开发过游戏的都应该很清楚，“刷屏”是多么的重要。其实开发应用也如此，当组件的数据被修改后，如何及时更新组件呈现出最新的数据与效果一样至关重要。

**那么这里Himi大概讲三种常用的方式：**

1. this.setState()  【最为常用】

这是在事件处理函数中和请求回调函数中触发 UI 更新的主要方法。

一般情况下setState() 总是触发一次重绘，除非在 shouldComponentUpdate() 中实现了条件渲染逻辑。如果使用可变的对象，但是又不能在 shouldComponentUpdate() 中实现这种逻辑，仅在新 state 和之前的 state 存在差异的时候调用 setState() 可以避免不必要的重新渲染。

举例、代码段如下：

1. constructor(props) {

4. super(props);

7. this.state = {

10. myName:'I am MyName!',

13. };


17. }



22. testFun(){

25. this.setState({myName:'组件被刷新了'});
27. }





34. render() {

37. ......

40. <TouchableHighlight

43. underlayColor='#4169e1'

46. onPress={this.testFun.bind(this)}

49. >

52. <Image
53. source={require('./res/himi.png')}
54. style={{width: 70, height: 70}}
55. />

58. </TouchableHighlight>

61. ......

64. )}

1. 在this.state中添加了一个 字符串变量 myName 用于Text 组件的文字显示

2. 自定义了一个 testFun 函数，用于 Touchable的回调处理，其中调用了this.setState 函数并修改了myName的数据

3. 在render中添加了一个高亮触摸组件，用于演示效果。

【注】假如有 a、b、c三个组件，b 是 a 的子组件，c是 b 的子组件，那么 b 中 setState 之后，b 和 c 会 rerender，而 a 不会。 因此建议大家可以把每个组件提供一个重绘接口，然后提供给其他组件调用。

效果如下：（点击查看动态效果）

2.  this.forceUpdate() 【较为常用，但是不推荐】

如果 render() 方法从 this.props 或者 this.state 之外的地方读取数据，你需要通过调用 forceUpdate() 告诉 React 什么时候需要再次运行 render()。如果直接改变了 this.state，也需要调用 forceUpdate()。

调用 forceUpdate() 将会导致 render() 方法在相应的组件上被调用，并且子级组件也会调用自己的 render()，但是如果标记改变了，那么 React 仅会更新 DOM。通常情况下，应该尽量避免所有使用 forceUpdate() 的情况，在 render() 中仅从 this.props 和 this.state 中读取数据。这会使应用大大简化，并且更加高效。

举例、在上面的示例代码基础上修改，如下：

1. testFun(){

4. this.state.myName='组件被刷新了';

7. }

10. testForceFun(){

13. this.forceUpdate();

16. }



21. render() {

24. ......

27. <TouchableHighlight

30. underlayColor='#4169e1'

33. onPress={this.testFun.bind(this)}

36. >

39. <Image
40. source={require('./res/himi.png')}
41. style={{width: 70, height: 70}}
42. />

45. </TouchableHighlight>



50. <TouchableHighlight

53. underlayColor='#4169e1'

56. onPress={this.testForceFun.bind(this)}

59. >

62. <Image
63. source={require('./res/himi.png')}
64. style={{width: 40, height: 40}}
65. />

68. </TouchableHighlight>

71. ......
73. ｝

改动说明：

a)  修改 testFun 函数，让其作用只是修改了 myName 的值，并没有setState！所以不重绘

b)  添加 testForceFun 函数，作用是调用强制重绘函数。

c) render中多加了一个按钮，来触发 testForceFun函数。

运行效果图如下：（注意  testForceFun 函数对应的是图片较小的那个哦～ ）

通过示意图可以知道，Himi 调用了 testFun、接着调用testForceFun ，才能看到刷新。

3. this.setProps()   【不常用】

同一个节点上再次调用 React.render() 来更新根组件是首选的方式，也可以调用 setProps() 来改变组件的属性，触发一次重新渲染。

但是！此方法仅在根组件上面调用。也就是说，仅在直接传给 React.render() 的组件上可用，在它的子级组件上不可用。如果你倾向于在子组件上使用 setProps()，不要利用响应式更新，而是当子组件在 render() 中创建的时候传入新的 prop 到子组件中。

说的通俗一点：父组件不能直接修改 prop属性，只能父类在使用传入的时候进行设置。