🡪

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，则其为包裹组件可以写为<Component></Component>.

🡪

包裹组件中将使用props.children， 它将指代包裹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。

一般在包裹组件的render方法中将使用{this.props.children}。然后生成组件的时候所有包含在包裹组件开标签以及闭合标签之间的内容，都将放置在包裹组件render方法中的{this.props.children}位置。

🡪

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；

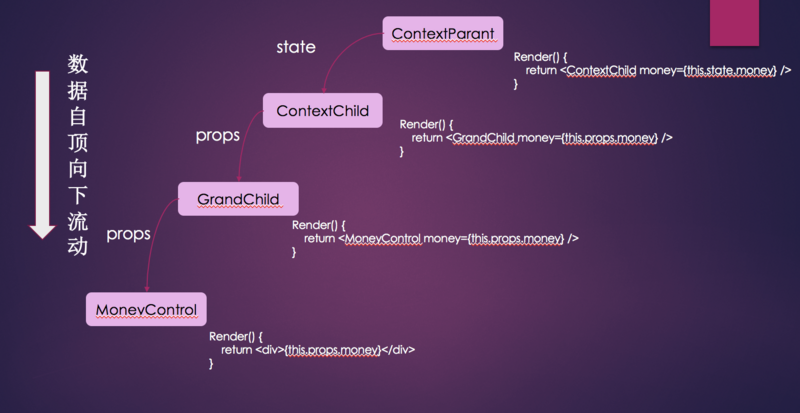
P.s但是如果组件的层级很多，则props需要逐层进行传递，十分麻烦。***基于此React 16.3开始，推出了React自身的新版Context机制 （16.3以前React也有Context机制但是React不推荐使用）***，Context 机制通过组件树提供了一个传递数据的方法，从而避免了在每一个层级手动的传递 props 属性。实现跨层级进行数据传递。

P.s 使用React提供的Context机制的时候，组件自身的state和props也可以照常使用。所以我们可以混合使用。

P.s Context所传递的数据可以是普通数据也可以是父组件想传递给子组件的回调方法。

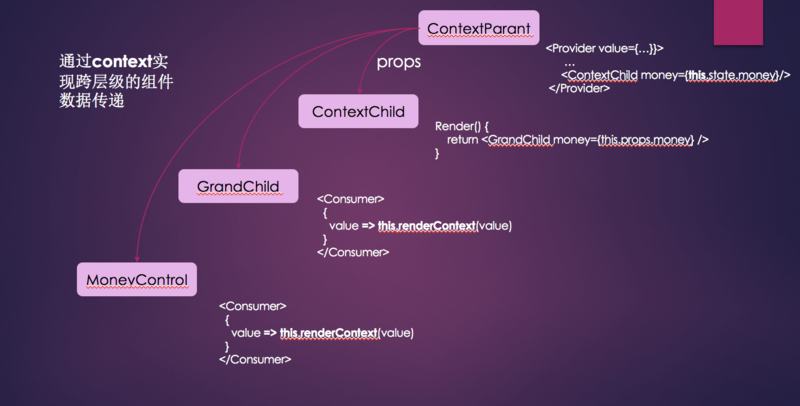
P.s React 16.3提供的Context机制一般用于小型程序。如果是大型程序，一般使用Redux或者React-Redux来处理。

**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：

🡪

React.createContext(defaultValue)该方法返回一个Context对象。defaultValue为默认数据，该数据可以在定义Provider的时候被重写。

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

🡪

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对象

例如：

<Provider value={/\*共享的数据\*/}>

/\*里面可以渲染对应的内容\*/

</Provider>

🡪

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需要在结构上嵌套在生产者当中。[Consumer嵌套在Provider中很多时候并非直接嵌套，而是Provider包裹A组件，然后A组件的子组件是B组件，B组件使用Consumer. 这样整体扩展开以后，Consumer也是包裹在Provider当中]

Value将为context对象，使用context.数据名 获取其中的某一个域的数据。

例如：

<Consumer>

{value => /\*根据上下文 进行渲染相应内容\*/}

</Consumer>

🡪

指定组件的contextType静态属性

例如：在组件中使用static contextType = MyContext, 则该组件context为React.createContext()所生成的MyContext. 从而该组件可以使用this.context.xxx来获取数据。

🡪