1.下面是父子组件，也就是组件的嵌套使用。思路是在父组件的render方法中放入子组件。



2.

从父组件传递数据到子组件。思路是通过在父组件中传入数据，成为子组件的props作为数据传递的方式。



3.

子组件传递数据到父组件。子组件传递数据给父组件的过程需要通过调用父组件传递给子组件的方法中的参数进行传递。例如下面例子，我们先在父组件定义函数，从父组件传递方法给子组件，作为子组件的props.然后子组件通过调用这个方法，然后将需要传递的参数，作为该方法的参数进行从子组件传递数据到父组件。

下面例子中，子组件以自己的state.msg作为需要传递的参数，调用父组件的方法，将需要传递的数据作为该方法的参数进行调用。然后在父组件定义的该方法中，通过this(父组件),来设置父组件中的state数据域。



4.父组件传递数据给子组件的联动变化问题。

<script type="text/babel">  
 **class** Node2 **extends** React.Component{  
 render(){  
 **return** (  
 <div >Node2 Component => {**this**.props.setMsg}</div>  
 )  
 }  
 }  
 **class** Node **extends** React.Component{  
 **constructor**(){  
 **super**();  
 **this**.state = {  
 msg : 'Node1 Component Message'  
 }  
 }  
 show(){  
 **this**.setState({  
 'msg' : "New Node1 Message"  
 });  
 }  
 render(){  
 **return** (  
 <div>  
 <div onClick={**this**.show.bind(**this**)}>Node Component => {**this**.state.msg}</div>  
 <Node2 setMsg={**this**.state.msg}/>  
 </div>  
 )  
 }  
 }  
 ReactDOM.render(<Node/>, app);  
</script>

通过上面例子我们可以实现组件的嵌套，就是在一个组件的render方法里面加上另一个组件。并且我们还可以通过x=y的方式向另一个组件传递数据，并作为其property.

我们需要注意的是子组件现在显示的信息的它的props.setMsg域。 一般来说props是不可变，但是在这里我们发现，如果我们点击Node组件，那么Node组件的state将会变化，Node组件信息改变并且重新渲染，并且Node1组件也会重新渲染。原因是我们直接将Node的状态作为Node1的props，没有在Node1中设置额外的Node1里面的state作为中间状态。所以Node的msg状态和Node1的setMsg property是同一数据线上的，当Node的msg状态改变Node1也会被重新渲染。

如果我们通过如下方式书写：

<script type="text/babel">  
 **class** Node2 **extends** React.Component{  
 **constructor**(){  
 **super**()  
 **this**.state={  
 'msg' : ''  
 }  
 }  
 componentWillMount(){  
 **this**.setState({  
 'msg' : **this**.props.setMsg  
 })  
 }  
 render(){  
 **return** (  
 <div >Node2 Component => {**this**.state.msg}</div>  
 )  
 }  
 }  
 **class** Node **extends** React.Component{  
 **constructor**(){  
 **super**();  
 **this**.state = {  
 msg : 'Node1 Component Message'  
 }  
 }  
 show(){  
 **this**.setState({  
 'msg' : "New Node1 Message"  
 });  
 }  
 render(){  
 **return** (  
 <div>  
 <div onClick={**this**.show.bind(**this**)}>Node Component => {**this**.state.msg}</div>  
 <Node2 setMsg={**this**.state.msg}/>  
 </div>  
 )  
 }  
 }  
 ReactDOM.render(<Node/>, app);  
</script>

这样当我们点击Node的时候，只有Node的信息会改变。而Node1的信息不会改变，因为我们现在在Node1组件上显示的是Node1组件他自己的一个state, 而初始化是通过componentWillMount的props.msg来复制。当Node的信息改变，Node1的props改变，但是此时不会再改变Node1的状态，所以Node1的信息不会改变。