**State –** like a save point where all data of the program at the specific time is saved in. (Note: if the program/script is not expected to change, non-dynamic, then we shouldn’t be using setState in React. Props should be used instead.)

**Data flow** in React is one way, from parent to child.

**Create Component**

//create a component named MessageComponent

Var MessageComponent = React.createClass({ // create a class in react

Render: function(){ //what we want the this component to do

Return (

<div>{this.props.message}</div> // we want it to return the message attached

);

}

});

// Render an instance of MessageComponent into document.body (note: this part is always needed to activate the component we want

ReactDOM.render(

< MessageComponent message =”hello!” />, //we want to display the message hello!

document.body // we want to display the component in the <body>

);

**Interactivity and dynamic UIs**

//create a component named MessageComponent

Var likeButton = React.createClass({ // create a class in react

getInitialState: function(){ //state is set inside the component

Return {liked: false}; //set initial state to be not liked

},

handleClick: function(event){

this.setState({liked: !this.state.liked}); //change the sate to be the opposite of what it was

},

Render: function(){

Var text = this.state.liked ? ‘like’ : ‘havent\’t liked’; //test if the state is liked or not

Return {

<p onClick={this.handleClick}>

You {text} this. Click to toggle, // toggle the handleClick when cliked on

</p>

);

}

});

ReactDOM.render(

< LikeButton />,

document.getElementbyId(‘example’) // we want to display the component in the id “example”

);

Note: with react, you can use ref=”name” and use this.refs.name.value to get the reference input.