Introduction to Events and event Handling in React **Day**41/100

React elements have properties that are based on the standard DOM events. React elements are able to fire all the same events as the usual DOM elements. They are named using camelCase so don't forget to use this convention on React Elements. So if you want to use onclick, it will actually be onClick. This is quite similar to how instead of using class, you need to use className. React handles a lot of different types of events, but we'll stick to the basics here.

Note that:

- React events are named using camelCase, rather than lowercase.

- With JSX you pass a function as the event handler, rather than a string.

HTML Syntax:

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

React syntax:

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

React has a unique approach to handling events: declaring event handlers in JSX. The differentiating factor with event handling in React components is that it's declarative. Contrast this with something like jQuery, where you have to write imperative code that selects the relevant DOM elements and attaches event handler functions to them.

The advantage with the declarative approach to event handlers in JSX markup is that they're part of the UI structure. Not having to track down the code that assigns event handlers is mentally liberating.

Declaring events handlers.

In this section, you'll write a basic event handler, so that you can get a

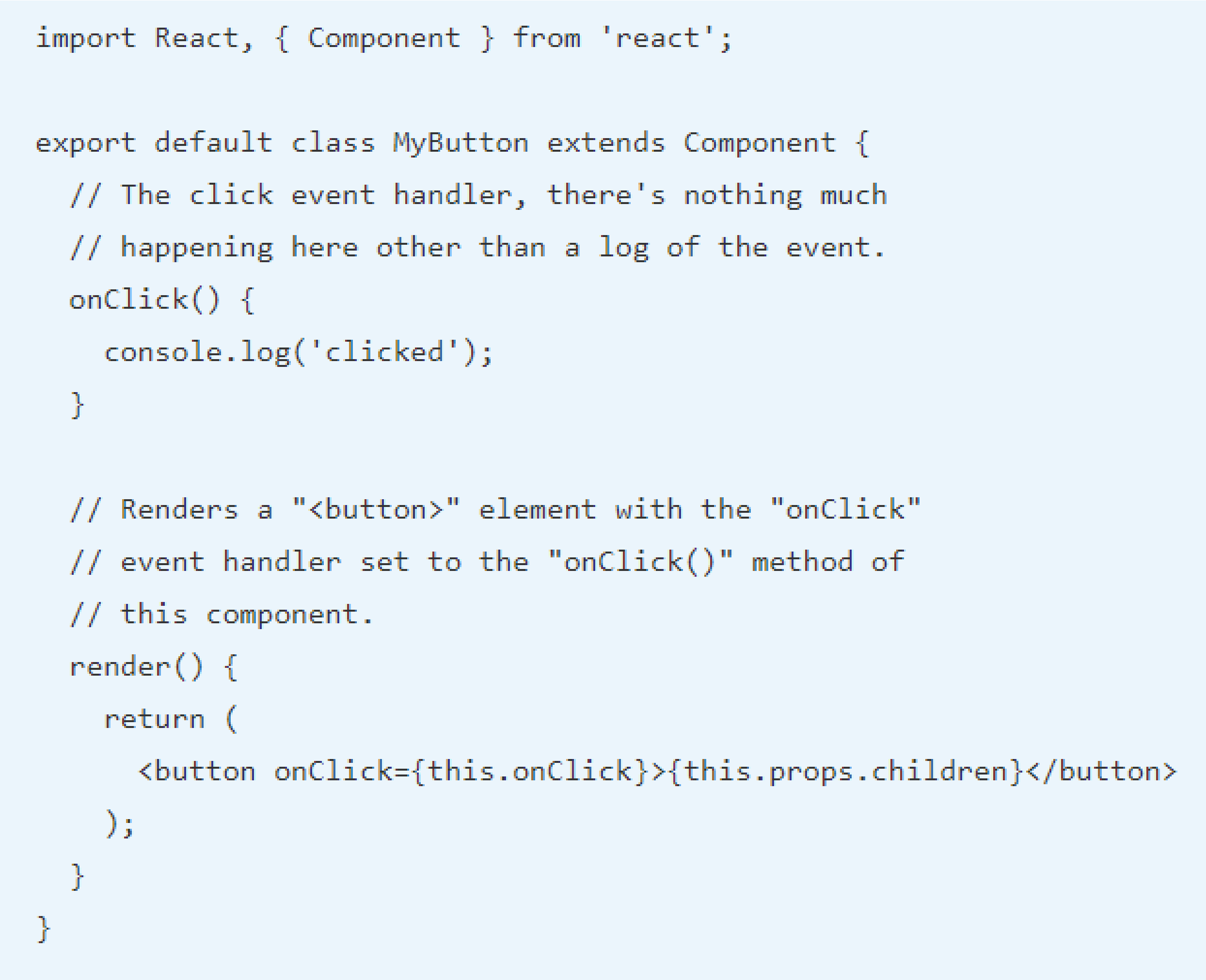
feel for the declarative event handling syntax found in React applications.

Then, we will see how to use generic event handler functions

Declaring handler function.

The event handler function, this.onClick(), is passed to the onClick property of the element. By looking at the markup below, it's clear what code is going to run when the button is clicked.

You can find the file on GitHub.
Link: https://github.com/HarunHM/React-Demo/blob/master/MyButton.js



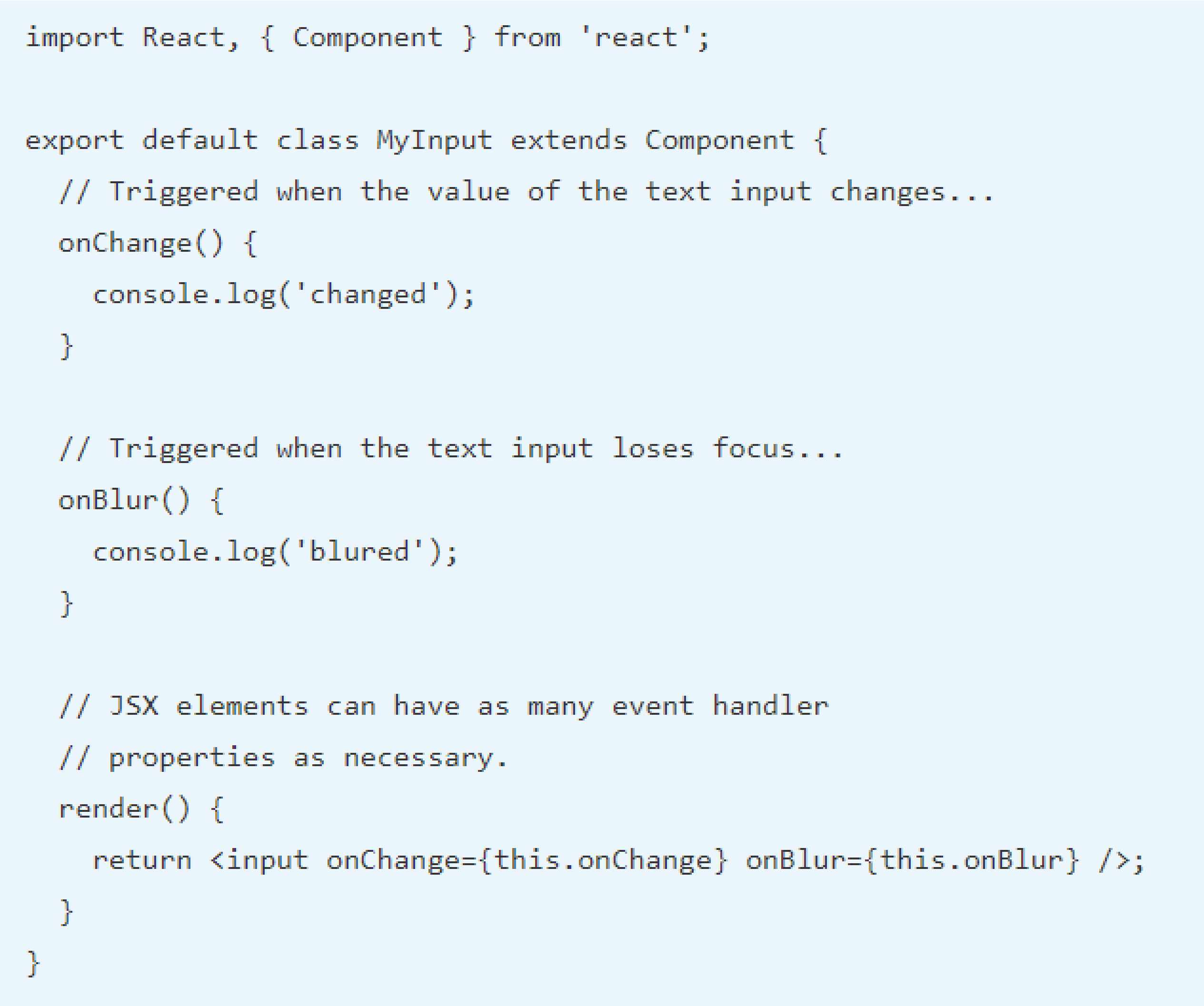
```
import React, { Component } from 'react';

export default class MyButton extends Component {
  // The click event handler, there's nothing much
  // happening here other than a log of the event.
  onClick() {
    console.log('clicked');
  }

  // Renders a "<button>" element with the "onClick"
  // event handler set to the "onClick()" method of
  // this component.
  render() {
    return (
      <button onClick={this.onClick}>{this.props.children}</button>
    );
  }
}
```

Multiple event handlers.

What I really like about the declarative event handler syntax is that it’s easy to read when there’s more than one handler assigned to an element. Sometimes, for example, there are two or three handlers for an element. Imperative code is difficult to work with for a single event handler, let alone several of them.

When an element needs more handlers, it’s just another JSX attribute. This scales well from a code maintainability perspective.

You can find the file on GitHub.
Link: https://github.com/HarunHM/React-Demo/blob/master/src/MyInput.js



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

export default class MyInput extends Component {
  // Triggered when the value of the text input changes...
  onChange() {
    console.log('changed');
  }

  // Triggered when the text input loses focus...
  onBlur() {
    console.log('blured');
  }

  // JSX elements can have as many event handler
  // properties as necessary.
  render() {
    return <input onChange={this.onChange} onBlur={this.onBlur} />;
  }
}
```

This input element could have several more event handlers, and the code would be just as readable.

As you keep adding more event handlers to your components, you’ll notice that a lot of them do the same thing. Next, we’ll learn how to share generic handler functions across components.

Importing generic handlers.

Any React application is likely going to share the same event handling logic for different components. For example, in response to a button click, the component should sort a list of items. It’s these types of generic behaviors that belong in their own modules so that several components can share them.

https://www.youtube.com/watch?v=tAmil2Ed2M8

https://www.youtube.com/watch?v=OcM__8q6p4

https://reactjs.org/docs/handling-events.html

Best Wishes Lux Tech Academy.