Stateless Components

Introduction

There exist components which do not use state. Such components just print out what is given to them via props, or they just render the same thing. For performance reasons and for clarity of code, it is recommended that such components (those that have only render()) are written as functions rather than classes: a function that takes in props and just renders based on it. It's as if the component's view is a pure function of its props, and it is stateless. The render() function itself can be the component. If a component does not depend on props, it can be written as a simple function whose name is the component name.

Stateless components are those components which don't have any state at all, which means you can't use this.setState inside these components.It has no lifecycle, so it is not possible to use lifecycle methods. When react renders our stateless component, all that it needs to do is just call the stateless component and pass down the props.

Note: A functional component is always a stateless component, but the class component can be stateless or stateful.

When would you use a stateless component??

- When you just need to present the props
- When you don't need a state, or any internal variables
- When creating element that does not need to be interactive
- When you want reusable code

When would you use a stateful component?

- When building element that accepts user input
- Element that is interactive on page
- When dependent on state for rendering, such as, fetching data before rendering
- When dependent on any data that cannot be passed down as props

Ways of creation of stateless components:

- The first is the ES2015 arrow function style with only the return value as an expression. There are no curly braces, and no statements, just a JSX expression

```
…
        const IssueRow = (props) => ( …)
…
```

- second style, a little less concise, is needed when the function is not a single expression. The main difference is the use of curly braces to indicate that there's going to be a return value, rather than the expression within the round braces being an implicit return of that expression's result.

```
…
        function IssueTable(props) {
                …
        }
…
```

Coding Example 1:Simple code illustration of stateless components

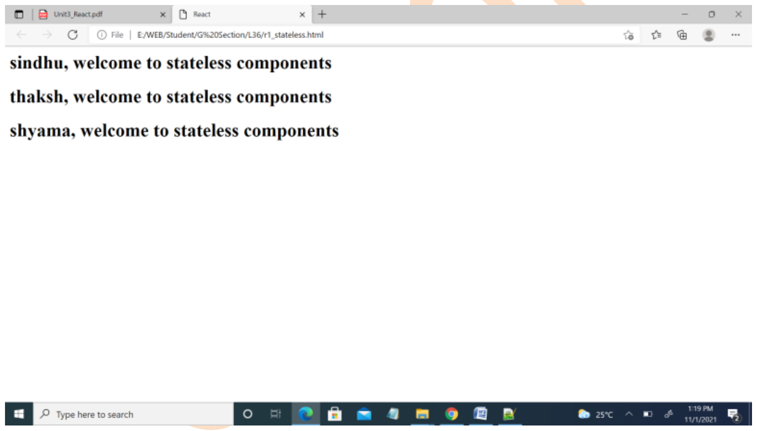
```
<div id = "root"></div>
<script type = "text/babel">
        function Sample(props)
        {       return <h1> welcome to stateless components</h1>                }
        ReactDOM.render(<Sample/>, document.getElementById("root")
</script>
```

Coding Example 2:Usage of props in stateless components

```
<body>
<div id="root"></div>
<script type = "text/babel">
        function Sample(props)
        {
            return <h1> {props.name}, welcome to stateless components</h1>
        }

        ReactDOM.render(<div><Sample name = "sindhu"/>
            <Sample name = "thaksh"/>
            <Sample name = "shyama"/>
            </div>
        ,document.getElementById("root"))
</script>
</body>
```

Output:



sindhu, welcome to stateless components

thaksh, welcome to stateless components

shyama, welcome to stateless components

What if we change the rendering code as below? Is it possible to access the content of tag/componenet inside the stateless componenet creation code? If yes, how?

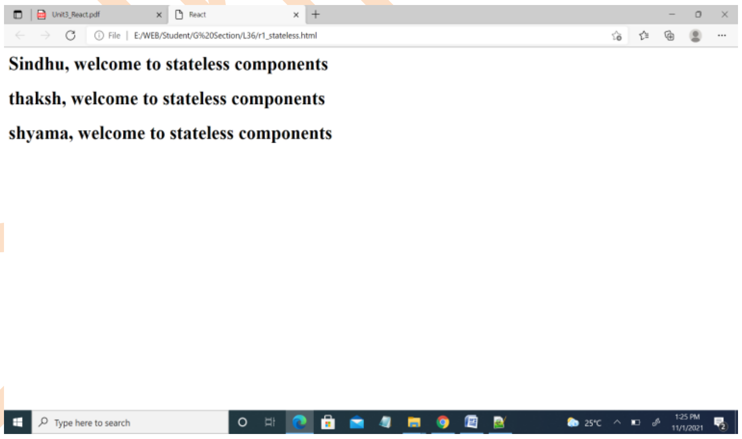
```
ReactDOM.render(<div><Sample>Sindhu</Sample>
                <Sample>thaksh</Sample>
                <Sample>shyama</Sample>
                </div>
            ,document.getElementById("root"))
```

Answer: Possible using props.children

Coding Example 3:Usage of props.children in stateless components

```
<body>
        <div id="root"></div>
        <script type = "text/babel">
                function Sample(props)
                {
                return <h1> {props.children}, welcome to stateless components</h1>
                }
                ReactDOM.render(<div><Sample>Sindhu</Sample>
                        <Sample>thaksh</Sample>
                        <Sample>shyama</Sample>
                        </div>
                ,document.getElementById("root"))
        </script>
</body>
```

Output:



```
Sindhu, welcome to stateless components
thaksh, welcome to stateless components
shyama, welcome to stateless components
```

Nesting of stateless Components

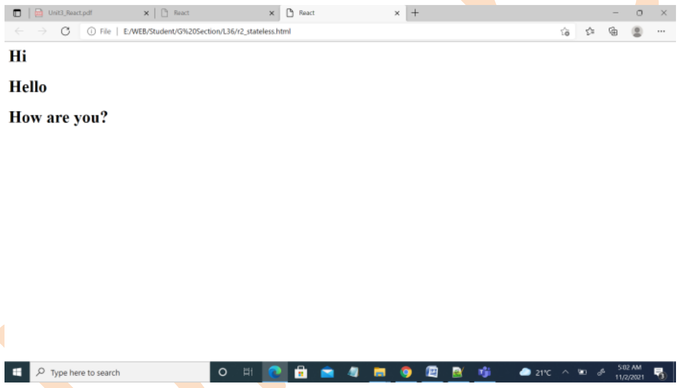
Consider the below code to understand the concept of nesting.

Coding Example 1:

```
        <div id="root"></div>
        <script type = "text/babel">
```

```
function Sample(props){
        return <Sample1>{props.text}</Sample1>  }
function Sample1(props){
        return <h1>{props.children}</h1>   }
ReactDOM.render(<div><Sample     text= "Hi"/>
        <Sample text = "Hello"/>
        <Sample text = "How are you?"/>
        </div>
,document.getElementById("root"))
```

Output:



```
E:/WEB/Student/G%20Section/L36/r2_stateless.html

Hi
Hello
How are you?
```

If we render the component with below code, what changes must be done in the above code? Think!!

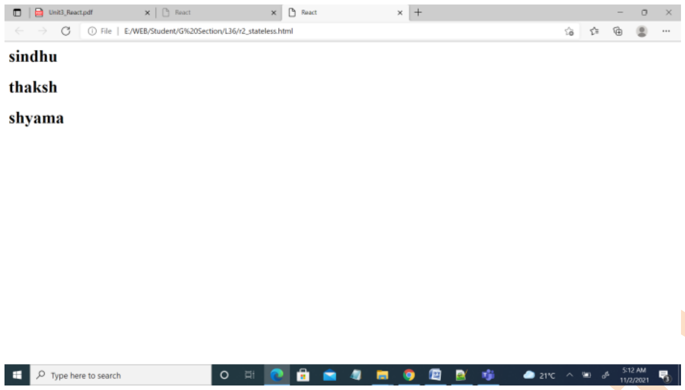
Coding Example 2:

```
ReactDOM.render(<div><Sample>sindhu</Sample>
        <Sample>thaksh</Sample>
        <Sample>shyama</Sample>
        </div>
,document.getElementById("root"))
```

Answer: Only change is as below

```
function Sample(props)
{       return <Sample1>{props.children}</Sample1>      }
```

Output:



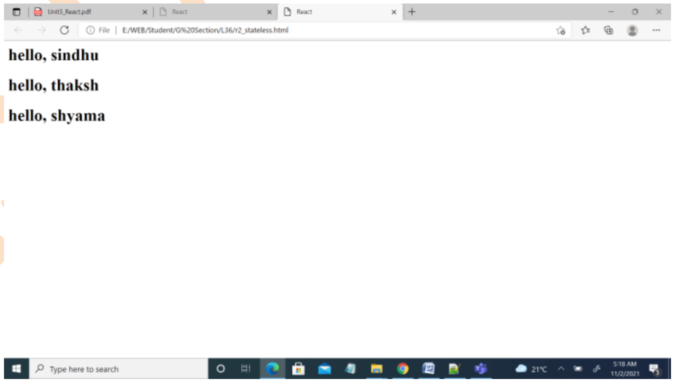
```
sindhu
thaksh
shyama
```

Coding Example 3: Usage of variable within the component and no change in rendering the component

```
function Sample(props)
        {       var greeting = "hello,"
                return <Sample1>{greeting} {props.children}</Sample1>
        }
```

\

Output:



```
hello, sindhu
hello, thaksh
hello, shyama
```

Stateful vs Stateless Components

Stateful Components	**Stateless Components**
Also known as container or smart components.	Also known as presentational or dumb components.
Have a state	Do not have a state
Can render both props and state	Can render only props
Props and state are rendered like *{this.props.name}* and *{this.state.name}* respectively.	Props are displayed like *{props.name}*
A stateful component is always a *class* component.	A Stateless component can be either a functional or class component.