ReactJS - Stateless Component

React component with internal state is called Stateful component and React component without any internal state management is called Stateless component. React recommends to create and use as many stateless component as possible and create stateful component only when it is absolutely necessary. Also, React does not share the state with child component. The data needs to be passed to the child component through child's properties.

An example to pass date to the *FormattedDate* component is as follows −

```
<FormattedDate value={this.state.item.spend_date} />
```

The general idea is not to overcomplicate the application logic and use advanced features only when necessary.

Create a stateful component

Let us create a React application to show the current date and time.

First, create a new react application, *react-clock-app* using *Create React App* or *Rollup* bundler by following instruction in *Creating a React application* chapter.

Next, open the application in your favorite editor.

Next, create *src* folder under the root directory of the application.

Next, create *components* folder under src folder.

Next, create a file, *Clock.js* under *src/components* folder and start editing.

Next, import *React* library.

```
import React from 'react';
```

Next, create *Clock* component.

```
class Clock extends React.Component {
constructor(props) {  super(props);
}
}
```

Next, initialize state with current date and time.

```
constructor(props) {
super(props);
this.state = {
date: new Date()
   }
}
```

Next, add a method, *setTime()* to update the current time −

```
setTime() {
   console.log(this.state.date);
this.setState((state, props) => (
      {
         date: new Date()
      }
   ))
}
```

Next, use JavaScript method, *setInterval* and call *setTime()* method every second to ensure that the component's state is updated every second.

```
constructor(props) {
super(props);
this.state = {
date: new Date()
   }     setInterval( () => this.setTime(),
1000);  }
```

Next, create a *render* function.

```
render() {
}
Next, update the render() method to show the current time.
render() {
return (
      <div><p>The current time is {this.state.date.toString()}</p></div>
   );
}
```

Finally, export the component.

```
Export default Clock;
```

The complete source code of the

Clock component is as follows −

```
import React from 'react';

class Clock extends React.Component
{    constructor(props) {
super(props);        this.state = {
date: new Date()
      }
      setInterval( () => this.setTime(), 1000);
   }
setTime() {
      console.log(this.state.date);
this.setState((state, props) => (
         {
date: new Date()
         }
      ))    }
render() {
return (
<div>
            <p>The current time is {this.state.date.toString()}</p>
         </div>
      );
   }
}
export default Clock;
```

Next, create a file, *index.js* under the src folder and use *Clock* component.

```
import React from 'react'; import
ReactDOM from 'react-dom'; import
Clock from './components/Clock';

ReactDOM.render(
   <React.StrictMode>
      <Clock />
</React.StrictMode>,
document.getElementById('root')
);
```

Finally, create a *public* folder under the root folder and create *index.html* file.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8">
      <title>Clock</title>
   </head>
   <body>
      <div id="root"></div>
      <script type="text/JavaScript" src="./index.js"></script>
   </body>
</html>
```

Next, serve the application using npm command.

```
npm start
```

Next, open the browser and enter *http://localhost:3000* in the address bar and press enter. The application will show the time and update it every second.

```
The current time is Wed Nov 11 2020 10:10:18 GMT+0530(Indian Standard Time)
```

The above application works fine but throws an error in the console.

```
Can't call setState on a component that is not yet mounted.
```

The error message indicates that the setState has to be called only after the component is mounted.

What is mounting?

React component has a life-cycle and *mounting* is one of the stages in the life cycle. Let us learn more about the life-cycle in the upcoming chapters.