Events in React

React Session-5

CLARUSWAY
WAY TO REINVENT YOURSELF

Table of Contents

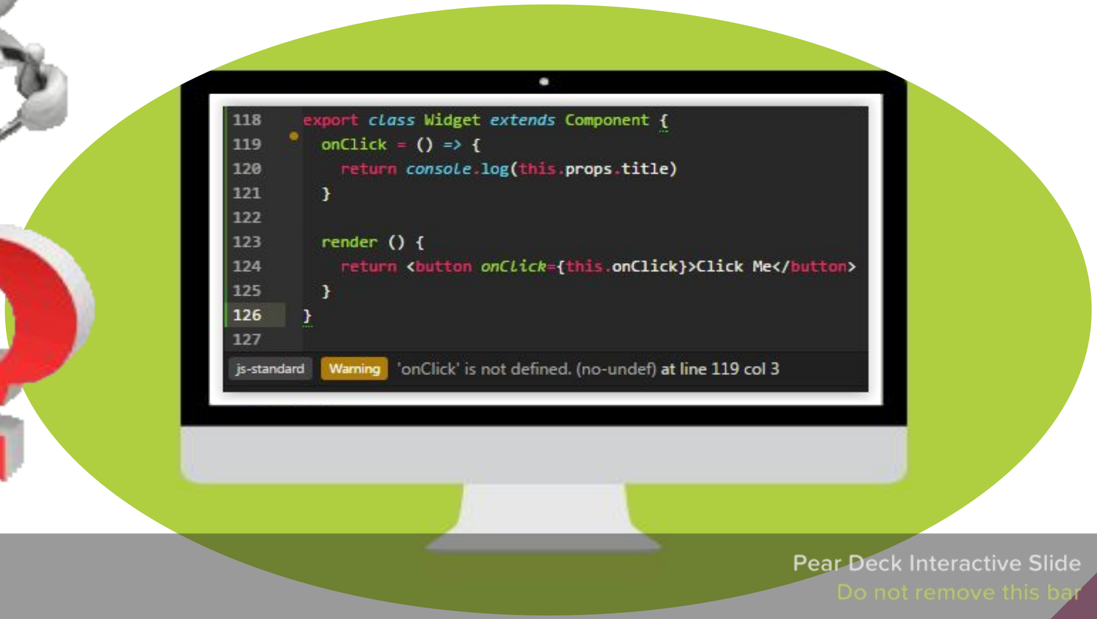
Did you finish React pre-class material?

YES NO

React JS

Kahoot!

Write React Events.



```
118  export class Widget extends Component {
119    onClick = () => {
120      return console.log(this.props.title)
121    }
122
123    render () {
124      return <button onClick={this.onClick}>Click Me</button>
125    }
126  }
127
```

'onClick' is not defined. (no-undef) at line 119 col 3

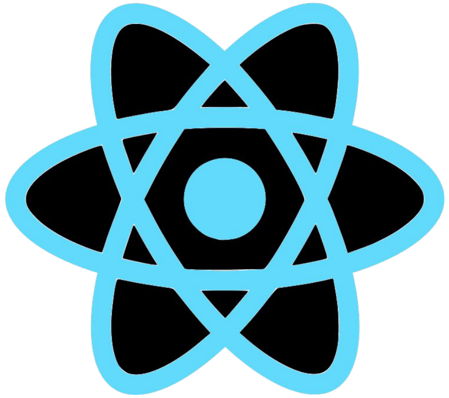
USWAY
EINVENT YOURSELF

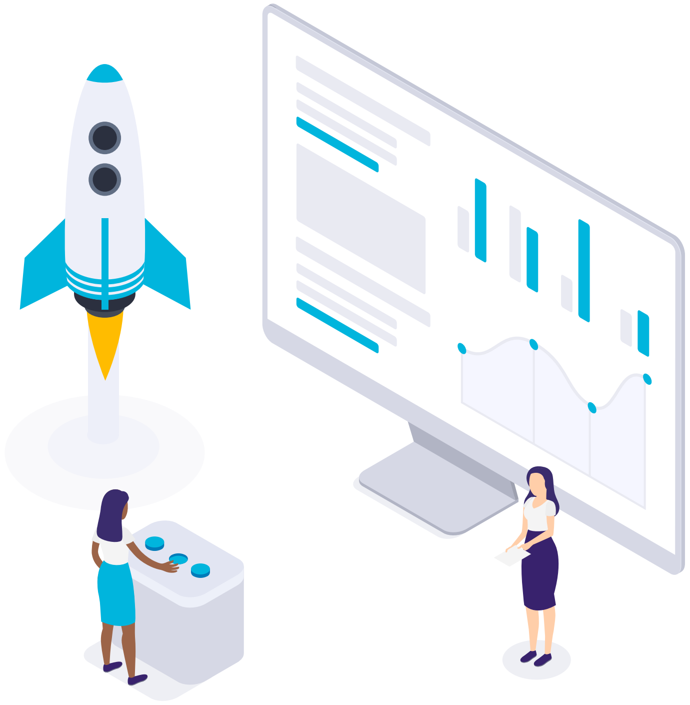
Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

1

What is React Events?





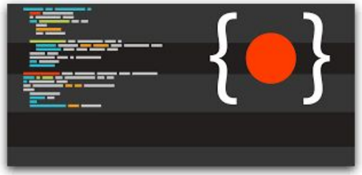
CLARUSWAY
WAY TO REINVENT YOURSELF

What is React Events?

Just like HTML, React can perform actions based on user events.

React has the same events as HTML: click, change, mouseover etc.

React events are written in camelCase syntax: `onClick` instead of `onclick`.



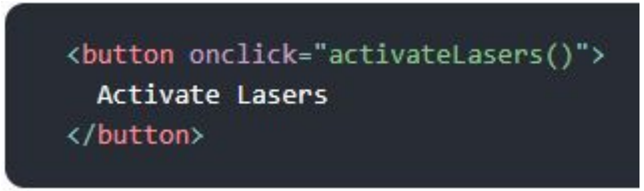
CLARUSWAY
WAY TO REINVENT YOURSELF

Differences between handling DOM events and React

Handling events with React elements is very similar to handling events on DOM elements. There are some syntax differences:

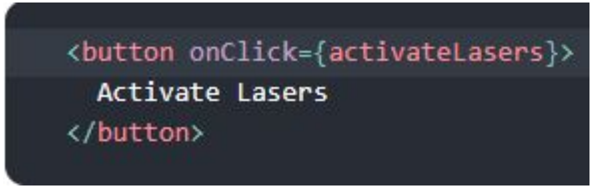
- React events are named using camelCase, rather than lowercase
- With JSX you pass a function as the event handler, rather than string

For Example, the HTML:



```
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

in React:

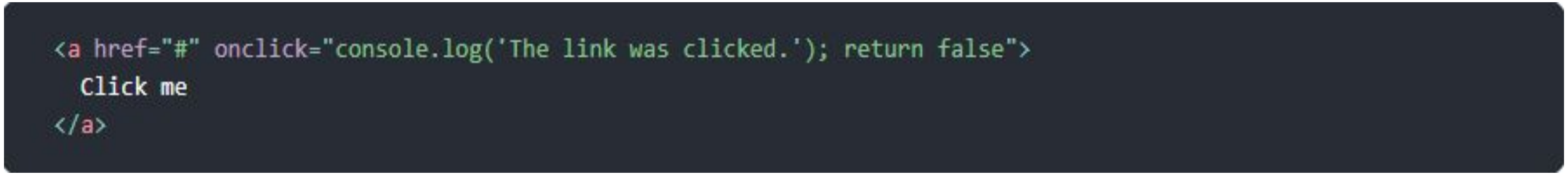


```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

CLARUSWAY
WAY TO REINVENT YOURSELF

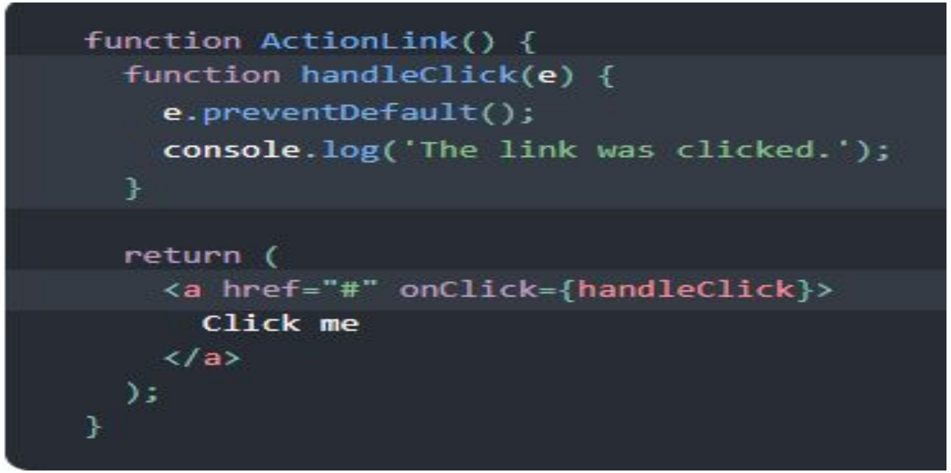
Differences between handling DOM events and React

Another difference is that you cannot return `false` to prevent default behavior in React. You must call `preventDefault` explicitly. For example, with plain HTML, to prevent the default link behavior of opening a new page, you can write:



```html
<a href="#" onclick="console.log('The link was clicked.'); return false">
  Click me
</a>
```

In React, this could instead be:



```jsx
function ActionLink() {
  function handleClick(e) {
    e.preventDefault();
    console.log('The link was clicked.');
  }

  return (
    <a href="#" onClick={handleClick}>
      Click me
    </a>
  );
}
```

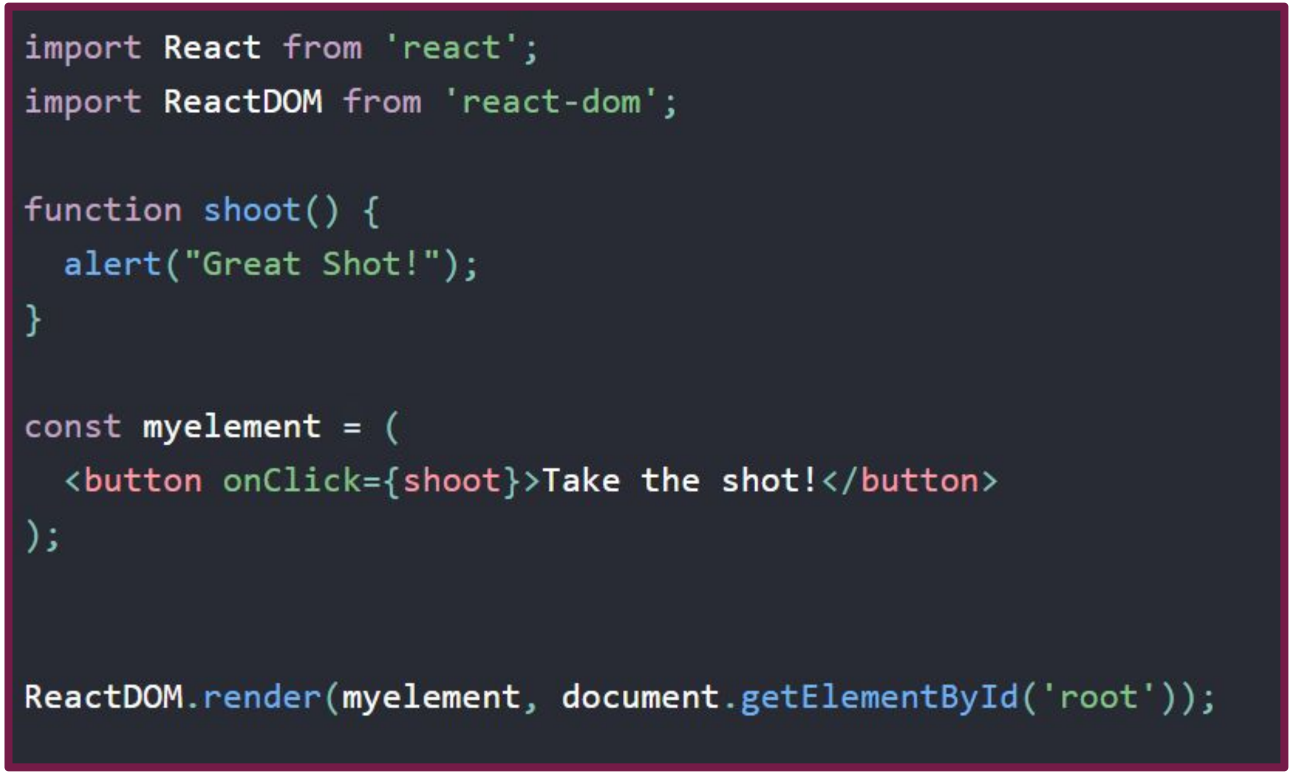
CLARUSWAY
WAY TO REINVENT YOURSELF

2 Adding Events

Adding Events

`onClick="shoot()"`

`onClick={shoot}`



```
import React from 'react';
import ReactDOM from 'react-dom';

function shoot() {
  alert("Great Shot!");
}

const myelement = (
  <button onClick={shoot}>Take the shot!</button>
);


ReactDOM.render(myelement, document.getElementById('root'));
```

CLARUSWAY
WAY TO REINVENT YOURSELF

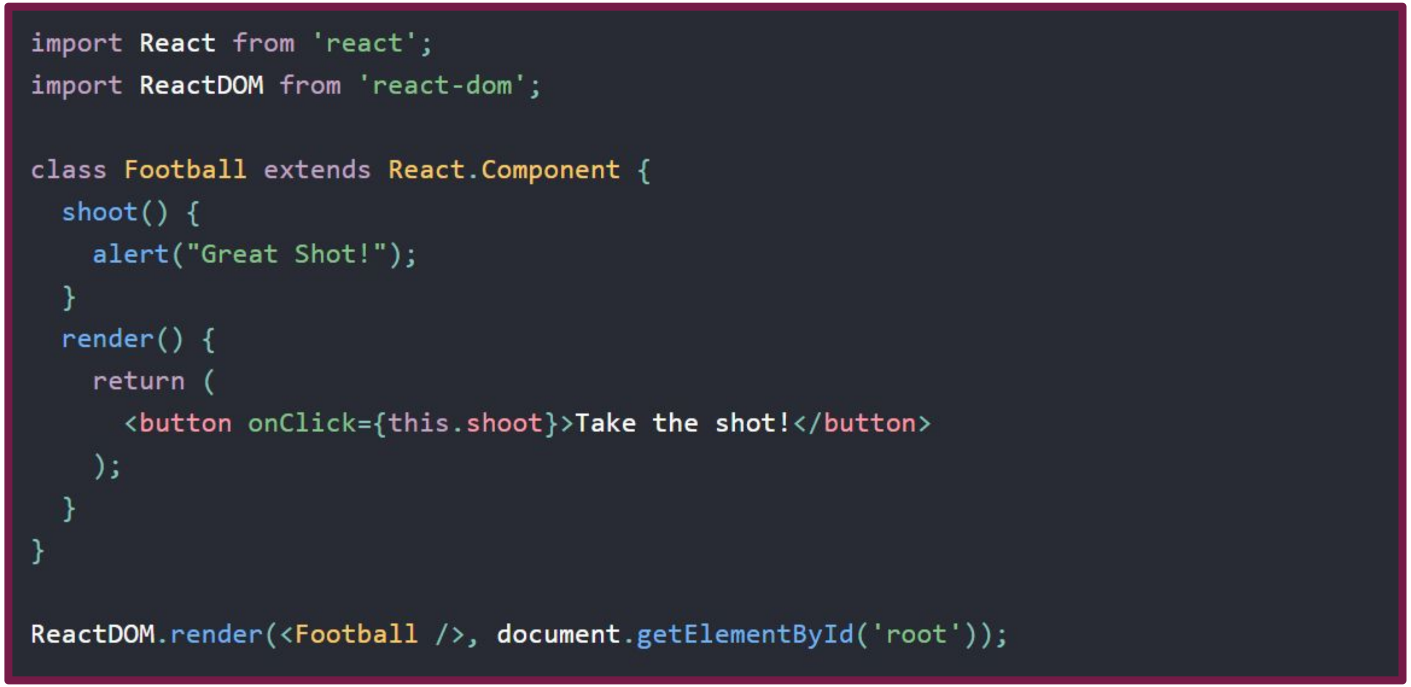
3

Event Handlers

CLARUSWAY
WAY TO REINVENT YOURSELF

Event Handlers

A good practice is to put the event handler as a method in the component class:

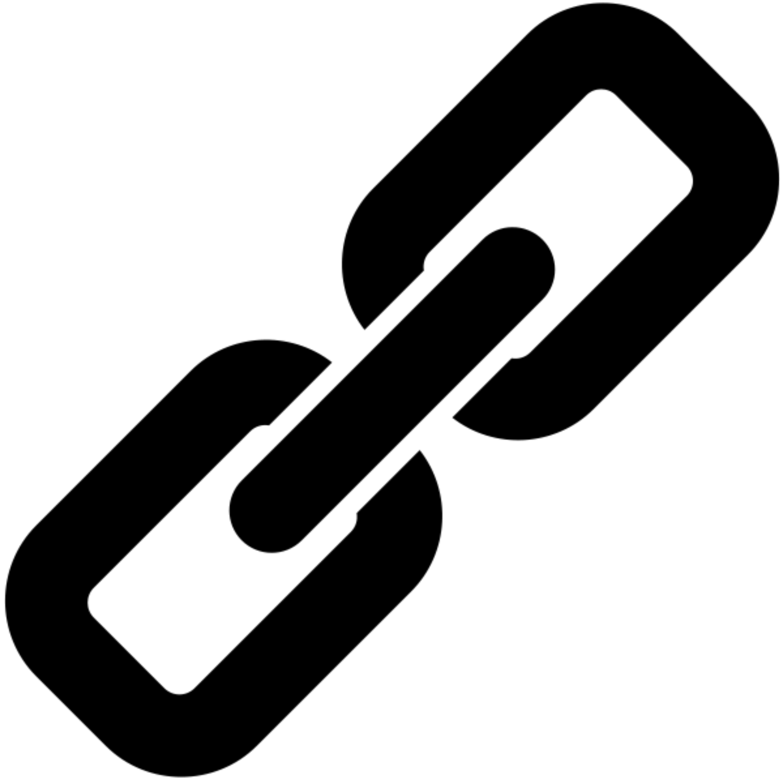


```
import React from 'react';
import ReactDOM from 'react-dom';

class Football extends React.Component {
  shoot() {
    alert("Great Shot!");
  }
  render() {
    return (
      <button onClick={this.shoot}>Take the shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

CLARUSWAY
WAY TO REINVENT YOURSELF

4

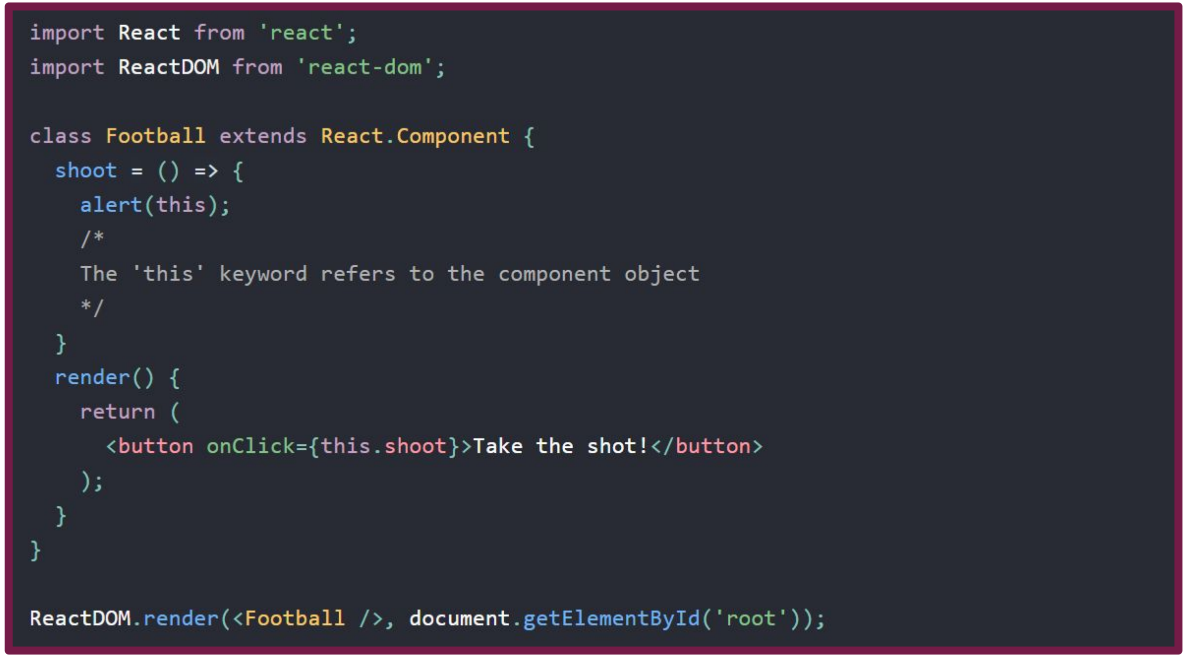
Bind `this`



CLARUSWAY
WAY TO REINVENT YOURSELF

Bind `this`

- For methods in React, the `this` keyword should represent the component that owns the method.
- That is why you should use arrow functions. With arrow functions, `this` will always represent the object that defined the arrow function.



```
import React from 'react';
import ReactDOM from 'react-dom';

class Football extends React.Component {
  shoot = () => {
    alert(this);
    /*
    The 'this' keyword refers to the component object
    */
  }
  render() {
    return (
      <button onClick={this.shoot}>Take the shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

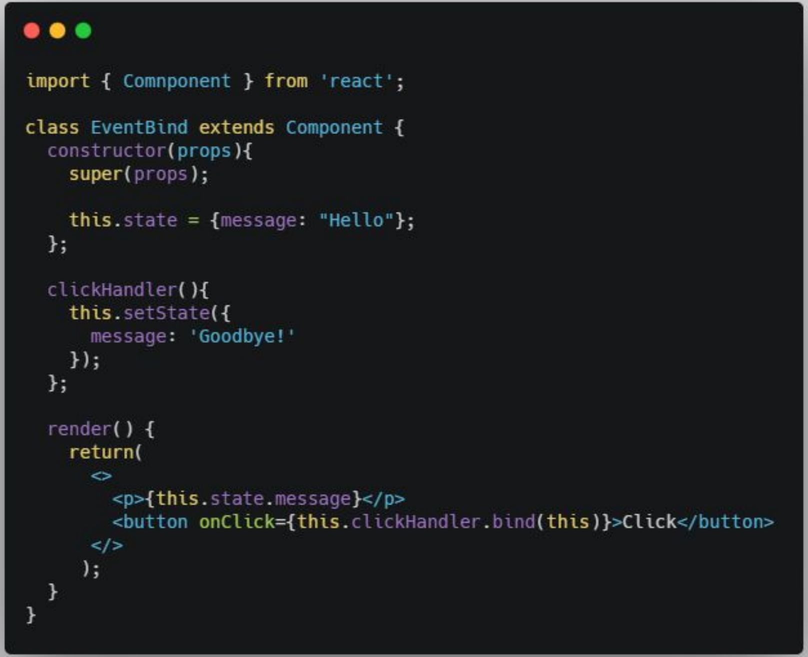
CLARUSWAY
WAY TO REINVENT YOURSELF

Approaches to bind `this` into our event handlers

1. Binding in render method.
2. Binding in render method but with arrow function.
3. Binding in constructor.
4. Class property as an arrow function.

Binding in render method

As your component or your project gets more complex, this approach is not prefered because of performance implications.

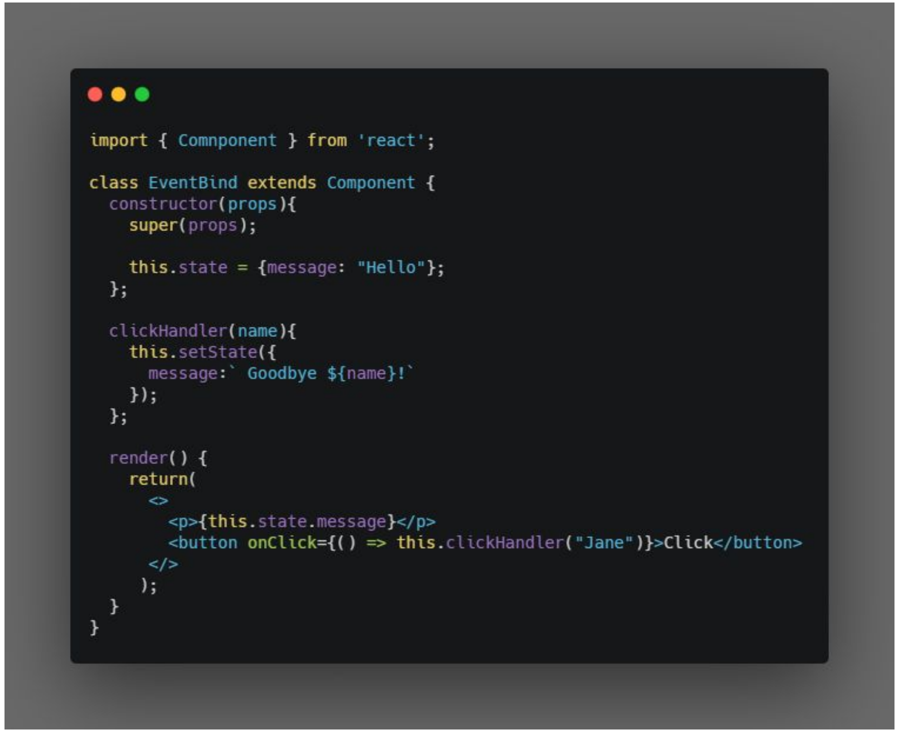


```
import { Comnponent } from 'react';

class EventBind extends Component {
  constructor(props){
    super(props);

    this.state = {message: "Hello"};
  };

  clickHandler(){
    this.setState({
      message: 'Goodbye!'
    });
  };

  render() {
    return(
      <>
        <p>{this.state.message}</p>
        <button onClick={this.clickHandler.bind(this)}>Click</button>
      </>
     );
  }
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Binding in render method with arrow functions

This approach will work well, if you want to pass argument to your event handler functions

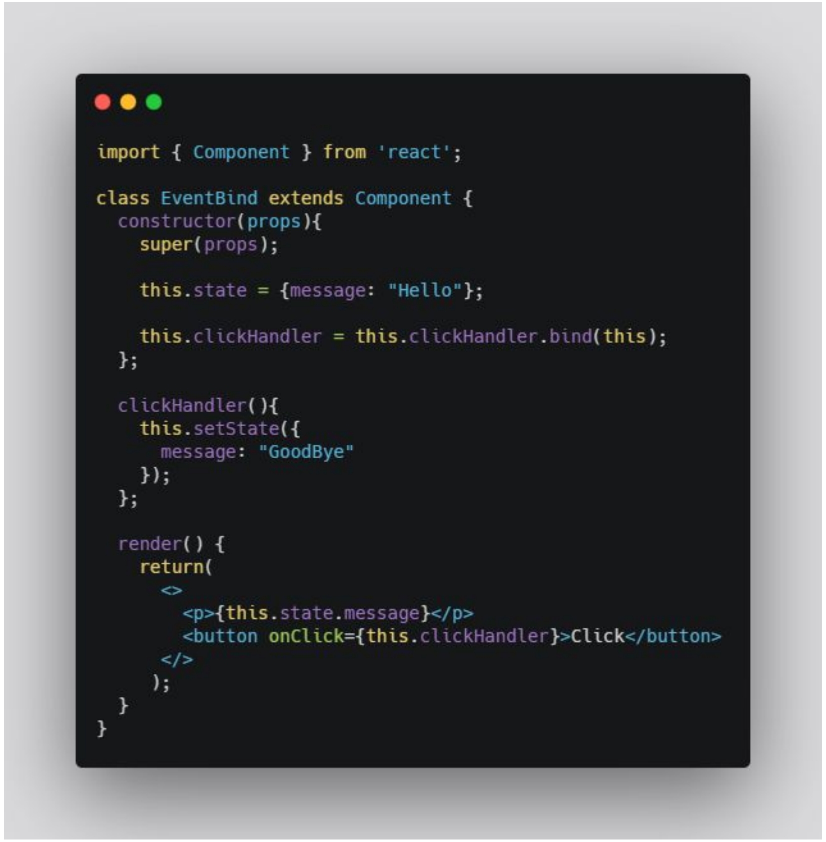


```
import { Comnponent } from 'react';

class EventBind extends Component {
  constructor(props){
    super(props);

    this.state = {message: "Hello"};
  };

  clickHandler(name){
    this.setState({
      message:` Goodbye ${name}!`
    });
  };

  render() {
    return(
      <>
        <p>{this.state.message}</p>
        <button onClick={() => this.clickHandler("Jane")}>Click</button>
      </>
     );
  }
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Binding in the constructor

This is the approach in the official react documentation, and it is the best option right now.

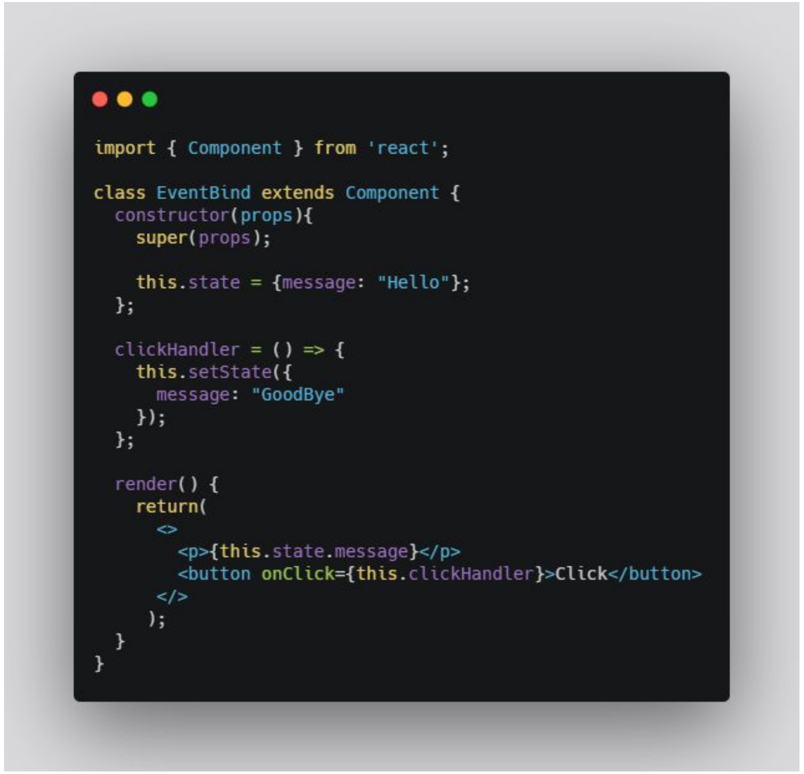


```
import { Component } from 'react';

class EventBind extends Component {
  constructor(props){
    super(props);

    this.state = {message: "Hello"};

    this.clickHandler = this.clickHandler.bind(this);
  };

  clickHandler(){
    this.setState({
      message: "GoodBye"
    });
  };

  render() {
    return(
      <>
        <p>{this.state.message}</p>
        <button onClick={this.clickHandler}>Click</button>
      </>
    );
  }
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Class property as an arrow function

This is the approach which is basically change the way you define your method in the class.



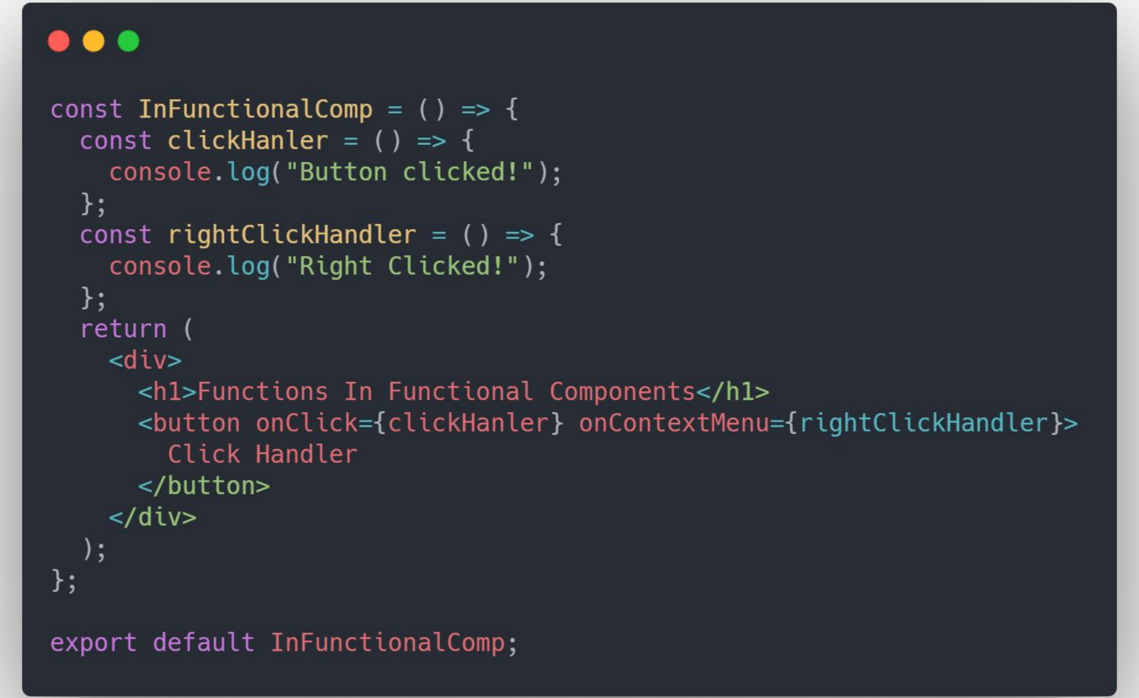
```
import { Component } from 'react';

class EventBind extends Component {
  constructor(props){
    super(props);

    this.state = {message: "Hello"};
  };

  clickHandler = () => {
    this.setState({
      message: "GoodBye"
    });
  };

  render() {
    return(
      <>
        <p>{this.state.message}</p>
        <button onClick={this.clickHandler}>Click</button>
      </>
    );
  }
}
```

CLARUSWAY
WAY TO REINVENT YOURSELF

5

Functions in Class & Functional Components

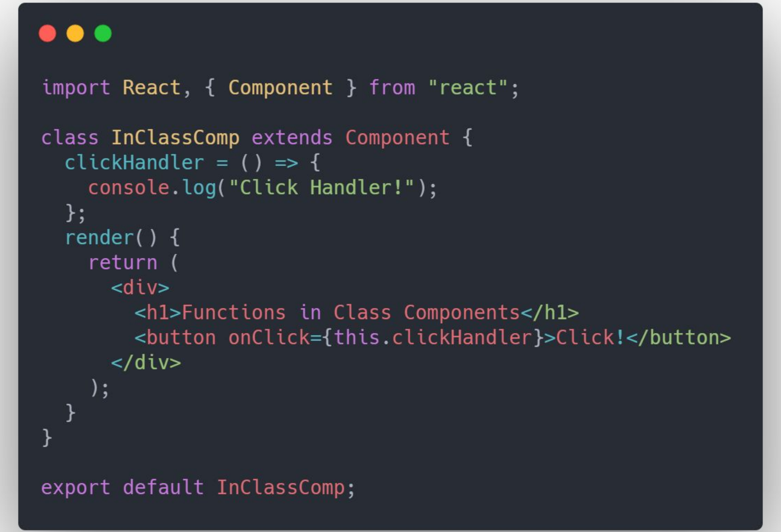
Functions in Func. Comp.



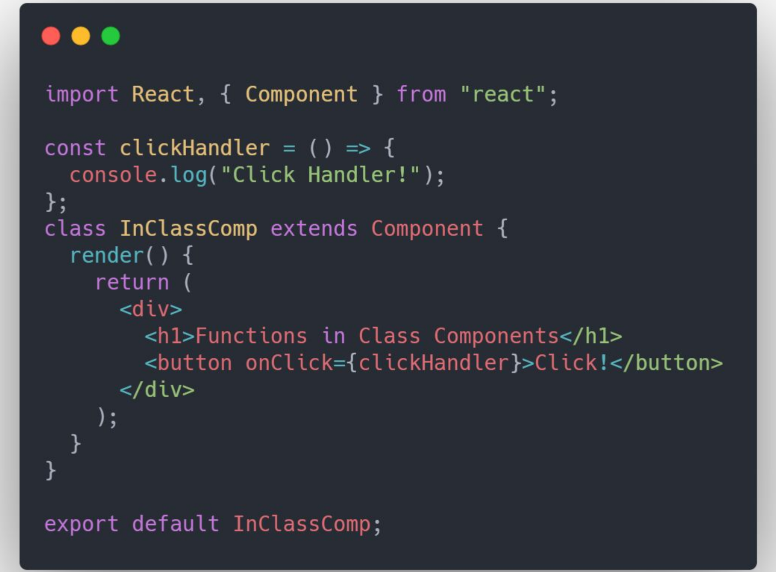
```
const InFunctionalComp = () => {
  const clickHanler = () => {
    console.log("Button clicked!");
  };
  const rightClickHandler = () => {
    console.log("Right Clicked!");
  };
  return (
    <div>
      <h1>Functions In Functional Components</h1>
      <button onClick={clickHanler} onContextMenu={rightClickHandler}>
        Click Handler
      </button>
    </div>
  );
};

export default InFunctionalComp;
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Functions in Class Comp.



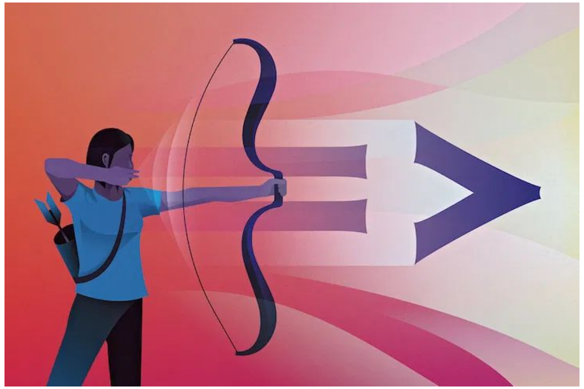
```
import React, { Component } from "react";

class InClassComp extends Component {
  clickHandler = () => {
    console.log("Click Handler!");
  };
  render() {
    return (
      <div>
        <h1>Functions in Class Components</h1>
        <button onClick={this.clickHandler}>Click!</button>
      </div>
    );
  }
}

export default InClassComp;
```



```
import React, { Component } from "react";

const clickHandler = () => {
  console.log("Click Handler!");
};
class InClassComp extends Component {
  render() {
    return (
      <div>
        <h1>Functions in Class Components</h1>
        <button onClick={clickHandler}>Click!</button>
      </div>
    );
  }
}

export default InClassComp;
```

5

Why Arrow Functions?

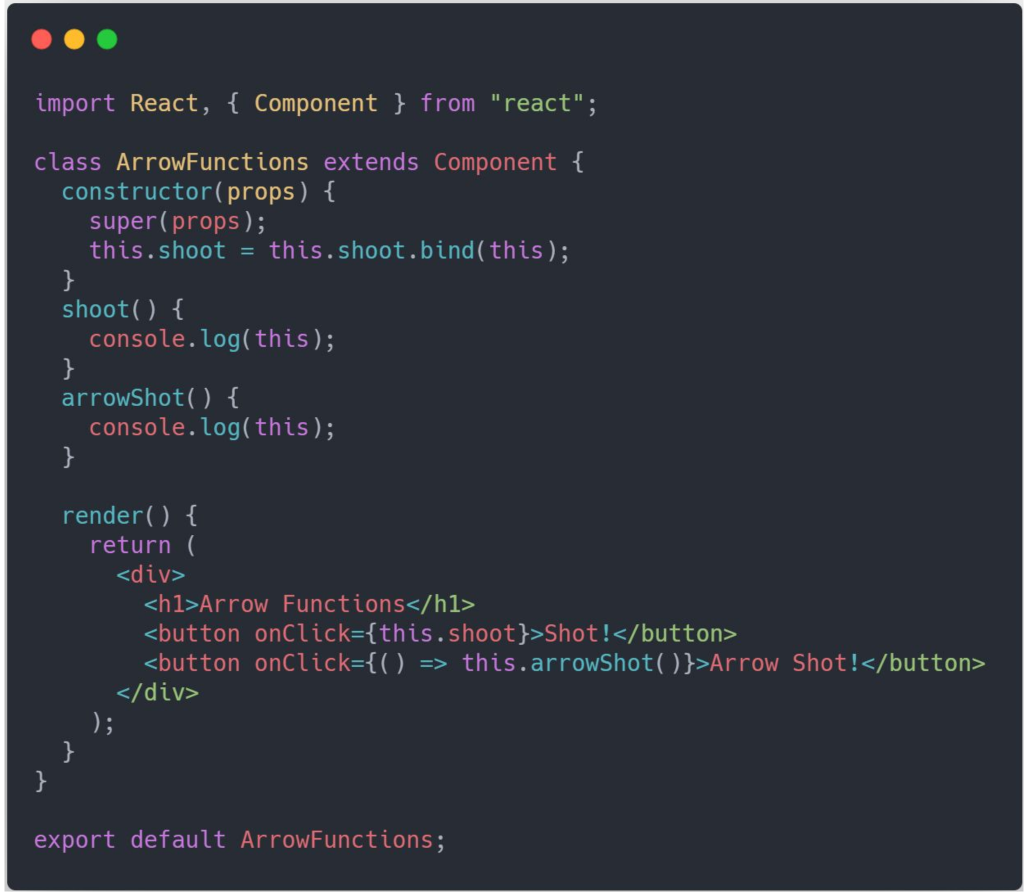


CLARUSWAY
WAY TO REINVENT YOURSELF

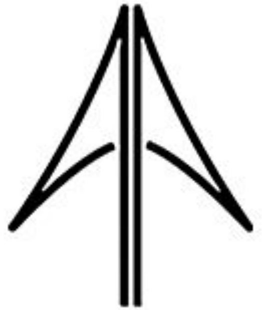
Why Arrow Functions?

- In class components, the `this` keyword is not defined by default, so with regular functions the `this` keyword represents the object that called the method, which can be the global window object, a HTML button, or whatever.
- If you *must* use regular functions instead of arrow functions you have to bind `this` to the component instance using the `bind()` method:
- Without the binding, the `this` keyword would return `undefined`.

Arrow Functions In Class Comp.

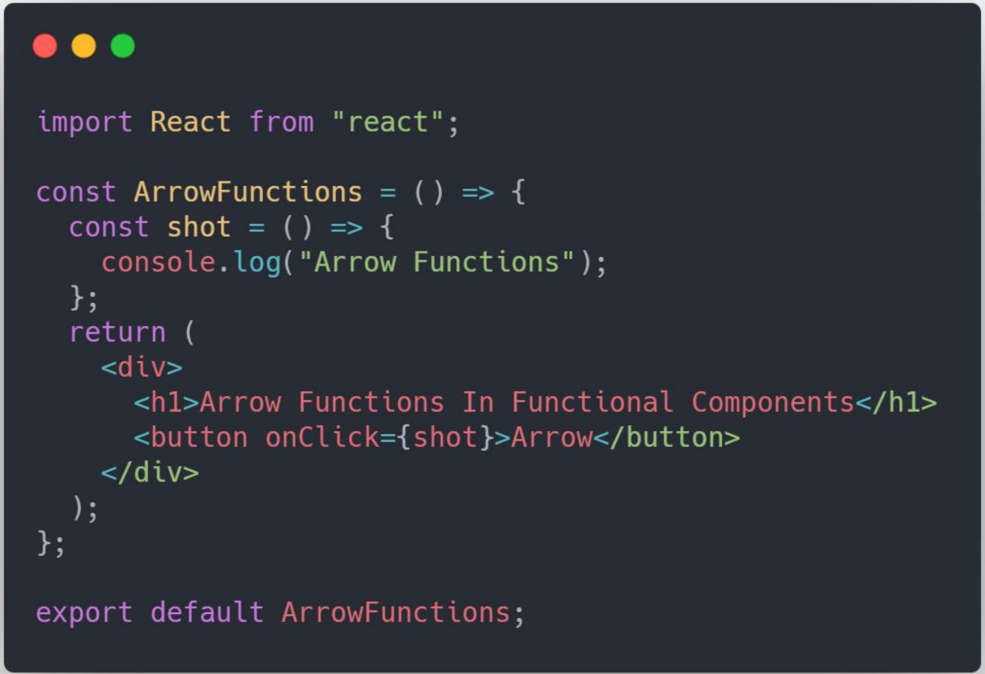


```
import React, { Component } from "react";

class ArrowFunctions extends Component {
  constructor(props) {
    super(props);
    this.shoot = this.shoot.bind(this);
  }
  shoot() {
    console.log(this);
  }
  arrowShot() {
    console.log(this);
  }

  render() {
    return (
      <div>
        <h1>Arrow Functions</h1>
        <button onClick={this.shoot}>Shot!</button>
        <button onClick={() => this.arrowShot()}>Arrow Shot!</button>
      </div>
    );
  }
}

export default ArrowFunctions;
```

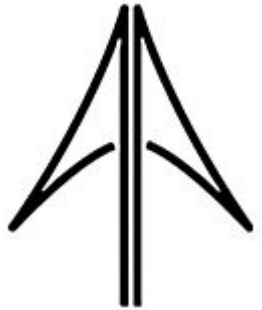


CLARUSWAY
WAY TO REINVENT YOURSELF

Arrow Functions In Func Comp.



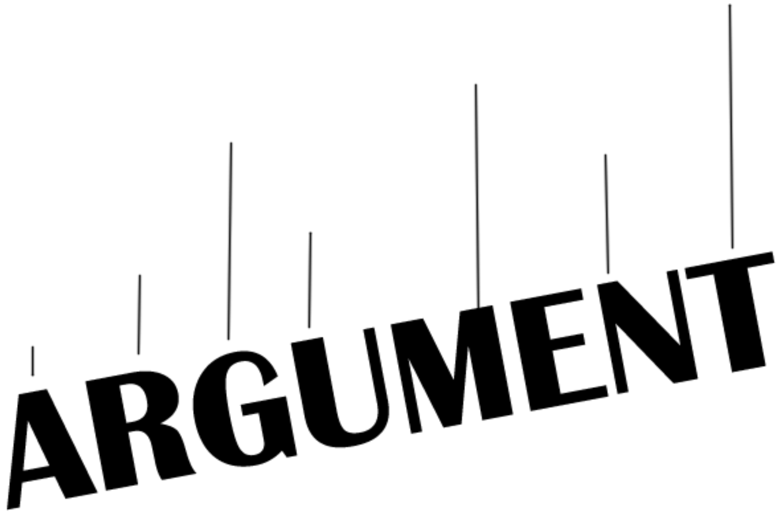
```
import React from "react";

const ArrowFunctions = () => {
  const shot = () => {
    console.log("Arrow Functions");
  };
  return (
    <div>
      <h1>Arrow Functions In Functional Components</h1>
      <button onClick={shot}>Arrow</button>
    </div>
  );
};

export default ArrowFunctions;
```



CLARUSWAY
WAY TO REINVENT YOURSELF

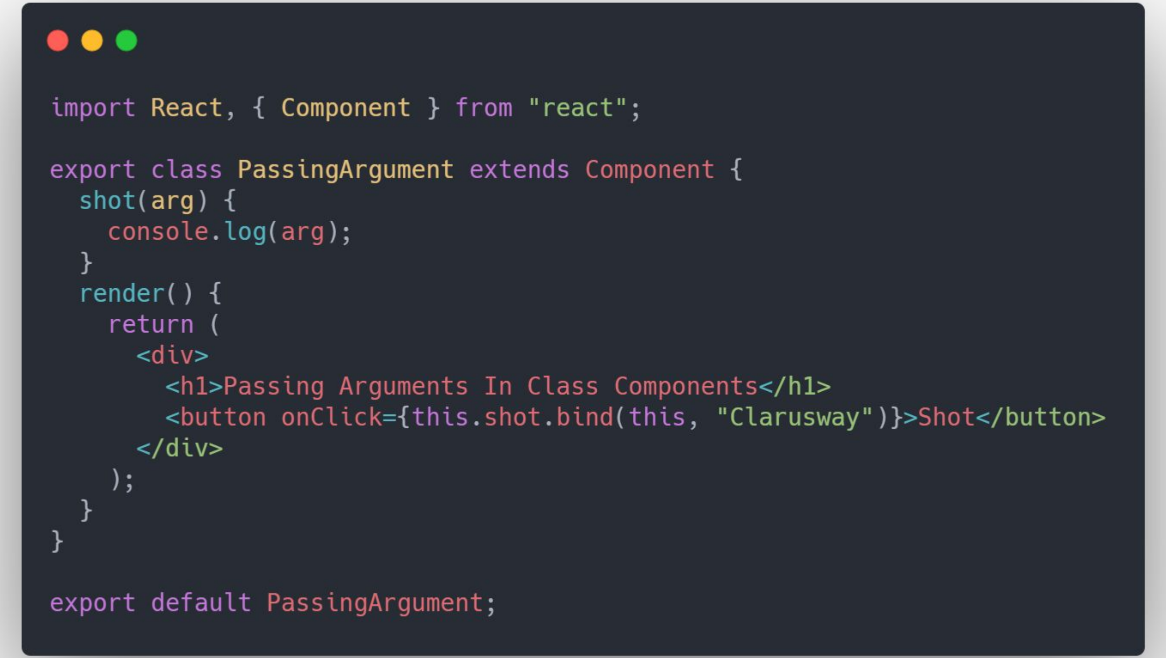
6

Passing Arguments



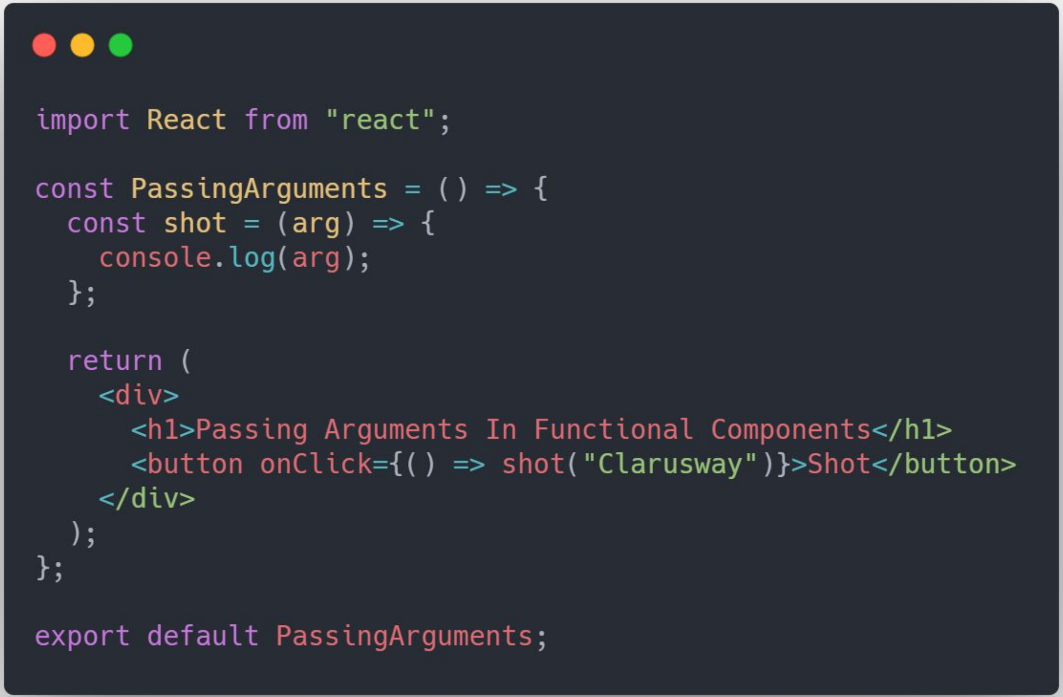
Passing Arguments In C.C.

2. Bind the event handler to `this`.
Note that the first argument has to be `this`.



```
import React, { Component } from "react";

export class PassingArgument extends Component {
  shot(arg) {
    console.log(arg);
  }
  render() {
    return (
      <div>
        <h1>Passing Arguments In Class Components</h1>
        <button onClick={this.shot.bind(this, "Clarusway")}>Shot</button>
      </div>
    );
  }
}

export default PassingArgument;
```

Passing Arguments In F.C.



```
import React from "react";

const PassingArguments = () => {
  const shot = (arg) => {
    console.log(arg);
  };

  return (
    <div>
      <h1>Passing Arguments In Functional Components</h1>
      <button onClick={() => shot("Clarusway")}>Shot</button>
    </div>
  );
};

export default PassingArguments;
```

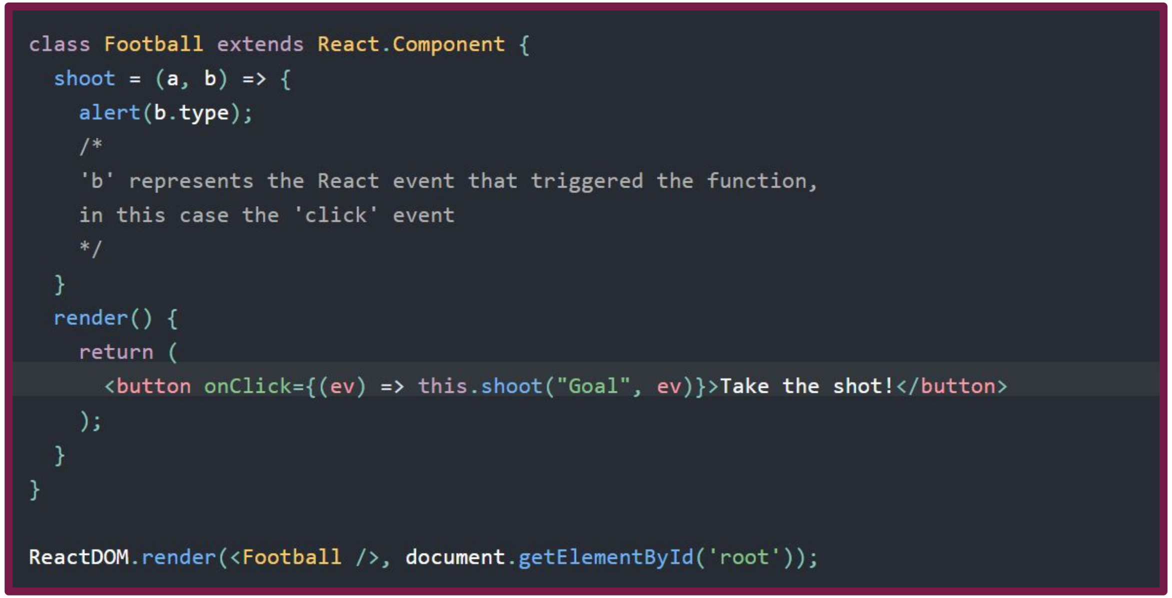
CLARUSWAY
WAY TO REINVENT YOURSELF

7

React Event Object

React Event Object

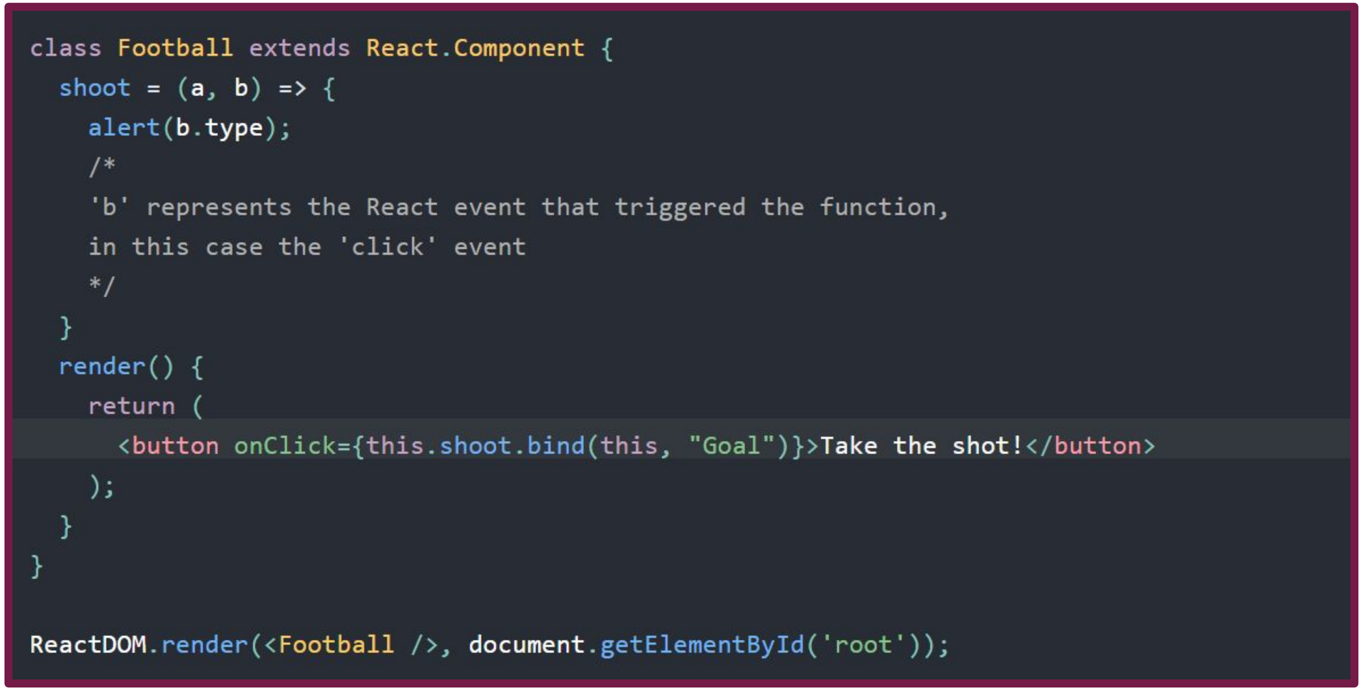
- Event handlers have access to the React event that triggered the function.
- In our example the event is the "click" event. Notice that once again the syntax is different when using arrow functions or not.
- With the arrow function you have to send the event argument manually:



```
class Football extends React.Component {
  shoot = (a, b) => {
    alert(b.type);
    /*
    'b' represents the React event that triggered the function,
    in this case the 'click' event
    */
  }
  render() {
    return (
      <button onClick={(ev) => this.shoot("Goal", ev)}>Take the shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

React Event Object

- Without arrow function, the React event object is sent automatically as the last argument when using the `bind()` method:



```
class Football extends React.Component {
  shoot = (a, b) => {
    alert(b.type);
    /*
    'b' represents the React event that triggered the function,
    in this case the 'click' event
    */
  }
  render() {
    return (
      <button onClick={this.shoot.bind(this, "Goal")}>Take the shot!</button>
    );
  }
}

ReactDOM.render(<Football />, document.getElementById('root'));
```

CLARUSWAY
WAY TO REINVENT YOURSELF

THANKS!

Any questions?