e>
▶ <e code="class TodoApp extends React.Component { construc...">..</e>
▶ <e code="class MarkdownEditor extends React.Component { c...">..</e>
▶ <HelloMessage name="Taylor">..</HelloMessage>
▶ <Timer>..</Timer>
▶ <>TodoApp>..</TodoApp> == $r
▶ <>MarkdownEditor>..</MarkdownEditor>
```

The "Props" panel shows an "Empty object". The "State" panel shows:

- `items: Array[1]`
- `text: "buy "` (highlighted in yellow)



If type on right side states...

The screenshot shows the React Developer Tools interface over a browser window displaying the React documentation. The browser's address bar shows the URL <https://reactjs.org>.

The React DevTools sidebar on the right shows the component tree and state:

- Props:** Empty object
- State:**
 - items:** Array[1]
 - 0:** {}
 - id:** 1533114363398
 - text:** "buy m | I"
 - text:** "play video games"

The component tree on the left shows the following structure:

```
<HelloMessage name="Taylor"></HelloMessage>
<Timer></Timer>
<TodoApp>...</TodoApp> == $r
<MarkdownEditor>...</MarkdownEditor>
```

The browser window displays the React documentation for a Todo application. It includes:

- A code snippet showing the `TodoApp` component's render method.
- A **RESULT** section showing the application's state:
 - A **TODO** list with one item: "hey there".
 - A text input field with placeholder "What needs to be done?" containing "play video games".
 - A button labeled "Add #2".
- A section titled "A Component Using External Plugins" explaining how the `MarkdownEditor` component uses the `remarkable` library to convert Markdown input to rich text.

The screenshot shows the React developer tools integrated into a browser window. The left panel displays the code for a TodoList component and its rendered result. The right panel shows the component tree and the state of the TodoApp component.

Code (Left Panel):

```
SNIPPET
<h3>1000</h3>
<TodoList items={this.state.items} />
<form onSubmit={this.handleSubmit}>
  <label htmlFor="new-todo">
    What needs to be done?
  </label>
  <input type="text" id="new-todo" value="buy milk" />
  <button type="submit">Add #2</button>
</form>
```

RESULT (Left Panel):

TODO

- buy milk

What needs to be done?

Add #2

React DevTools (Right Panel):

- Elements
- Sources
- Console
- Network
- Performance
- React**
- ...
- X

Props
Empty object

State

```
items: Array[1]
  0: {}
    id: 1533114363390
    text: "buy milk"
    text: "play video games"
```

If click on Add #2

The screenshot shows the React Developer Tools interface over a browser window displaying the React.js documentation. The browser's address bar shows the URL `https://reactjs.org`.

The React DevTools sidebar on the left displays the following code snippet:

```
    <h1>TODO</h1>
    <TodoList items={this.state.items} />
    <form onSubmit={this.handleSubmit}>
      <label htmlFor="new-todo">
        what needs to be done?
      </label>
      <input type="text" id="new-todo" value=""/>
      <button type="submit">Add #3</button>
    </form>
```

The main pane shows the rendered application state:

- RESULT**: Displays the rendered `TodoList` component containing two items:
 - buy milk
 - play video games
- TODO**: Displays the form state:
 - items**: An array of two objects, each with `id` and `text` properties.
 - text**: An empty string.
- What needs to be done?**: A text input field with the placeholder "What needs to be done?" and the value "buy milk".
- Add #3**: A button to add a new item.

The **React** tab of the DevTools is selected, showing the component tree and state structure. The `TodoApp` component is highlighted in blue. The state structure is as follows:

- Props**: Empty object.
- State**:
 - items**: An array of 2 items.
 - 0**: {
 id: 1533114363390,
 text: "buy milk"}
}
 - 1**: {
 id: 1533115039726,
 text: "play video games"}
}
 - text**: ""

Nice tool for interaction with components

DOM EVENTS

```
indexhtml •
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>Hey, ninjas</h1>
25                     <p>My name is: { this.state.name } and I am { this.state.age }</p>
26                     <button></button>
27                 </div>
28             )
29         }
30     }
31
32     ReactDOM.render(<App />, document.getElementById('app'));
33 </script>
34
35 </body>
36 </html>
```

A screenshot of the Visual Studio Code interface showing an open file named "index.html". The code is a simple React application. The interface includes a left sidebar with icons for file operations, a search bar, and a status bar at the bottom.

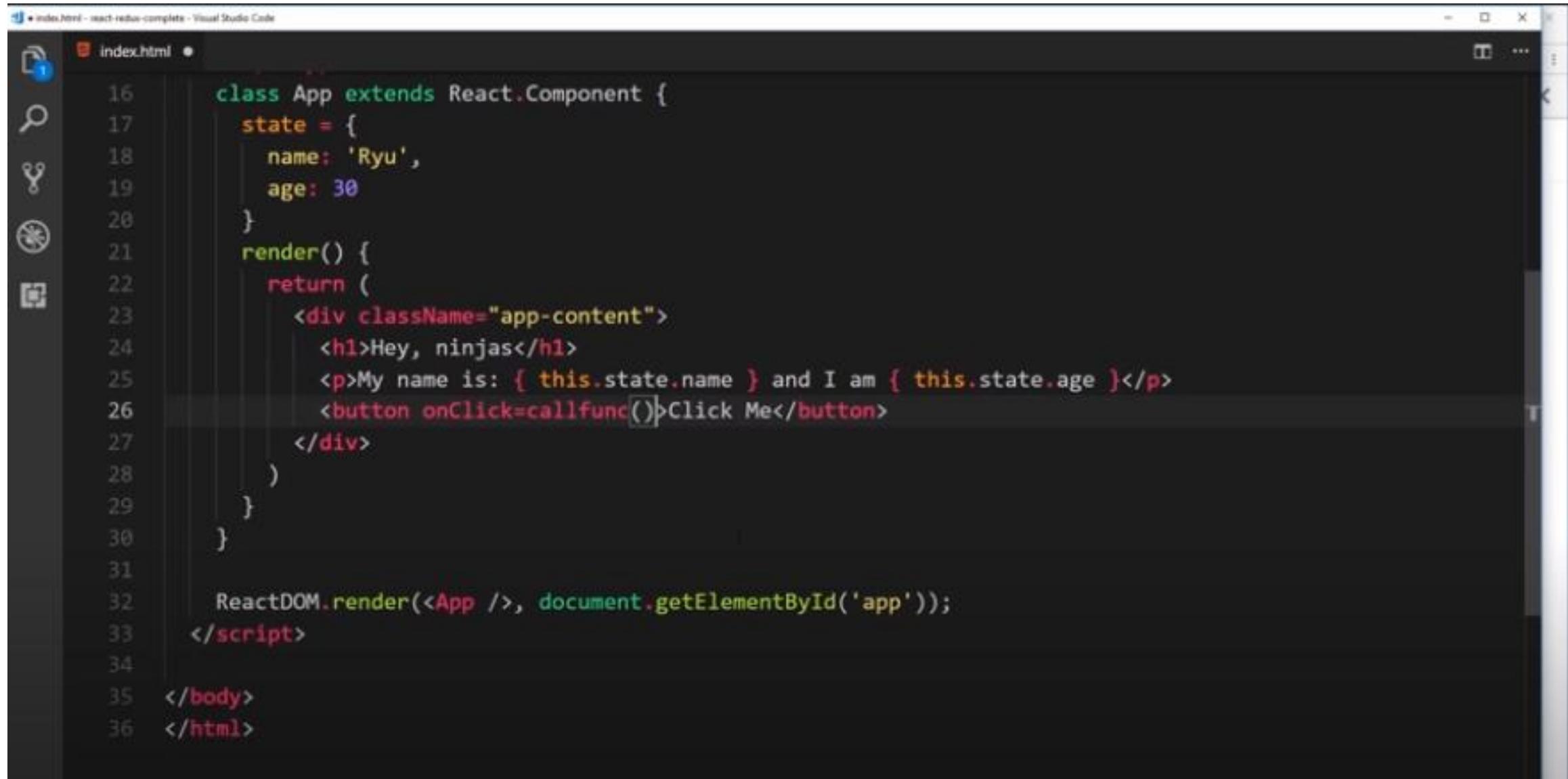
```
16  class App extends React.Component {  
17      state = {  
18          name: 'Ryu',  
19          age: 30  
20      }  
21      render() {  
22          return (  
23              <div className="app-content">  
24                  <h1>Hey, ninjas</h1>  
25                  <p>My name is: { this.state.name } and I am { this.state.age }</p>  
26                  <button>Click Me</button>  
27              </div>  
28          )  
29      }  
30  }  
31  
32  ReactDOM.render(<App />, document.getElementById('app'));  
33  </script>  
34  
35  </body>  
36  </html>
```

NOT WRITE LIKE THIS

A screenshot of the Visual Studio Code interface showing an `index.html` file. The code contains a class-based React component named `App`. The component has a state object with properties `name` and `age`, both set to their respective values. The `render` method returns a `<div>` element containing an `<h1>` and a `<p>` element. The `<p>` element uses `this.state.name` and `this.state.age` as its text content. Below the component definition, there is a `<script>` block that renders the component into the DOM using `ReactDOM.render`.

```
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>Hey, ninjas</h1>
25          <p>My name is: { this.state.name } and I am { this.state.age }</p>
26          <button onClick="">Click Me</button>
27        </div>
28      )
29    }
30  }
31
32  ReactDOM.render(<App />, document.getElementById('app'));
33</script>
34
35</body>
36</html>
```

NOT WRITE LIKE THIS



A screenshot of the Visual Studio Code interface showing an `index.html` file. The code contains several anti-patterns in React development:

- The component is defined as a class that extends `React.Component`.
- The state is stored directly on the component instance instead of using `useState` or `useReducer`.
- The button's `onClick` handler is named `callfunc()`, which is a poor choice of name.
- The script ends with a closing `>` symbol on the last line.

```
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>Hey, ninjas</h1>  
25          <p>My name is: { this.state.name } and I am { this.state.age }</p>  
26          <button onClick=callfunc()>Click Me</button>  
27        </div>  
28      )  
29    }  
30  }  
31  
32  ReactDOM.render(<App />, document.getElementById('app'));  
33</script>  
34  
35</body>  
36</html>
```

```
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>Hey, ninjas</h1>  
25                  <p>My name is: { this.state.name } and I am { this.state.age }</p>  
26                  <button onClick={()>}>Click Me</button>  
27              </div>  
28          )  
29      }  
30  }  
31  
32  ReactDOM.render(<App />, document.getElementById('app'));  
33  </script>  
34  
35  </body>  
36  </html>
```

```
14
15  <script type="text/babel">
16      class App extends React.Component {
17          state = {
18              name: 'Ryu',
19              age: 30
20          }
21          handleClick(e){
22              |
23          }
24          render() {
25              return (
26                  <div className="app-content">
27                      <h1>Hey, ninjas</h1>
28                      <p>My name is: { this.state.name } and I am { this.state.age }</p>
29                      <button onClick={()>}Click Me</button>
30                  </div>
31          )
32      }
33  }
34
35  ReactDOM.render(<App />, document.getElementById('app'));
36
```

index.html ●

```
14  
15 <script type="text/babel">  
16   class App extends React.Component {  
17     state = {  
18       name: 'Ryu',  
19       age: 30  
20     }  
21     handleClick(e){  
22       console.log(e.target);  
23     }  
24     render() {  
25       return (  
26         <div className="app-content">  
27           <h1>Hey, ninjas</h1>  
28           <p>My name is: { this.state.name } and I am { this.state.age }</p>  
29           <button onClick={}>Click Me</button>  
30         </div>  
31       )  
32     }  
33   }  
34  
35   ReactDOM.render(<App />, document.getElementById('app'));  
36 </script>
```

Not like this

A screenshot of Visual Studio Code showing an `index.html` file. The code defines a `App` component that logs the clicked element to the console. A tooltip for the `handleClick` prop is visible over the `onClick={handleClick}` attribute.

```
14<script type="text/babel">
15  class App extends React.Component {
16    state = {
17      name: 'Ryu',
18      age: 30
19    }
20    handleClick(e){
21      console.log(e.target);
22    }
23    render() {
24      return (
25        <div className="app-content">
26          <h1>Hey, ninjas</h1>
27          <p>My name is: { this.state.name } and I am { this.state.age }</p>
28          <button onClick={handleClick}>Click Me</button>
29        </div>
30      )
31    }
32  }
33}
34
35ReactDOM.render(<App />, document.getElementById('app'));
```

index.html

```
14<script type="text/babel">
15    class App extends React.Component {
16        state = {
17            name: 'Ryu',
18            age: 30
19        }
20        handleClick(e){
21            console.log(e.target);
22        }
23        render() {
24            return (
25                <div className="app-content">
26                    <h1>Hey, ninjas</h1>
27                    <p>My name is: { this.state.name } and I am { this.state.age }</p>
28                    <button onClick={this.handleClick}>Click Me</button>
29                </div>
30            )
31        }
32    }
33}
34
35ReactDOM.render(<App />, document.getElementById('app'));
```

The screenshot shows a browser window with the address bar pointing to "127.0.0.1:5550/index.html". The main content area displays the text "Hey, ninjas" and "My name is: Ryu and I am 30" followed by a button labeled "Click Me". On the right side, the developer tools' "Console" tab is active, showing a yellow warning message: "⚠ You are using the in-browser Babel transformer. Be sure to precompile your scripts for production - <https://babeljs.io/docs/setup/>". Below the message, the DOM structure of the button is visible: "<button>Click Me</button>". To the right of the DOM structure, the text "Inline Babel script:8" is displayed. The browser's sidebar icons are partially visible on the left.

Hey, ninjas

My name is: Ryu and I am 30

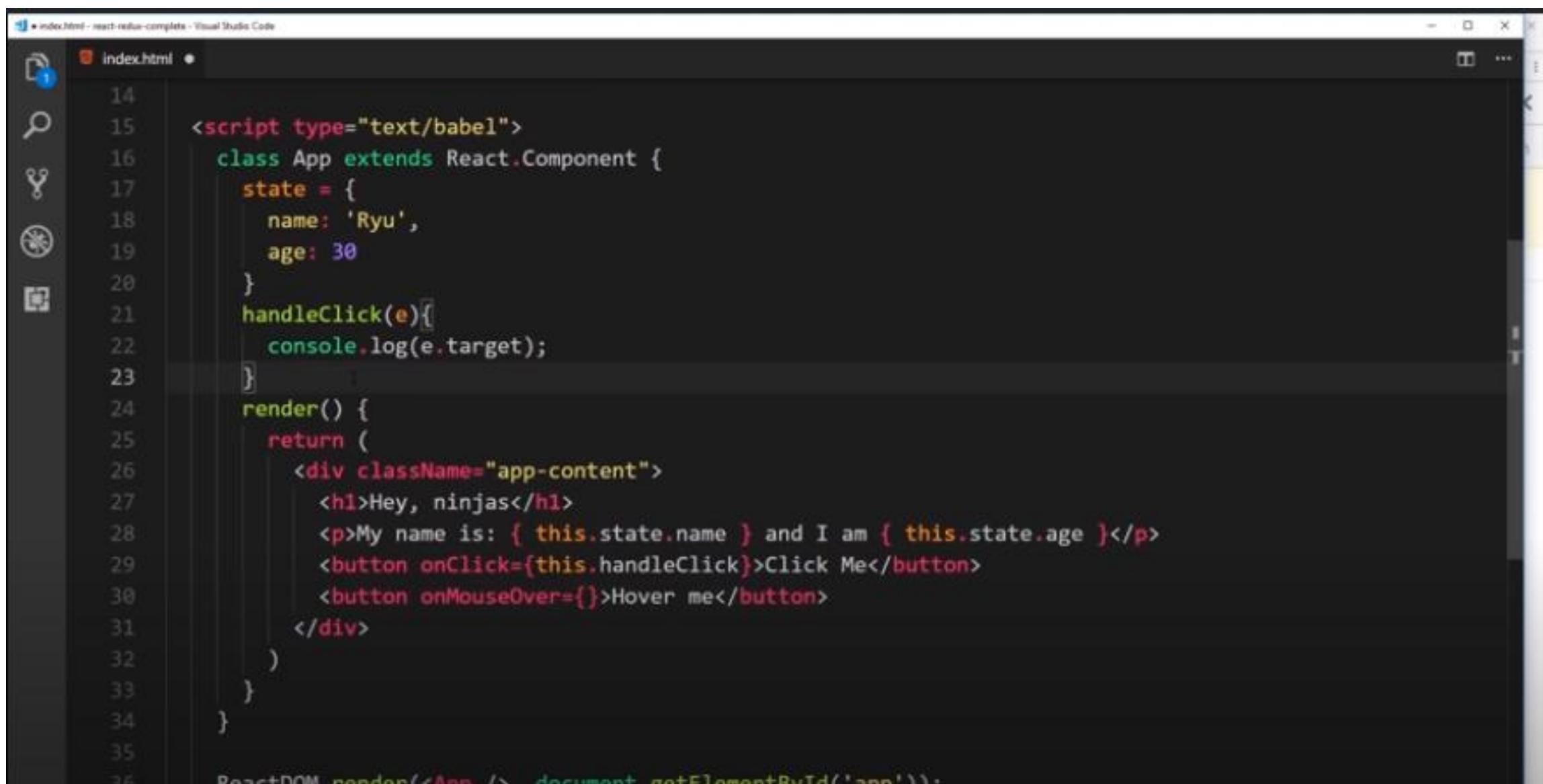
Click Me

⚠ You are using the in-browser Babel transformer. Be sure to precompile your scripts for production - <https://babeljs.io/docs/setup/>

<button>Click Me</button>

Inline Babel script:8

Another Event



A screenshot of the Visual Studio Code interface showing an open file named "index.html". The code is written in JSX (JavaScript with XML-like structures) and defines a React component named "App". The component has a state object with properties "name" (set to "Ryu") and "age" (set to 30). It contains two buttons: one with an "onClick" event handler that logs the target element to the console, and another with an "onMouseOver" event handler that also logs its target. The component's render method returns a div containing an h1 and a p element with dynamic text, and two buttons.

```
index.html - react-value-complete - Visual Studio Code
index.html • 14

14
15 <script type="text/babel">
16   class App extends React.Component {
17     state = {
18       name: 'Ryu',
19       age: 30
20     }
21     handleClick(e){
22       console.log(e.target);
23     }
24     render() {
25       return (
26         <div className="app-content">
27           <h1>Hey, ninjas</h1>
28           <p>My name is: { this.state.name } and I am { this.state.age }</p>
29           <button onClick={this.handleClick}>Click Me</button>
30           <button onMouseOver={()>Hover me</button>
31         </div>
32       )
33     }
34   }
35
36 ReactDOM.render(<App />, document.getElementById('app'));
```

A screenshot of the Visual Studio Code interface, showing the file `index.html` open. The code is a React component named `App` with state variables `name` and `age`, and methods `handleClick` and `handleMouseOver`. It also includes a `render` method returning a UI with an `h1` element and two `button` elements.

```
14<script type="text/babel">
15  class App extends React.Component {
16    state = {
17      name: 'Ryu',
18      age: 30
19    }
20    handleClick(e){
21      console.log(e.target);
22    }
23    handleMouseOver(e){
24      console.log(e)
25    }
26  }
27  render() {
28    return (
29      <div className="app-content">
30        <h1>Hey, ninjas</h1>
31        <p>My name is: { this.state.name } and I am { this.state.age }</p>
32        <button onClick={this.handleClick}>Click Me</button>
33        <button onMouseOver={this.handleMouseOver}>Hover me</button>
34      </div>
35    )
36  }
37</script>
```

A screenshot of the Visual Studio Code interface, showing the file `index.html`. The code is a React component named `App` with state variables `name` and `age`, and methods `handleClick` and `handleMouseOver`. It also includes a `render` method that returns a `div` containing an `h1`, a `p` element with a template string, two `button` elements, and a `p` element with an `onCopy` event handler.

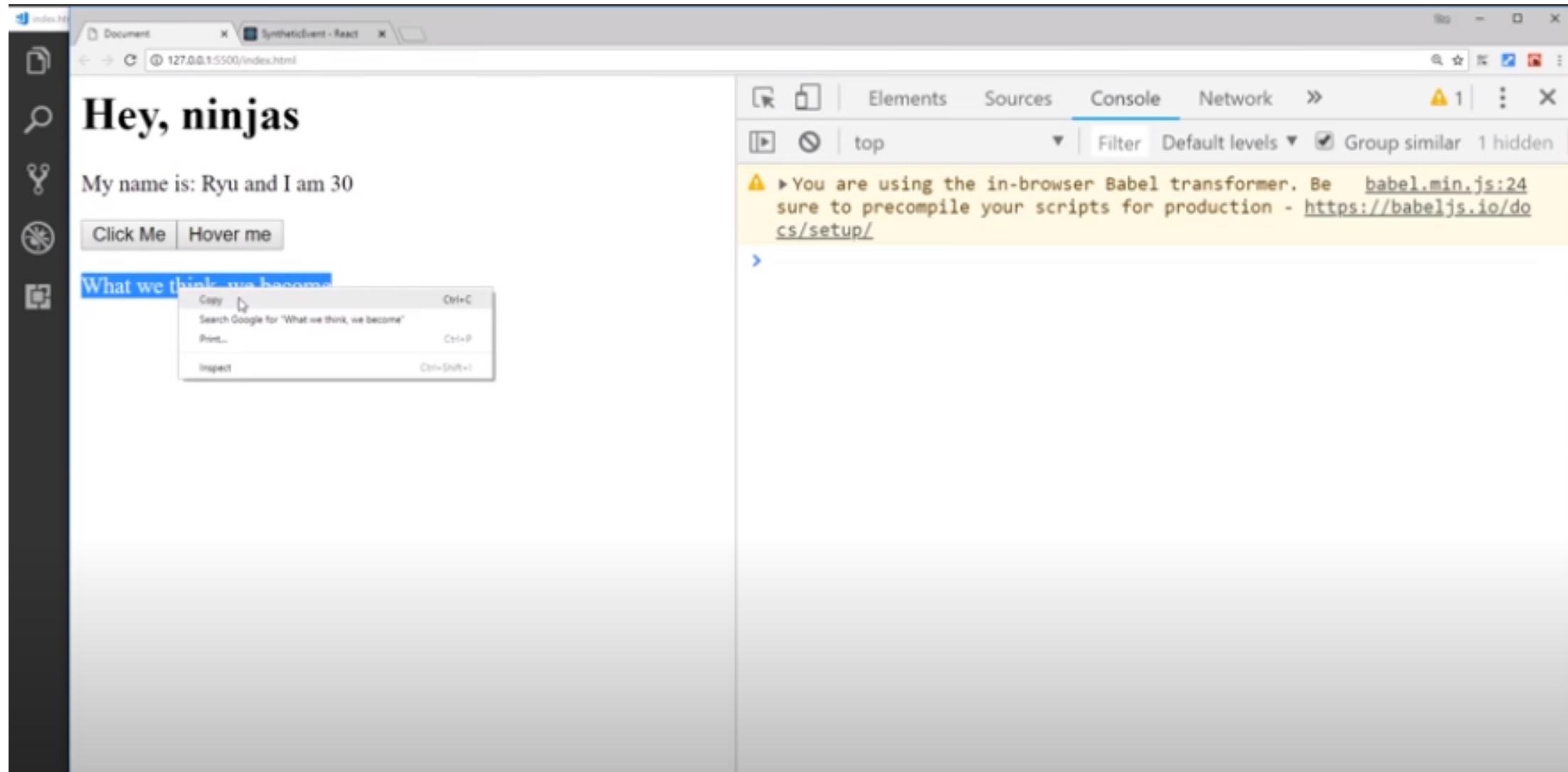
```
14<script type="text/babel">
15  class App extends React.Component {
16    state = {
17      name: 'Ryu',
18      age: 30
19    }
20    handleClick(e){
21      console.log(e.target);
22    }
23    handleMouseOver(e){
24      console.log(e.target, e.pageX)
25    }
26    render() {
27      return (
28        <div className="app-content">
29          <h1>Hey, ninjas</h1>
30          <p>My name is: { this.state.name } and I am { this.state.age }</p>
31          <button onClick={this.handleClick}>Click Me</button>
32          <button onMouseOver={this.handleMouseOver}>Hover me</button>
33          <p onCopy={()>What we think, we become</p>
34        </div>
35      )
36    }
37  }
38  export default App;
```

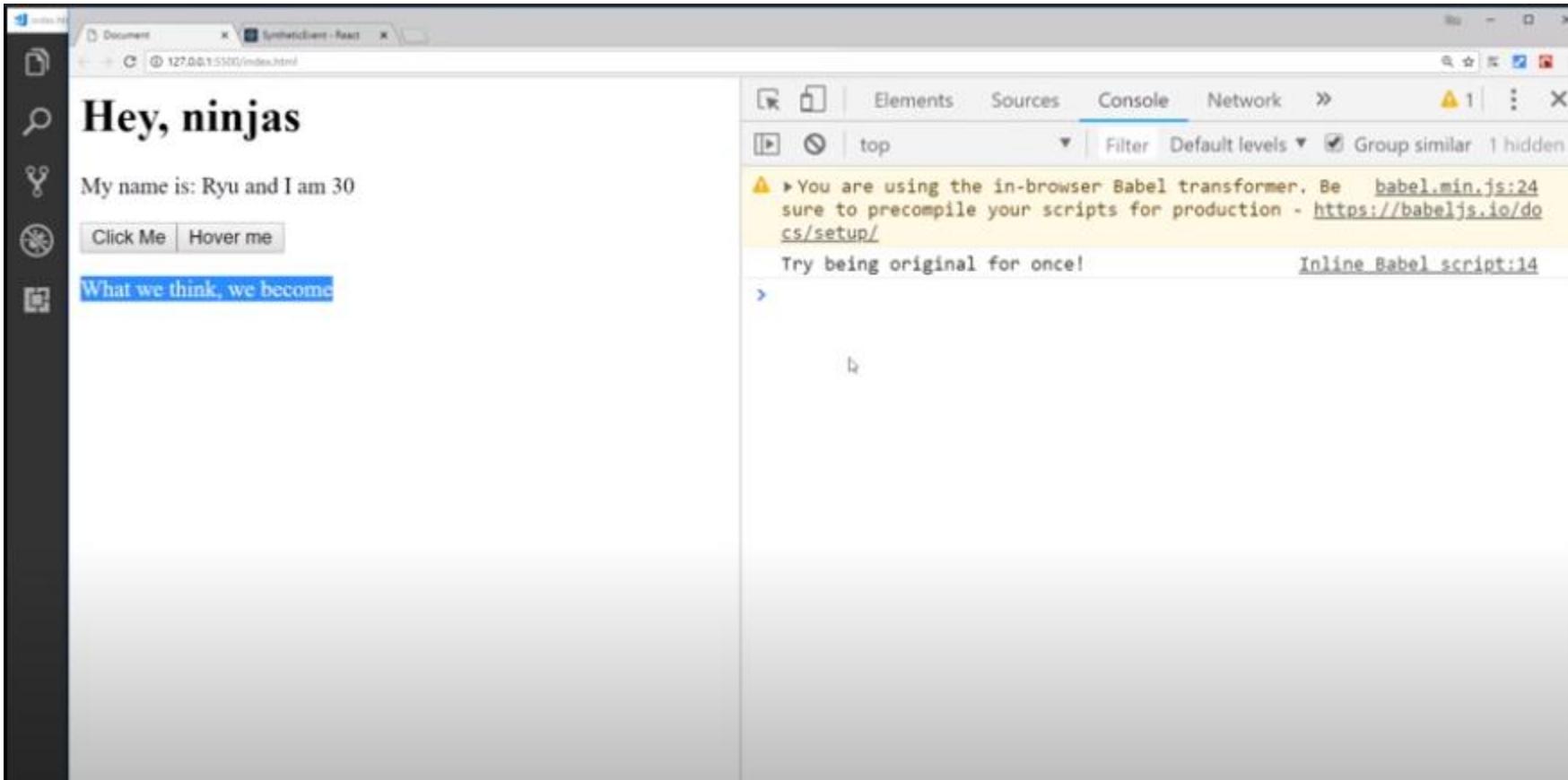
A screenshot of the Visual Studio Code interface showing an open file named "index.html". The code is written in JavaScript and uses React components. The code defines a class "App" that extends "React.Component". It has a state object with properties "name" and "age". It includes event handlers for click, mouse over, and copy events, all of which log information to the console. It also includes a render method that creates a div with a class name of "app-content", contains an h1 element with the text "Hey, ninjas", and a p element with the text "My name is: { this.state.name } and I am { this.state.age }". It also contains a button with an onClick handler that logs the event target to the console.

```
14<script type="text/babel">
15  class App extends React.Component {
16    state = {
17      name: 'Ryu',
18      age: 30
19    }
20    handleClick(e){
21      console.log(e.target);
22    }
23    handleMouseOver(e){
24      console.log(e.target, e.pageX)
25    }
26    handleCopy(e){
27      console.log('Try being original for once!');
28    }
29    render() {
30      return (
31        <div className="app-content">
32          <h1>Hey, ninjas</h1>
33          <p>My name is: { this.state.name } and I am { this.state.age }</p>
34          <button onClick={this.handleClick}>Click Me</button>
35        </div>
36      )
37    }
38  }
39  export default App;
```

index.html •

```
20      }
21      handleClick(e){
22          console.log(e.target);
23      }
24      handleMouseOver(e){
25          console.log(e.target, e.pageX)
26      }
27      handleCopy(e){
28          console.log('Try being original for once!');
29      }
30      render() {
31          return (
32              <div className="app-content">
33                  <h1>Hey, ninjas</h1>
34                  <p>My name is: { this.state.name } and I am { this.state.age }</p>
35                  <button onClick={this.handleClick}>Click Me</button>
36                  <button onMouseOver={this.handleMouseOver}>Hover me</button>
37                  <p onCopy={this.handleCopy}>What we think, we become</p>
38              </div>
39          )
40      }
41  }
```





Events in React

<https://reactjs.org/docs/events.html#supported-events>

```
index.html - react-redux-complete - Visual Studio Code
index.html ×

16 class App extends React.Component {
17     state = {
18         name: 'Ryu',
19         age: 30
20     }
21     handleClick(e){
22         //console.log(e.target);
23         console.log(this.state.age) this.state.age
24     }
25     handleMouseOver(e){
26         console.log(e.target, e.pageX)
27     }
28     handleCopy(e){
29         console.log('Try being original for once!');
30     }
31     render() {
32         return (
33             <div className="app-content">
34                 <h1>Hey, ninjas</h1>
35                 <p>My name is: { this.state.name } and I am { this.state.age }</p>
36                 <button onClick={this.handleClick}>Click Me</button>
37             </div>
38         )
39     }
40 }
```

Can we access state like this?

The screenshot shows a browser window with the URL `127.0.0.1:5500/index.html`. The page content includes:

- A title **Hey, ninjas**.
- A statement **My name is: Ryu and I am 30**.
- A button labeled **Click Me**.
- A button labeled **Hover me**.
- A statement **What we think, we become**.

The browser's developer tools are open, specifically the **Console** tab. It displays two error messages:

- Uncaught TypeError: Cannot read property 'state' of undefined** (Inline Babel script:9)
Stack trace:
 - at handleClick (<anonymous>:35:24)
 - at HTMLUnknownElement.callCallback (react-dom.development.js:140)
 - at Object.invokeGuardedCallbackDev (react-dom.development.js:178)
 - at Object.invokeGuardedCallback (react-dom.development.js:227)
 - at Object.invokeGuardedCallbackAndCatchFirstError (react-dom.development.js:241)
 - at executeDispatch (react-dom.development.js:599)
 - at executeDispatchesInOrder (react-dom.development.js:621)
 - at executeDispatchesAndRelease (react-dom.development.js:719)
 - at executeDispatchesAndReleaseTopLevel (react-dom.development.js:730)
 - at forEachAccumulated (react-dom.development.js:700)
- Uncaught TypeError: Cannot read property 'state' of undefined** (react-dom.development.js:280)
Stack trace:
 - at handleClick (<anonymous>:35:24)
 - at HTMLUnknownElement.callCallback (react-dom.development.js:140)
 - at Object.invokeGuardedCallbackDev (react-dom.development.js:178)
 - at Object.invokeGuardedCallback (react-dom.development.js:227)
 - at Object.invokeGuardedCallbackAndCatchFirstError (react-dom.development.js:241)
 - at executeDispatch (react-dom.development.js:599)
 - at executeDispatchesInOrder (react-dom.development.js:621)
 - at executeDispatchesAndRelease (react-dom.development.js:719)

Context of this is loss

In render func React handle it by self

But in our custom functions we have to deal with it manually

ARROW FUNCTIONS

index.html - react-redux-complete - Visual Studio Code

index.html

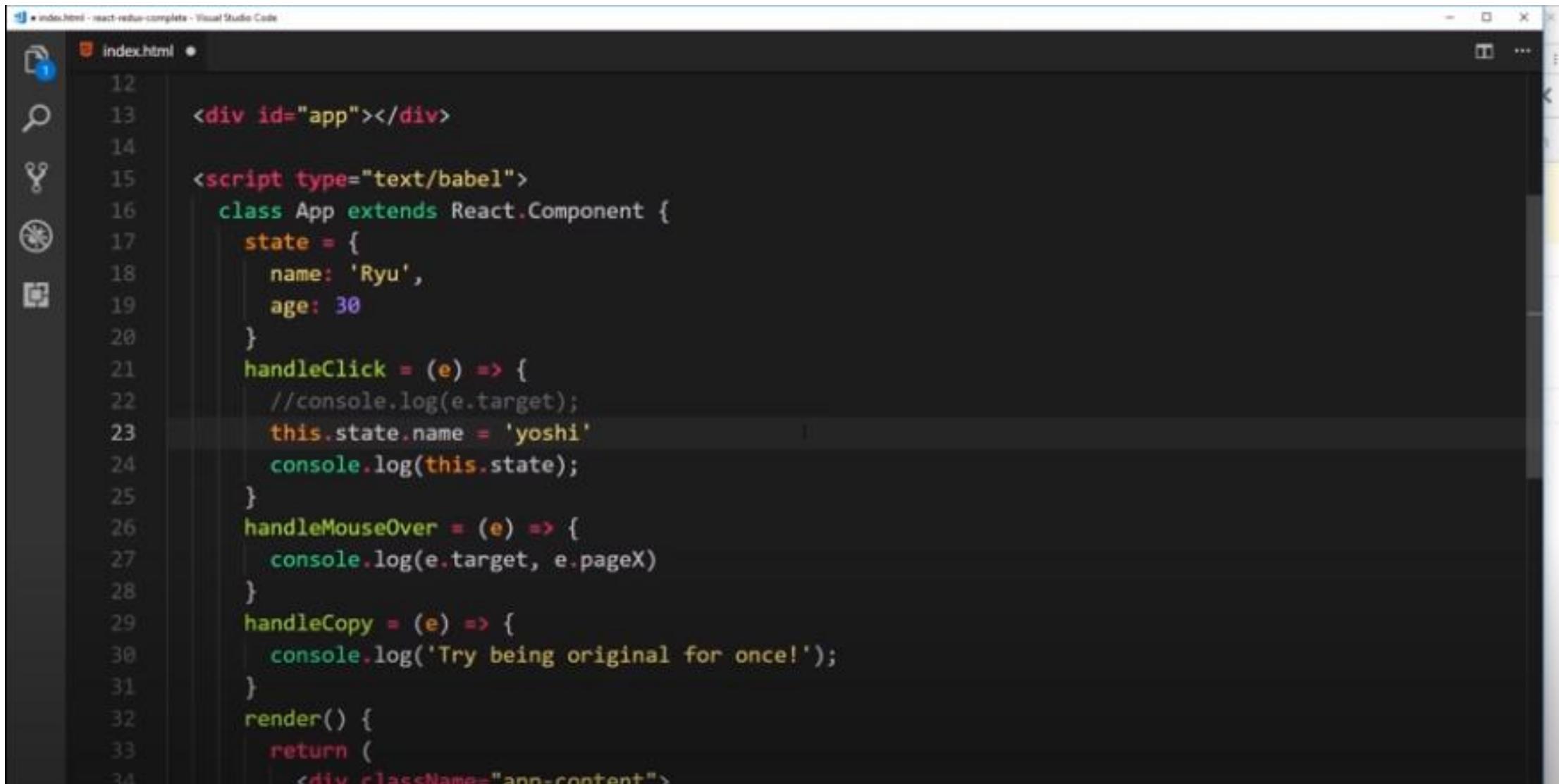
```
18     name: 'Ryu',
19     age: 30
20   }
21   handleClick = (e) => {
22     //console.log(e.target);
23     console.log(this.state);
24   }
25   handleMouseOver(e){
26     console.log(e.target, e.pageX)
27   }
28   handleCopy(e){
29     console.log('Try being original for once!');
30   }
31   render() {
32     return (
33       <div className="app-content">
34         <h1>Hey, ninjas</h1>
35         <p>My name is: { this.state.name } and I am { this.state.age }</p>
36         <button onClick={this.handleClick}>Click Me</button>
37         <button onMouseOver={this.handleMouseOver}>Hover me</button>
38         <p onCopy={this.handleCopy}>What we think, we become</p>
39       </div>

```

```
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    handleClick = (e) => {
22      //console.log(e.target);
23      console.log(this.state);
24    }
25    handleMouseOver = (e) => {
26      console.log(e.target, e.pageX)
27    }
28    handleCopy = (e) => [
29      console.log('Try being original for once!');
30    ]
31    render() {
32      return (
33        <div className="app-content">
34          <h1>Hey, ninjas!</h1>
```

CHANGING STATE (AND 'THIS')

Not a good Practice



A screenshot of the Visual Studio Code interface showing the file `index.html`. The code is a React component named `App` with several event handlers: `handleClick`, `handleMouseOver`, and `handleCopy`. The `handleClick` handler logs the target element to the console and changes the state's name to 'yoshi'. The `handleMouseOver` and `handleCopy` handlers log the target element and its page X position to the console.

```
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    handleClick = (e) => {
22      //console.log(e.target);
23      this.state.name = 'yoshi'
24      console.log(this.state);
25    }
26    handleMouseOver = (e) => {
27      console.log(e.target, e.pageX)
28    }
29    handleCopy = (e) => {
30      console.log('Try being original for once!');
31    }
32    render() {
33      return (
34        <div className="app-content">
```

A screenshot of the Visual Studio Code interface showing an open file named "index.html". The code is written in JSX (JavaScript with XML-like structures) and defines a React component named "App". The component has state variables "name" and "age", and methods for handling click, mouse over, copy, and render events.

```
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    handleClick = (e) => {
22      //console.log(e.target);
23      this.setState({
24        name: 'Yoshi'
25      });
26      console.log(this.state);
27    }
28    handleMouseOver = (e) => {
29      console.log(e.target, e.pageX)
30    }
31    handleCopy = (e) => {
32      console.log('Try being original for once!');
33    }
34    render() {
```

A screenshot of a web browser window displaying a local file at `127.0.0.1:5500/index.html`. The page content includes the text "Hey, ninjas", a statement "My name is: Yoshi and I am 30", and two buttons labeled "Click Me" and "Hover me". Below the page content, the browser's developer tools are open, specifically the "Console" tab. The console output shows a warning message about using an in-browser Babel transformer and a log entry for an object with properties `name: "Ryu"` and `age: 30`, with the source being an "Inline Babel script:12".

Hey, ninjas

My name is: Yoshi and I am 30

Click Me Hover me

What we think, we become

Elements Sources Console Network ▾ ⚠ 1

top Filter Default levels Group similar 1 hidden

⚠ You are using the in-browser Babel transformer. Be [babel.min.js:24](#) sure to precompile your scripts for production - <https://babeljs.io/docs/setup/>

▶ {name: "Ryu", age: 30} Inline Babel script:12

index.html

```
12  
13     <div id="app"></div>  
14  
15     <script type="text/babel">  
16         class App extends React.Component {  
17             state = {  
18                 name: 'Ryu',  
19                 age: 30  
20             }  
21             handleClick = (e) => {  
22                 //console.log(e.target);  
23                 this.setState({  
24                     name: 'Yoshi',  
25                     age: 25  
26                 });  
27                 console.log(this.state);  
28             }  
29             handleMouseOver = (e) => {  
30                 console.log(e.target, e.pageX)  
31             }  
32             handleCopy = (e) => {  
33                 console.log('Try being original for once!');  
34             }  
35         }
```

