Event Handler

Learn how to define event handlers in a React app. You will also get to learn the importance of using Arrow function inside event handlers.

Now we'll cover event handlers in elements. In the application that you are building, you are using the following button element to dismiss an item from the list.

```
...

<button
  onClick={() => this.onDismiss(item.objectID)}
  type="button"
>
  Dismiss
</button>

...
```

This function is already complex because it passes a value to the class method and has to wrap it in another (arrow) function. Essentially, it has to be a function that is passed to the event handler. The code given below wouldn't work if you are following along on a local React setup, because the class method would be executed immediately when you open the application in the browser:

```
...

<button
  onClick={this.onDismiss(item.objectID)}
  type="button"
>
  Dismiss
</button>

...
```

`onClick`

When using `onClick={doSomething()}`, the `doSomething()` function executes

When using `onClick={doSomething()}`, the `doSomething()` function executes immediately when the application is opened in a browser. The expression in the handler is evaluated. Since the returned value of the function isn't a function anymore, nothing would happen when you click the button. But using `onClick={doSomething}` where `doSomething` is a function, it would only be executed if the button is clicked.

The same rules apply for the `onDismiss()` class method.

However, using `onClick={this.onDismiss}` wouldn't suffice, because the `item.objectID` property needs to be passed to the class method to identify the item that should be dismissed. We wrap it into another function to sneak in the property. This concept is called higher-order functions in JavaScript, which we will cover briefly later.

```
...

<button
  onClick={() => this.onDismiss(item.objectID)}
  type="button"
>
  Dismiss
</button>

...
```

We can also define wrapping function outside the method, to pass only the defined function to the handler. Since it needs access to the individual item, it has to live inside the map function block.

```
class App extends Component {

  ...

  render() {
    return (
      <div className="App">
        {this.state.list.map(item => {
          const onHandleDismiss = () =>
            this.onDismiss(item.objectID);

          return (
            <div key={item.objectID}>
              <span>
                <a href={item.url}>{item.title}</a>
              </span>
              <span>{item.author}</span>
              <span>{item.num_comments}</span>
              <span>{item.points}</span>
```

```
              <span>
                <button
                  onClick={onHandleDismiss}
                  type="button"
                >
                  Dismiss
                </button>
              </span>
            </div>
          );
        }
        )}
      </div>
    );
  }
}
```

A function has to be passed to the element's handler. As an example, try this code instead:

```
class App extends Component {

  ...

  render() {
    return (
      <div className="App">
        {this.state.list.map(item =>
            ...
            <span>
              <button
                onClick={console.log(item.objectID)}
                type="button"
              >
                Dismiss
              </button>
            </span>
          </div>
        )}
      </div>
    );
  }
}
```

This method will run when you open your application in the browser, but not when you click the button. The following code would only run when you click the button, a function that is executed when you trigger the handler:

```
...

<button
  onClick={function () {
    console.log(item.objectID)
  }}
```

```
  type="button"
>
  Dismiss
</button>

...
```

Remember: You can transform functions into a JavaScript ES6 arrow function, just as we did with the `onDismiss()` class method:

```
...

<button
  onClick={() => console.log(item.objectID)}
  type="button"
>
  Dismiss
</button>

...
```

Newcomers to React often have difficulty using functions in event handlers, so don't get discouraged if you have trouble on the first pass. You should end up with an inlined JavaScript ES6 arrow function with access to the `objectID` property of the `item` object:

```
class App extends Component {
  ...

  render() {
    return (
      <div className="App">
        {this.state.list.map(item =>
          <div key={item.objectID}>
            ...
            <span>
              <button
                onClick={() => this.onDismiss(item.objectID)}
                type="button"
              >
                Dismiss
              </button>
            </span>
          </div>
        )}
      </div>
    );
  }
}
```

Using arrow functions in event handlers directly impacts your application's

performance. For instance, the `onClick` handler for the `onDismiss()` method wraps the method in another arrow function to pass the item identifier. Every time the `render()` method runs, the handler instantiates the higher-order arrow function. It can have an impact on your application performance, but in most cases you won't notice. If you have a huge table of data with 1000 items and each row or column has an arrow function in an event handler, it is worth thinking about the performance implications, so you could implement a dedicated Button component to bind the method in the constructor. Before that, though, it is premature optimization, and it is more prudent learn the basics of React before thinking about optimization.

Exercises:

- Try the different approaches of using functions in the `onClick` handler of your button