Functions as Components in React

In this lesson, we study functional stateless components which are a shorter way to create components that don't require to use lifecycle methods and don't need to maintain state

Functional Stateless Components

React uses the best of different programming paradigms. That's only possible because JavaScript is a many-sided programming language. On the object-oriented programming side, React's class components are a great way of leveraging the abilities of JavaScript classes (inheritance for the React component API, class methods and class properties such as this.state). On the other side, there are lots of concepts from functional programming used in React (and its ecosystem) too. For instance, React's functional stateless components are another way of defining components in React. The question which led us functional stateless components in React: What if components could be used like functions?

```
function (props) {
  return view;
}
```

```
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 a function (functional) which receives an input (e.g. props) and returns the displayed HTML elements (view). It doesn't need to manage any state (stateless) and doesn't need to be aware of any methods (class methods, lifecycle methods). Under the hood, the function only needs to use the rendering mechanism of the render() method from React components. That's

rendering mechanism of the render() method from React components. That's when functional stateless components were introduced.

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

import ReactDOM from 'react-dom';
import {Greeting} from './app.js';

ReactDOM.render(
  <Greeting greeting = "hello from a functional stateless component that takes props!"/>,
  document.getElementById('root')
);
```

Functional stateless components are the preferred way of defining components in React. They have less boilerplate, add less complexity, and are simpler to maintain than React class components. However, as of now, both have their right to exist.

Using arrow functions to simplify functional stateless components

Previously, the course mentioned JavaScript arrow functions and how they improve your React code. Let's apply them to your functional stateless components. The previous Greeting component has two different looks with JavaScript ES5 and ES6:

```
import React from 'react';

// JavaScript ES6 arrow function
export const Greeting = (props) => {
        return(<p>{props.greeting}</p>);
}
```

```
import React from 'react';

// JavaScript ES6 arrow function without body and implicit return
export const Greeting = (props) =>
        <p>{props.greeting}</p>;
```

JavaScript arrow functions are a great way of keeping your functional stateless components in React concise. Even more when there is no computation in between and thus the function body and return statement can be left out.