Higher-Order Functions in React

This lesson introduces the key concept of functional programming: High-Order functions, along with their application in React with the help of coding examples.

Higher-order functions are a great programming concept, especially when moving towards functional programming. In React, it makes total sense to know about these kinds of functions, because at some point you have to deal with higher-order components which can be explained best when knowing about higher-order functions in the first place.

Higher-order functions can be showcased in React early on without introducing higher-order components. For instance, let's say a rendered list of users can be filtered based on the value of an input field.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

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

It's not always desired to extract functions, because it can add unnecessary complexity, but on the other side, it can have beneficial learning effects for JavaScript. In addition, by extracting a function, you make it testable in isolation from the React component. So let's showcase it with the function which is provided to the built-in filter function.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

ReactDOM.render(
  <App />,
```

```
  document.getElementById('root')
);
```

The previous implementation doesn't work because the `doFilter()` function needs to know about the query property from the state. So you can pass it to the function by wrapping it with another function which leads to a higher-order function.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

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

Basically, a Higher-Order function is a function which returns a function. By using JavaScript ES6 arrow functions, you can make a higher-order function more concise. Furthermore, this shorthand version makes it more attractive composing functions into functions.

```
import React from 'react';
require('./style.css');

import ReactDOM from 'react-dom';
import App from './app.js';

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

Now the `doFilter()` function can be exported from the file and tested in isolation as pure (higher-order) function. After learning about higher-order functions, all the fundamental knowledge is established to learn more about React's higher-order components.

Extracting these functions into (higher-order) functions outside of a React component can be beneficial for testing React's local state management in isolation as well.

```
import React from 'react';
require('./style.css');
```

```
import ReactDOM from 'react-dom';
import Counter from './Counter.js';


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

Moving functions around your code base is a great way of learning about the benefits of having functions as first class citizens in JavaScript. It’s powerful when moving your code towards functional programming.