**Stateful and Stateless Components**

## State Review

In a component, state is data we import — typically to show the user — that is subject to change. It could change because the database we’re getting from may be updated, the user modified it — there are so many reasons that data changes!

## Stateful and Stateless Components

Stateful and stateless components have many different names.

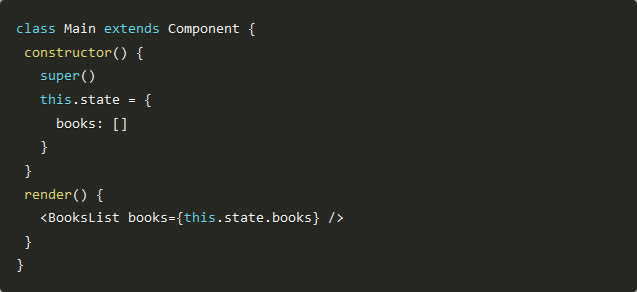
They are also known as:

– Container vs Presentational components

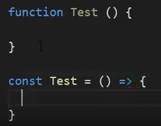
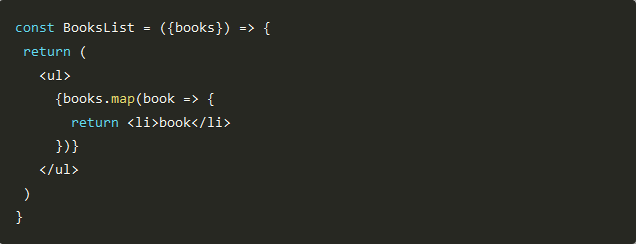
– Smart vs Dumb components

The literal difference is that one has state, and the other doesn’t. That means the stateful components are keeping track of changing data, while stateless components print out what is given to them via props, or they always render the same thing.

Stateful/Container/Smart component:



Stateless/Presentational/Dumb component:



### Functional Components

Functional components are just JavaScript functions. They take in an optional input which, as I've mentioned earlier, is what we call props.

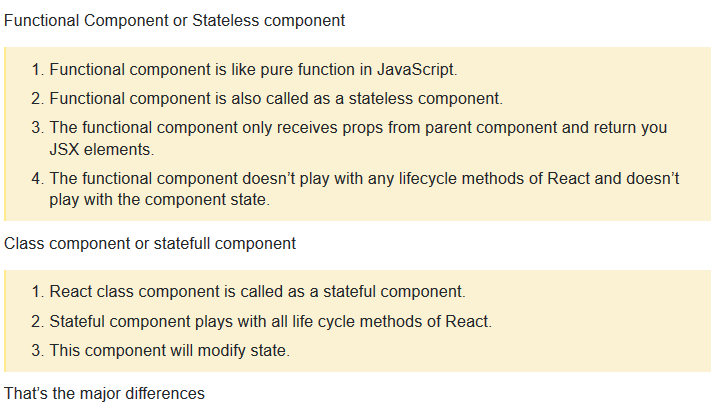
### Class Components

Class components offer more features, and with more features comes more baggage. The primary reason to choose class components over functional components is that they can have state.

### Stateless Components

You can use either a function or a class for creating stateless components. But unless you need to use a lifecycle hook in your components, you should go for stateless functional components. There are a lot of benefits if you decide to use stateless functional components here; they are easy to write, understand, and test, and you can avoid the this keyword altogether. However, as of React v16, there are no performance benefits from using stateless functional components over class components.

The downside is that you can't have lifecycle hooks. The lifecycle method ShouldComponentUpdate() is often used to optimize performance and to manually control what gets rerendered. You can't use that with functional components yet. Refs are also not supported.



## Functional Components

In React, function components are a way to write components that only contain a render method and don't have their own state. They are simply JavaScript functions that may or may not receive data as parameters. We can create a function that takes props(properties) as input and returns what should be rendered. A valid functional component can be shown in the below example.

1. function WelcomeMessage(props) {
2. return <h1>Welcome to the , {props.name}</h1>;
3. }

The functional component is also known as a stateless component because they do not hold or manage state. It can be explained in the below example.

### Example

1. import React, { Component } from 'react';
2. class App extends React.Component {
3. render() {
4. return (
5. <div>
6. <First/>
7. <Second/>
8. </div>
9. );
10. }
11. }
12. class First extends React.Component {
13. render() {
14. return (
15. <div>
16. <h1>TCA</h1>
17. </div>
18. );
19. }
20. }
21. class Second extends React.Component {
22. render() {
23. return (
24. <div>
25. <h2>www.TCA.com</h2>
26. <p>This websites contains the great CS tutorial.</p>
27. </div>
28. );
29. }
30. }
31. export default App;

## Class Components

Class components are more complex than functional components. It requires you to extend from React. Component and create a render function which returns a React element. You can pass data from one class to other class components. You can create a class by defining a class that extends Component and has a render function. Valid class component is shown in the below example.

1. class MyComponent extends React.Component {
2. render() {
3. return (
4. <div>This is main component.</div>
5. );
6. }
7. }

The class component is also known as a stateful component because they can hold or manage local state. It can be explained in the below example.

### Example

In this example, we are creating the list of unordered elements, where we will dynamically insert StudentName for every object from the data array. Here, we are using ES6 arrow syntax (=>) which looks much cleaner than the old JavaScript syntax. It helps us to create our elements with fewer lines of code. It is especially useful when we need to create a list with a lot of items.

1. import React, { Component } from 'react';
2. class App extends React.Component {
3. constructor() {
4. super();
5. this.state = {
6. data:
7. [
8. {
9. "name":"Abhishek"
10. },
11. {
12. "name":"Saharsh"
13. },
14. {
15. "name":"Ajay"
16. }
17. ]
18. }
19. }
20. render() {
21. return (
22. <div>
23. <StudentName/>
24. <ul>
25. {this.state.data.map((item) => <List data = {item} />)}
26. </ul>
27. </div>
28. );
29. }
30. }
31. class StudentName extends React.Component {
32. render() {
33. return (
34. <div>
35. <h1>Student Name Detail</h1>
36. </div>
37. );
38. }
39. }
40. class List extends React.Component {
41. render() {
42. return (
43. <ul>
44. <li>{this.props.data.name}</li>
45. </ul>
46. );
47. }
48. }
49. export default App;