ReactJS : Intro to Props Lab

# Lab Objectives

Familiarise ourselves with the use of Props to pass data between components

Use Props to pass functions between components, and change data held within one component from within another

Passing data

Passing data between two components is the most simple thing we can do with props. To do this we simply give the prop a name and value when we declare the sub component in the JSX, and use the same name to access ‘this.props.name’ within the sub component, this is shown in the example below:

**App.js**

class App extends Component {

number = 2;

render() {

return (

<div className="App">

<p>This is the app Component</p>

<SubComponent sentNumber={this.number}></SubComponent>

</div>

);

}

}

**SubComponent.js**

export class SubComponent extends Component {

render() {

return (

<div>

<p>This code is within SubComponent</p>

<p>The number passed to SubComponent was: {this.props.sentNumber}</p>

</div>

)

}

}

**Task:** Implement an example of passing a prop from one component to another and have this passed value displayed on the screen, then extend this example to assemble a sentence from multiple passed props at once

1. **Passing Fuctions**

We can pass functions through props in a similar manner to values, we define the function in the parent component, then when adding the subcomponent in the JSX define a prop name in the same way, however we make the value a fat arrow function containing a call to the function.

An example of this is shown below:

**App.js**

class App extends Component {

render() {

return (

<div className="App">

<p>This is the app Component</p>

<SubComponent onRender={() => this.whenSubComponentRender()}></SubComponent>

</div>

);

}

whenSubComponentRender() {

console.log('sub component rendered');

}

}

**SubComponent.js**

export class SubComponent extends Component {

render() {

this.props.onRender();

return (

<div>

<p>This code is within SubComponent</p>

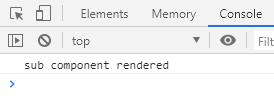
</div>

)

}

}

If you then access the console you will see:



# Task

To tell the UserDetail component what to display the User component needs to give it data; we can accomplish this using **props**. In this case the user data will be hard-coded on the User component, but this pattern is the same if we were fetching it from an external source.

Firstly, create a user data object in the User component. This should include a variety of different types and values. Next, we need send this data from the User component to the UserDetail component:

<UserDetail userData={myUserData} />

Next, we need set up the UserDetail component to use the userData prop. **this.props** contains anything we pass to the component via props, so in this case the data we want is in **this.props.userData.**

Using the JSX syntax, populate the UserDetail component with the data you’ve passed it. Try to make it as readable as possible. For example:  
  
