There is more JavaScript than React

Final notes by the author to his readers!

In conclusion, there is lots of JavaScript which can be harnessed in React. Whereas React has only a slim API surface area, developers have to get used to all the functionalities JavaScript has to offer. The saying is not without any reason: “being a React developer makes you a better JavaScript developer”. Let’s recap some of the learned aspects of JavaScript in React by refactoring a higher-order component.

```
function withLoading(Component) {
  return class WithLoading extends {
    render() {
      const { isLoading, ...props } = this.props;

      if (isLoading) {
        return <p>Loading</p>;
      }

      return <Component { ...props } />;
    }
  }
  };
}
```

This higher-order component is only used for showing a conditional loading indicator when the isLoading prop is set to true. Otherwise it renders the input component. You can already see the (rest) destructuring and the spread operator in action. The latter can be seen for the rendered Component, because all the remaining properties from the props object are passed to the Component.

The first step for making the higher-order component more concise is refactoring the returned React class component to a functional stateless component:

```
function withLoading(Component) {
  return function ({ isLoading, ...props }) {
```

```
    if (isLoading) {
      return <p>Loading</p>;
    }

    return <Component { ...props } />;
  };
}
```

You can see that the rest destructuring can be used in the function's signature too. Next, using JavaScript ES6 arrow functions makes the higher-order component more concise again:

```
const withLoading = Component => ({ isLoading, ...props }) => {
  if (isLoading) {
    return <p>Loading</p>;
  }

  return <Component { ...props } />;
}
```

And adding the ternary operator shortens the function body into one line of code. Thus the function body can be left out and the return statement can be omitted.

```
const withLoading = Component => ({ isLoading, ...props }) =>
  isLoading
    ? <p>Loading</p>
    : <Component { ...props } />
```

As you can see, the higher-order component uses various JavaScript and not React relevant techniques: arrow functions, higher-order functions, a ternary operator, destructuring and the spread operator. That's how JavaScript's functionalities can be used in React applications.

Conslusion:

Often people say that learning React has a steep learning curve. But it hasn't when only leaving React in the equation and leaving all the JavaScript out of it. React doesn't add any foreign abstraction layer on top as other web frameworks are doing it. Instead you have to use JavaScript. So hone your JavaScript skills and you will become a great React developer.