15. REACT Higher Order Components

Higher order components are JavaScript functions used for adding additional functionalities to the existing component. These functions are pure, which means they are receiving data and returning values according to that data. If the data changes, higher order functions are re-run with different data input. If we want to update our returning component, we don't have to change the HOC. All we need to do is change the data that our function is using.

15.1. Higher Order Component

Higher Order Component (HOC) is wrapping around "normal" component and provide additional data input. It is actually a function that takes one component and returns another component that wraps the original one.

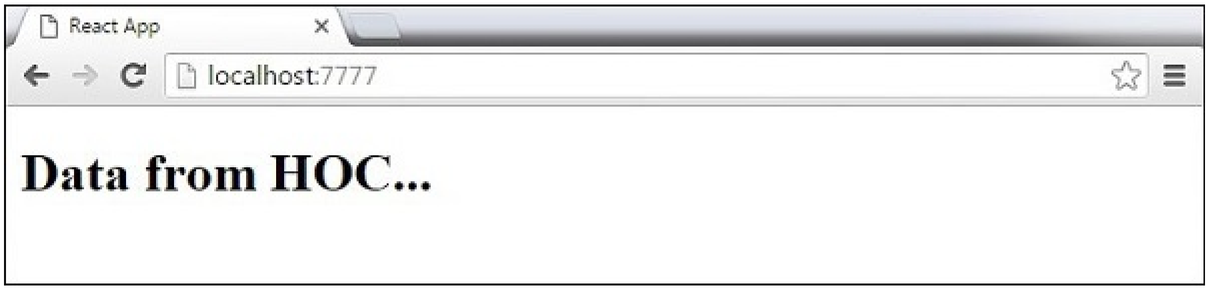
Let us take a look at a simple example to easily understand how this concept works. The **MyHOC** is a higher order function that is used only to pass data to *MyComponent*. This function takes *MyComponent*, enhances it with *newData* and returns the enhanced component that will be rendered on the screen.

Example:

```
import React from 'react';

var newData = {
   data: 'Data from HOC...',
}

var MyHOC = ComposedComponent => class extends React.Component {
   componentDidMount() {
      this.setState({
         data: newData.data
      });
   }
   render() {
      return <ComposedComponent {...this.props} {...this.state} />;
   }
};
class MyComponent extends React.Component {
   render() {
      return (
         <div>
            <h1>{this.props.data}</h1>
         </div>
      )
   }
}

export default MyHOC(MyComponent);
```

If we run the app, we will see that data is passed to **MyComponent**.

Output:



Data from HOC...

Note − Higher order components can be used for different functionalities. These pure functions are the essence of functional programming. Once you are used to it, you will notice how your app is becoming easier to maintain or to upgrade.