David Jöch
471 Followers · About Follow

Functional vs Class-Components in React

David Jöch Jul 11, 2018 · 3 min read ★

In this article I want to show you the differences between functional and class components in React and when you should choose which one.

But first let me give you a brief introduction to React components from the documentation:

The simplest way to define a component in React is to write a JavaScript function:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

It's just a function which accepts props and returns a React element.
But you can also use the ES6 class syntax to write components.

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Both versions are equivalent and will give you the exact same output.
Now you might ask yourself: *“When should I use a function and when a class?”*

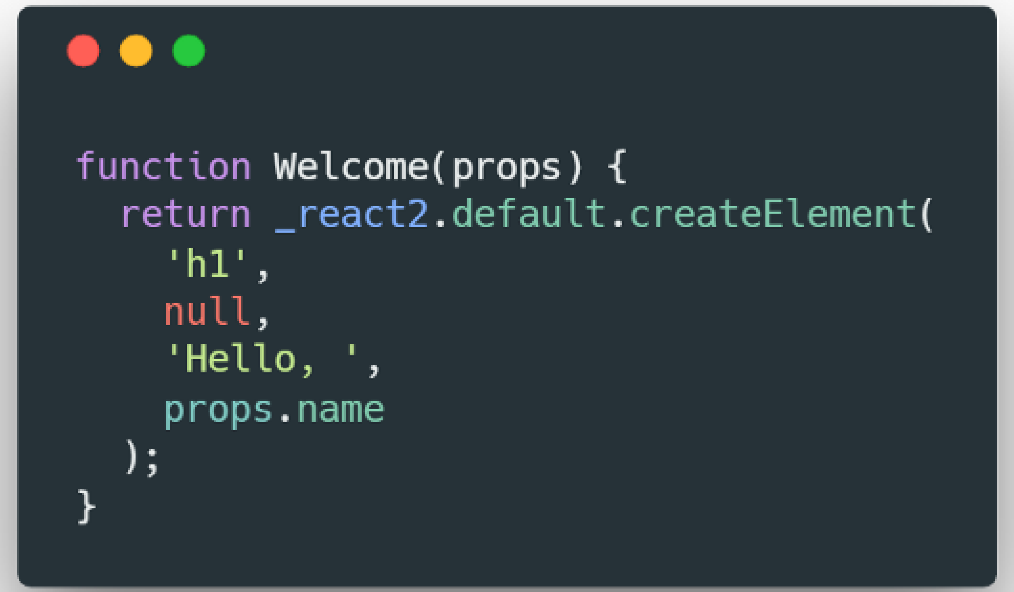
Differences between functional and class-Components

Syntax

The most obvious one difference is the syntax. A functional component is just a plain JavaScript function which accepts props as an argument and returns a React element.

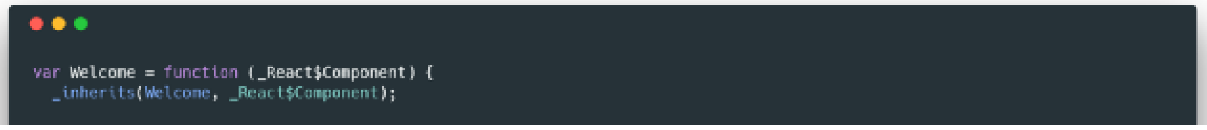
A class component requires you to extend from React.Component and create a render function which returns a React element. This requires more code but will also give you some benefits which you will see later on.

If you take a look at the transpiled code by Babel you will also see some major differences:

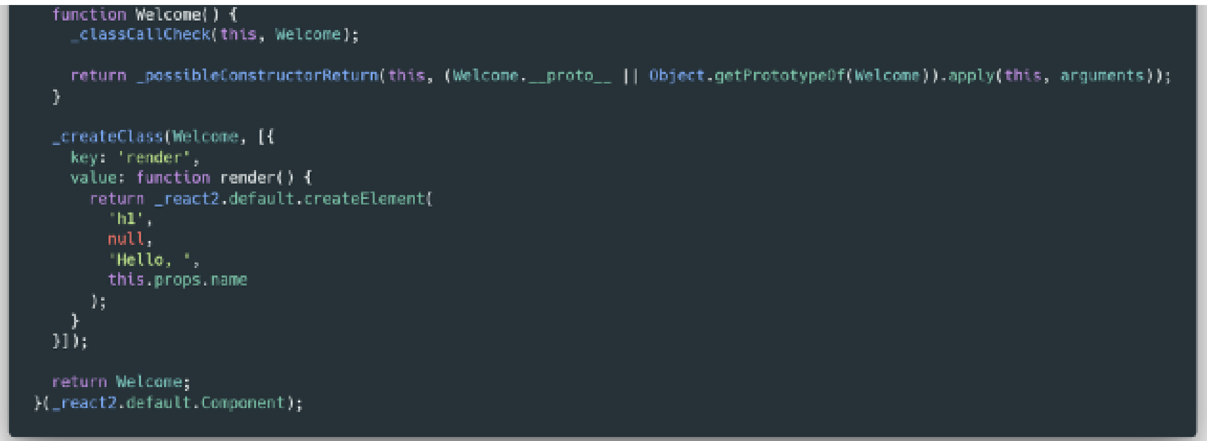


```
function Welcome(props) {
  return _react2.default.createElement(
    'h1',
    null,
    'Hello, ',
    props.name
  );
}
```

As functional component transpiled by Babel



```
var Welcome = function (_React$Component) {
  _inherits(Welcome, _React$Component);
```



```
  function Welcome() {
    _classCallCheck(this, Welcome);

    return _possibleConstructorReturn(this, (Welcome.__proto__ || Object.getPrototypeOf(Welcome)).apply(this, arguments));
  }

  _createClass(Welcome, [{
    key: 'render',
    value: function render() {
      return _react2.default.createElement(
        'h1',
        null,
        'Hello, ',
        this.props.name
      );
    }
  }]);

  return Welcome;
}(_react2.default.Component);
```

The exact same component written as class component transpiled with Babel

State

> *Edit (29.03.2019): This changed with the **React 16.8 Hooks** update!*
> *You can now use the useState hook to use state in your functional components.*

Because a functional component is just a plain JavaScript function, you cannot use setState() in your component. That’s the reason why they also get called functional stateless components. So everytime you see a functional component you can be sure that this particular component doesn’t have its own state.

If you need a state in your component you will either need to create a class component or you lift the state up to the parent component and pass it down the functional component via props.

Lifecycle Hooks

> *Edit (29.03.2019): This changed with the **React 16.8 Hooks** update!*
> *You can now use the useEffect hook to use lifecycle events in your functional components.*

Another feature which you cannot use in functional components are lifecycle hooks. The reason is the same like for state, all lifecycle hooks are coming from the React.Component which you extend from in class components.

So if you need lifecycle hooks you should probably use a class component.

So why should I use functional components at all?

You might ask yourself why you should use functional components at all, if they remove so many nice features. But there are some benefits you get by using functional components in React:

1. Functional component are much **easier to read and test** because they are plain JavaScript functions without state or lifecycle-hooks

2. You end up with **less code**

3. They help you to use **best practices**. It will get easier to separate container and presentational components because you need to think more about your component's state if you don't have access to setState() in your component

4. The React team mentioned that there may be a **performance** boost for functional component in future React versions

Conclusion

And so to answer our question before, you should use functional components if you are writing a presentational component which doesn't have its own state or needs to access a lifecycle hook. Otherwise you can stick to class components or take a look into the library recompose which allows you to write functional components and enhance them with a state or lifecycle hooks with HOCs!

Get in touch

Feel free to connect with me on LinkedIn or Twitter!

Further readings

If you want to learn more about the benefits and costs of functional components I can recommend you the following articles written by Cory House!

React Stateless Functional Components: Nine Wins You Might Have Overlooked

React .14 introduced a simpler way to define components called stateless functional components. These components use…

hackernoon.com

7 Reasons to Outlaw React's Functional Components

Are React's Functional Components Worth The Cost?

medium.freecodecamp.org

React JavaScript Reactjs