# Lab: Stateless Functional Components

**Duration: 30 mins**

## Learning objectives

* understand what a stateless react component is and why we might use one
* learn the syntax for writing a stateless component

## Brief

Your task is to read the information below about stateless functional components and refactor the comments application to use stateless functional components wherever possible.

## Stateless Functional Components

If a React component does not need to have its own internal state and is only rendering data passed down to it through props, there is an alternative syntax that we can use for creating it. This type of component is called a stateless component or stateless function.

### Benefits

There are a few reasons why we might use a stateless component. Firstly they usually take fewer lines of code to write so are both quicker to create and easier to read. Win!

Another reason is that because you can’t use state in one, it makes you think more carefully about what you’re putting into your React components and how you should stucture your app. Since the purpose of React is to keep all the state near the top of your component tree and have data flow down, any larger app will have plenty of components that only render data passed as props - most of these can be written as stateless components and doing so prevents you from adding unnecessary state too far down the hierarchy.

Finally, the React developers are aiming to make stateless components more efficient than full ones in the future by eliminating extra checks for lifecycle methods etc, so they’re good to know about for the future if you’re building a big app that needs to be optimized for performance.

### Syntax

Instead of using the extends React.Component syntax to make our component, we can use a plain JavaScript function. This takes in one argument, which is the props object. React will pass our stateless function component the props as an argument by default.

**const** MyComponent **=** (props) **=>** {

}

Then instead of using the render method, whatever we return from the function is rendered as our component. We can wrap this in brackets () to write it over multiple lines, using JSX as usual. Anything we want from props we can just get from the props parameter.

**const** MyComponent **=** (props) **=>** {

**return** (

**<**h1**>**Hello {props.name}**<**/h1>

);

}

This also is the perfect opportunity to use the es6 object destructuring feature.

**const** MyComponent **=** ({name}) **=>** {

**return** (

**<**h1**>**Hello {name}**<**/h1>

);

}

We can also use the arrow functions implicit return to make the syntax more concise.

**const** MyComponent **=** ({name}) **=>** **<**h1**>**Hello {name}**<**/h1>

Any extra functionality we need, like event listeners or functions to compute the props before displaying them, you just include above the return like in any other function.

Note: We will still have to import React from ‘react’ into our stateless functional components because JSX is using React methods to create elements and manipulate the DOM.

## Additional Resources

Documentation:

https://facebook.github.io/react/docs/reusable-components.html#stateless-functions

Good article on the benefits of using stateless components:

https://medium.com/@housecor/react-stateless-functional-components-nine-wins-you-might-have-overlooked-997b0d933dbc#.1ts0vetzf