🡪

React中一般我们将组件标签写为<Component />. 但是这并不是必须的，实际规则是如果标签不需要包裹其他内容便可以写成单闭合标签,就和<input />一样道理。如果需要包含其他内容，则所有组件都可以写为<Component></Component>.

🡪

props.children将指代包含组件开标签以及闭合标签之间的所有内容。主要的意义在于设计包裹组件，或者该包裹组件作为工具包让其他人使用时，我们不知道未来此包裹组件将具体包含什么内容，所以引入props.children进行统一的指代。如果只是自己设计组件，能够明确组件将包含具体哪些组件，则props.children意义不大。

🡪

React提供了几个自带的功能性函数用于处理props.children

|  |  |
| --- | --- |
| React.Children.map(children, function[(thisArg)] | React.Children.map函数需要2个参数。第一个参数是指定children,一般为this.props.children.第二个参数是一个函数，该函数接受一个参数为children中的每一个元素，React将对children中每一个元素调用该函数并且将结果放入数组中进行返回。 |
| React.cloneElement | 用于复制一个React元素，第一个参数是原React元素，第二个参数是新加入的props对象。返回的结果为一个克隆出来的新的React元素，并且key和ref都保留。  我们可以连用React.Children.map函数以及React.cloneElement函数，为组件的Children批量的新增props。[这样的原因是我们不能直接修改props.children] |

🡪

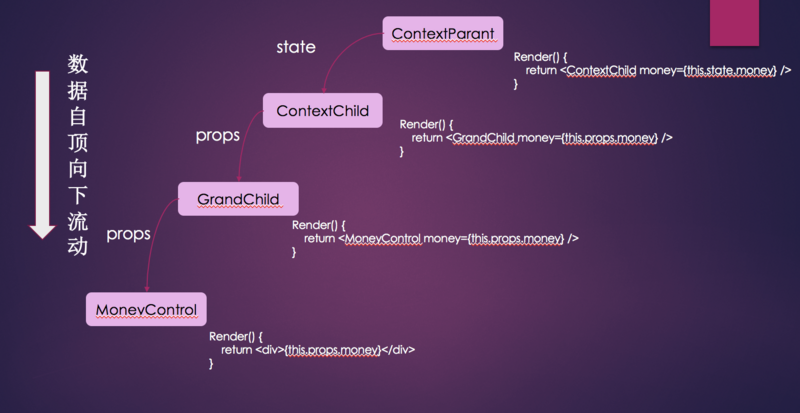
***React的Context概念：***

首先React中父子组件的通信机制为：

1、父组件向子组件传递数据(自身的state)时，是在调用子组件的时候通过参数传递给子组件，子组件通过this.props进行接收；  
2、子组件如果更改父组件的一些属性，则是通过父组件定义的回调方法方法来传递给子组件，子组件调用该方法更改父组件的state；  
3、如果父组件想要更改子组件的一些状态时，通过ref进行标记，可以获取子组件的所有信息，从而调用子组件的方法和值；

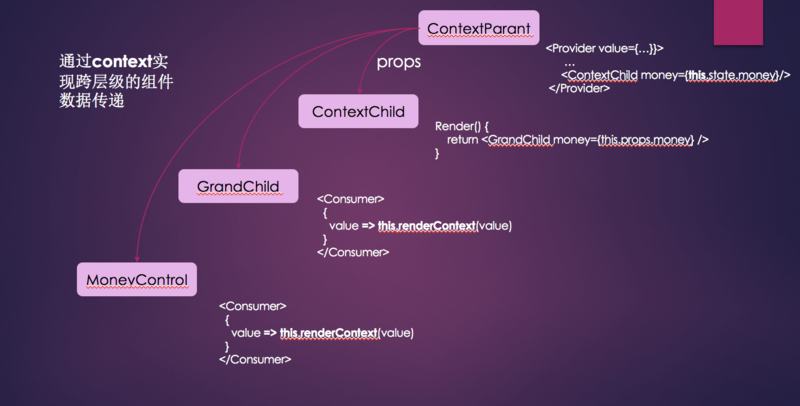
但是如果组件的层级很多，则props需要逐层进行传递，十分麻烦。***基于此React 16.3开始，推出了React自身的新版Context机制 （16.3以前React也有Context机制但是React不推荐使用）***，Context 机制通过组件树提供了一个传递数据的方法，从而避免了在每一个层级手动的传递 props 属性。实现跨层级进行数据传递。

**例如基本组件单向数据流动**：



在父组件的状态this.state.money通过props逐层传递下去，最终给MoneyControl控件使用。

**采用context实现跨层传递数据：**



在最外层的组件上，通过生产者Provider组件进行包裹，并存储共享数据到value中，当然可以是任何数据类型。需要用到共享数据的组件均可通过Consumer进行数据获取。

***P.s每当Provider的值发生改变时, 作为Provider后代的所有Consumers都会重新渲染***

***React Context相关用法：***

React context用法主要有：

1. React.createContext(defaultValue)该方法返回一个Context对象。

例如：

const MyContext = React.createContext(default); // 可以<MyContext.Provider>获取Provider，可以<MyContext.Consumer>获取Consumer

const {Provider, Consumer} = React.createContext(defaultValue); // 直接赋值给Provider以及Consumer

2. 使用Provider

Provider代表生产共享数据的地方，其中value定义放置的共享数据

例如：

<Provider value={/\*共享的数据\*/}>

/\*里面可以渲染对应的内容\*/

</Provider>

3. 使用Consumer

Consumer可以理解为消费者。 他是专门消费供应商(Provider 上面提到的)产生数据。Consumer需要嵌套在生产者下面。

例如：

<Consumer>

{value => /\*根据上下文 进行渲染相应内容\*/}

</Consumer>

4．指定组件的contextType属性

例如：组件名.contextType = MyContext, 其中MyContext为React.createContext()所生成的Context. 从而该组件会使用离他最近的该类型的Context.

5.