## React读书笔记-组件的生命周期

在我们的React中all is component，我们通过调用React的createClass方法（）或者ES6的class来继承React.Component对象来创建组件，并需要包含一个render方法，同时可以在render的同级创建描述该组件生命周期的方法。

接下来，我们先来看一下React定义的组件生命周期有哪些：

getDefaultProps(es5的语法)：在这个方法里面我们可以设置默认的父级传递给组件的属性，只在组件创建的时候调用一次。

getInitialState(es5的语法):在这个方法内我们可以预设组件的初始状态，此中return的对象就是该组件的默认状态，如果没有此步，我们就很难在后续的声明周期对组件进行操作。

componentWillMount()：组件将被装载，此方法在组件的声明周期中也只调用一次。我们一般可以在此不做state或者props的验证判断。此时组件还没有被插入DOM，进行DOM操作或者设置css就会报错。

render()：组件进行渲染。

componentDidMount()：组件装载完成，在此步组件就被插入到真实DOM中了，可以进行DOM操作，可以this.getDOMNode()来获取相应的DOM节点。我们一般在此生命周期进行ajax的异步请求数据来改变state来重新渲染组件。值得注意的是当执行异步请求的响应时，在更新 state 前，一定要先通过 this.isMounted() 来检测组件的状态是否还是 mounted。或者是调用setTimeout或者是setInterval中包含函数方法，这是为了防止阻塞。

**已上生命周期可以看作是组件在我们的javascript中怀胎十月的孕育直至呱呱落地的过程，此后所有的变化将不会重新调用这些步骤。**

componentWillReceiveProps(nextProps):在组件接收到父级新的Props时候的回调，在此，我们可以获得一次在重更组件之前的修改本组件this.state的机会。老的props可以通过this.props获取到，这个时候调用this.setState()将不会引起第二次渲染，因为此时还没有调用render()。

shouldComponentUpdate(nextProps,nextState):在接收到新的props或者state，将要渲染之前的调用，在此我们经常可以通过返回判断后的布尔值来控制组件是否更新渲染。如果 shouldComponentUpdate 返回 false，则 render() 将不会执行，直到下一次 state 改变。（另外，componentWillUpdate 和 componentDidUpdate 也不会被调用。）**需要注意的是** 该方法在初始化时候不会调用，并且在我们使用this.forceUpdate()强制更新时也不会调用。我们如果非常肯定新的props和state在某种情况的应该不更新组件，则该方法里应该return false。

componentWillUpdate(nextProps,nextState):可以在此方法中做一些准备工作，但是需要注意的是不能够在此this.setState(),这一步应该放到componentWillReceiveProps里面，就不会引起两次渲染。

componentDidUpdate(prevProps,prevState):组件的更新已经重新渲染到DOM中，在此方法中可以操作DOM元素和this.getDOMNode()。

componentWillUnmount():当组件从DOM中移除的时候被立即调用。一般我们可以在此方法中，清除掉组件调用的一些无效的定时器和临时创建的DOM元素等。

到此步，一个组件的生命周期完成。

componentWillMount(){  
 console.log("1componentWillMount")  
 }  
  
 componentDidMount(){  
 //一般ajax装载数据放在这一步  
 console.log("2componentDidMount")  
 }  
  
 shouldComponentUpdate(){  
 console.log("3shouldComponentUpdate");  
 //需要返回一个布尔值  
 **return true**;  
 }  
 componentWillUpdate(){  
 console.log("4componentWillUpdate")  
 }  
  
 componentDidUpdate(){  
 console.log("5componentDidUpdate")  
 }  
  
 componentWillUnmount(){  
 //组件的未响应ajax中断一般在此调用  
 //this.serverRequest.abort();  
 console.log("6componentWillUnmount");  
 }