react学习笔记：v0.14.1
http://www.ruanyifeng.com/blog/2015/03/react.html
http://wiki.jikexueyuan.com/project/react/displaying-data.html

1. {...}
    i. 表示渲染为动态值
2. 子节点
    a. 特殊的属性this.props.children
        i.
```
// 主节点
var Father = React.createClass({
    render: function(){
        return (
            <div className="father">
                <h2>这个是父节点</h2>
                {this.props.children}
            </div>
        );
    }
});
// 置入文档
ReactDOM.render(
    <Father>
        <p>123</p>
    </Father>,
    document.getElementById('search')
);
```
3. 属性
    a. 特殊的属性
        i. key
            a.
```
{limit.map(this.props.option, function(val){
    return <option key={val.value} value={val.value}>{val.key}</option>;
})}
```
        ii. ref
            a.
```
var Father = React.createClass({
    render: function(){
        return (
            <div>
                <div ref="title">标题</div>
            </div>
        );
    },
    componentDidMount: function(){
        console.log(this.refs.title)
    }
});
```
        iii. dangerouslySetInnerHTML
    b. 特殊属性
        i. class => className
        ii. for => htmlFor <label>
    c. 内置的样式
        i.
```
// 主节点
var Father = React.createClass({
    render: function(){
        return (
            <div style={ {color: "#F00"} }>我是内容</div>
        );
    }
});
```
4. 事件

5. JSX注释
    a. 安全的注释
        i.
```
{/* 我是注释 */}
```
6. 生命周期
    a. 实例化
        1. getDefaultProps[获取默认的属性]
        2. getInitialState[获取初始的状态]
        3. componentWillMount[组件即将安装]
        4. render[渲染]
        5. componentDidMount[组件完成安装]
            i. ReactDOM.findDOMNode(this)[获取实例]
                a.
```
componentDidMount: function(){
    ReactDOM.findDOMNode(this);
},
```
    b. 存在期
        1. componentWillReceiveProps[组件即将获取属性，由于props在新版中无法被更改，也就是setPrps方法被遗弃]
            i.
```
me.setProps({content: "bbbb"});
```
设置这个会触发该方法但是会警告

        2. shouldComponentUpdate[是否将组件进行更新，需要返回一个布尔类型]
        3. componentWillUpdate[组件将完成更新]
        4. componentDidUpdate[组件完成更新]
    c. 销毁期
        1. componentWillUnmount

i.
```
// 子节点
var Child = React.createClass({
    render: function(){
        return (
            <div>
                {this.props.content}
            </div>
        );
    },
    componentWillUnmount: function(){
        console.log('Child componentWillUnmount', arguments);
    }
});
// 父节点
var Father = React.createClass({
    getInitialState: function(){
        return {
            show: true
        };
    },
    render: function(){
        return (
            <div>
                {this.state.show && <Child content="我是内容" />}
            </div>
        );
    },
    componentDidMount: function(){
        var me = this;
        setTimeout(function(){
            me.setState({show: false});
        }, 1000);
    }
});
// 置入文档
ReactDOM.render(
    <Father />,
    document.getElementById('search')
);
```

7. pros state
    i. propTypes [类型匹配]

a.

```
getDefaultProps: function(){
    return {
        testString: "a",
        testBoolean: true,
        testFunction: function(){},
        testNumber: 1,
        testArray: [],
        testArrayOf: ["a1", "a2"],
        testObject: {},
        testObjectOf: {a: "a1"},
        testOneOf: "a1",
        testOneOfType: "a1",
        testAny: 1,
        testElement: <Seed />,
        testNode: [<Seed />], // 1 "a" undefined null false <Seed /> [...]
        testInstanceOf: new String("a"),
        testShape: {
            string: "a1",
            number: 123
        },
        other: "a2" //
    };
},
propTypes: {
    testString: React.PropTypes.string,
    testBoolean: React.PropTypes.bool,
    testFunction: React.PropTypes.func,
    testNumber: React.PropTypes.number,
    testArray: React.PropTypes.array,
    testArrayOf: React.PropTypes.arrayOf(React.PropTypes.string),
    testObject: React.PropTypes.object,
    testObjectOf: React.PropTypes.objectOf(React.PropTypes.string),
    testOneOf: React.PropTypes.oneOf(["a1", "a2"]),
    testOneOfType: React.PropTypes.oneOfType([React.PropTypes.string, React.PropTypes.bool]),
    testAny: React.PropTypes.any,
    testElement: React.PropTypes.element,
    testNode: React.PropTypes.node,
    testInstanceOf: React.PropTypes.instanceOf(String),
    testShape: React.PropTypes.shape({
        string: React.PropTypes.string,
        number: React.PropTypes.number
    }),
    other: function(props, propName, componentName){
        if(props[propName] !== "a1"){
            return new Error("error")
        }
    }
},
```

ii. props 在creatClass的时候就被调用了

1. 传递的方式
   i. `<Child content={"我是内容"} />,`
   ii. `<Child content="我是内容" />,`
   iii. `<Child {...{content:"我是内容"}} />,`
   iv.
   ```
   getDefaultProps: function(){
       return { content: "属性" };
   },
   ```
2. 可以传递，但是是单向的
3. 属性是只读的不可修改引用[对于引用对象还是能修改其值的]
4. 用作数据源

iii. state

1. 传递的方式
   i.
   ```
   getInitialState: function(){
       return { name: "状态" };
   },
   ```
2. 只是内部属性外部组件是访问不了的
3. 属性不是只读的，可以修改引用

i.
```
var Child = React.createClass({
    getInitialState: function(){
        return {
            show: true
        };
    },
    render: function(){
        var some;
        if(this.state.show){
            some = <span>要显示的内容</span>;
        };
        return (
            <div onClick={this.clickHandler}>
                点我 {some}
            </div>
        );
    },
    clickHandler: function(){
        this.setState({show: !this.state.show});
    }
});
```

4. 用作状态机
5. setState[变更某几项] replaceState[替换整个状态]

iv. 避免反模式

1. 避免在初始化state的时候进行运算
2. 尽量保持state的纯粹
   i. 不要
```
getInitialState: function(){
    return {
        content: forMate(this.props.initContent)
    };
},
```
   ii. 而是
```
getInitialState: function(){
    return {
        content: this.props.initContent
    };
},
render: function(){
    return (
        <div>
            我是Father <Child content={forMate(this.state.content)} />
        </div>
    );
},
```

v. 强制刷新试图
   a. `me.forceUpdate();`

8. 事件
   i. 使用方式
      a. `onClick={this.clickHandler}`
   ii. *触控事件启用 React.initializeTouchEvents(true);
9. 父、子组件
   i. 通信 父 => 子
      a. 通过props的单项传递
   ii. 通信 子 => 父
      a. 通过props的回调方法(钩子)去传递
   iii. 两个无关的通信
      a. 通过全局的广播事件
10. 混入 mixins
   i. 混入的是个对象
   ii. 对私有component方法(除了render)都能叠加定义，并且保持顺序执行

a.
```
var Seed = {
    getDefaultProps: function(){
        return {
            content: "我是内容"
        };
    }
};
var Father = React.createClass({
    mixins: [Seed],
    getDefaultProps: function(){
        return {
            title: "我是标题"
        };
    }
    render: function(){
        return (
            <span>
                我是Father {this.props.content} {this.props.title}
            </span>
        );
    }
});
```

11. 表单
    i. 约束性表单
12. 兼容IE8
    i. 要引入es5-shim v4.1.7
    ii. 再引入es5-sham v4.1.7
13. Flux 事件系统
14.