2. React Events

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

1. React events are named using **camelCase**, rather than **lowercase**.

Example:

HTML	JSX
onclick	onClick
onblur	onBlur
onchange	onChange

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

Example:

HTML

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

JSX

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

We should not call the function when we add an event in JSX.

JSX

```
class MyComponent extends Component {
  handleClick = () => {
    console.log("clicked")
  }
```

```
  render() {
    return <button onClick={this.handleClick()}>Click Me</button>
  }
}
```

In the above function, the

`handleClick` is called instead of passed as a reference.

JSX

```jsx
class MyComponent extends Component {
  handleClick = () => {
    console.log("clicked")
  }
  render() {
    return <button onClick={this.handleClick}>Click Me</button>
  }
}
```

In the above function, the

`handleClick` is passed as a reference. So, the function is not being called every time the component renders.

Providing Arrow Functions

To not change the context of

`this` , we have to pass an arrow function to the event.

JSX

```jsx
class MyComponent extends Component {
  handleClick() {
    console.log(this)  // undefined
  }
  render() {
    return <button onClick={this.handleClick}>Click Me</button>
  }
}
```

JSX

```jsx
class MyComponent extends Component {
  handleClick = () => {
    console.log(this)  // MyComponent {...}
  }
  render() {
    return <button onClick={this.handleClick}>Click Me</button>
  }
}
```

3