React (Features to make Component Reusable)

1. Render props
2. Higher Order Components

**Render Props**

There are two key syntactical things:

1. render prop (used as props)
2. this.props.render

**Pre requisite:** Inside the reusable component (Mouse), within render method you define what data this component is supposed to pass down to child components(Cat), this is done using **{this.props.render(this.state)}.**

**Usage:**

1. Inside component Animals you’re going to use both Mouse and Cat together.

{this.props.render(this.state)}

1. Call Mouse with render prop
2. Add callback and pass props to Cat component

<Mouse render={

**mouse** => (<Cat mouse={**mouse**} />)

}

Example:

class Cat extends React.Component {

render() {

const mouse = this.props.mouse;

return (

<img src="/cat.jpg" style={{ position: 'absolute', left: mouse.x, top: mouse.y }} />

);

}

}

class Mouse extends React.Component {

constructor(props) {

super(props);

this.handleMouseMove = this.handleMouseMove.bind(this);

this.state = { x: 0, y: 0 };

}

handleMouseMove(event) {

this.setState({

x: event.clientX,

y: event.clientY

});

}

render() {

return (

<div style={{ height: '100%' }} onMouseMove={this.handleMouseMove}>

{/\*

Instead of providing a static representation of what <Mouse> renders,

use the `render` prop to dynamically determine what to render.

\*/}

{this.props.render(this.state)}

</div>

);

}

}

class MouseTracker extends React.Component {

render() {

return (

<div>

<h1>Move the mouse around!</h1>

<Mouse render={ **mouse** => ( <Cat mouse={**mouse**}/> ) }

/>

</div>

);

}

}