# React的组件

1. 组件封装的基本思路就是面向对象思想。交互基本以DOM为主，逻辑是结构上哪里需要变，我们就操作哪里。
2. 基本的封装性
3. 简单的生命周期呈现
4. 明确的数据流动
5. React组件的构建

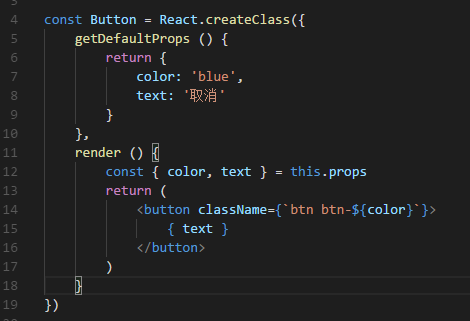
组件元素被描述成纯粹的JSON对象，意味着可以使用方法或类来构建。

React组件基本上由3个部分组成 属性（props）、状态（state）以及生命周期方法。

React组件可以接收参数，也可以由自身状态，一旦接收到的参数或自身状态有所改变，react组件就会执行相应的生命周期方法。最后渲染。整个过程完全符合传统组件所定义的组件职责。

1. React组件的构建方法
2. react组件基本上由组件的构建方式、组件内的属性状态与生命周期方法组成。
3. 官方在react组件构建上提供了3中不同的方法：React.createClass、es6 classes和无状态函数（seateless function）

（1）React.createClass



从表面看react.createClass方法就是构建一个组件对象，当另一个组件需要调用Button组件是，只用写成<Button/>， 就可以被解析成react.createElement(Button)方法来创建Button实例，这意味着在一个应用中调用几次Button，就会创建几次Button实例。

1. es6 classes

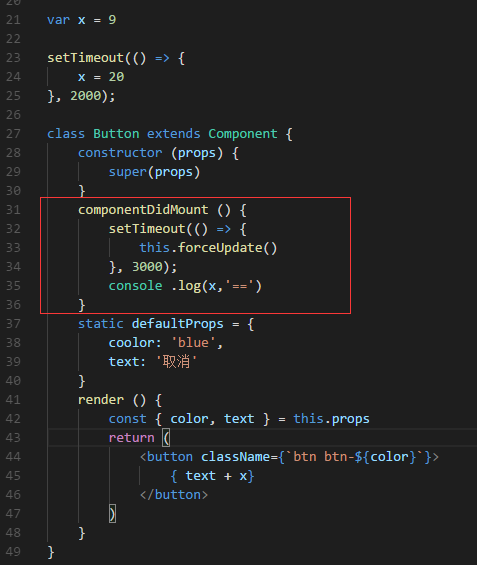


这里的直观感受是从调用内部方法变成了用类来实现。与createClass的结果相同的是，调用类实现的组件会创建实例对象

react所有组件都继承自顶层类React.Component，他的定义非常简洁，知识初始化了react.component方法，声明了props, context, refs等，并在原型上定义了setState和forceUpdate方法。内部初始化的生命周期方法与createClass方式使用的是同一个方法创建的。

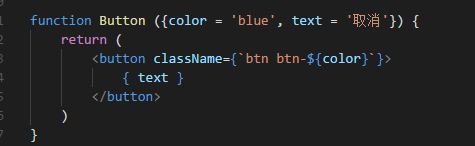
forceUpdate就是重新render。有些变量不在state上，但是你又想达到这个变量更新的时候，刷新render；或者state里的某个变量层次太深，更新的时候没有自动触发render。这些时候都可以手动调用forceUpdate自动触发render。所以建议使用immutable来操作state，redux等flux架构来管理state。

例如：



1. 无状态函数

使用无状态函数构建的组件称为无状态组件，这种构建方式，是官方颇为推崇的。示例：



**无状态组件**只传入props和context两个参数，也就是说**不存在state, 也没有生命周期方法**，没有上面两种react组件构建方法中的render方法。 不过像propTypes和defaultProps还是可以通过向方法设置静态属性来实现的。

在适合的情况下，我们都应该且必须使用无状态组件。无状态组件不像上述两种方法在调用时会创建新实例，它创建时始终保持了一个实例，避免了不必要的检查和内存分配，做到了内部优化

1. React数据流

React中，数据是自顶向下单向流动的，即从父组件到子组件。这条原则让组件之间的关系变得简单且可预测

State与props是react组件中最重要的概念。

如果顶层组件初始化props，那么react会向下遍历整个组件树，重新尝试渲染所有相关的子组件。

而state只关心每个组件自己内部的状态，这些状态只能在组件内改变。

把组件看成一个函数，那么他接收了props作为参数，内部由state作为函数的内部参数，返回一个Virtual DOM的实现。

1. State

在组件中想要更改state使用库那种的setState方法，setState是一个异步方法，一个生命周期内所有的setState方法会合并操作。

我们完全可以只用react完成对行为的控制、数据的更新和界面的渲染。然而，随着内容的深入，我们并不推荐开发者滥用state，过多的内部状态会让数据流混乱，程序变得难以维护。

1. Props

Props是properties的缩写。Props是react用来让组件之间互相联系的一种机制，通俗的说就像方法的参数一样。

Props的传递过程，对于react组件来说是非常直观的。主要的流动管道就是props。

Props本身是不可变的。

如果要渲染一个对props加工过的值，最简单的方法就是使用局部变量或者直接在jsx中计算结果

一般定义一个组件，会有以下几项：

1. class Name：根节点的class，为了方便覆盖其原始样式。
2. classPrefix：class前缀，对于组件来说，定义一个统一的class前缀，对样式的交互分离起了非常重要的作用。
3. 子组件props

React中有一个重要的内置prop——children，他代表组件的子组件的集合。

子组件集合的渲染通过react.children.map方法遍历子组件。React.children是官方提供的一系列操作children的方法，他提供如map\ forEatch\count等实用函数，可以为我们处理子组件提供便利。

使用react.children调用方式称为动态子组件，他指的是组件内的子组件是通过动态计算得到的。我们可以对任何数据、字符串、数组或对象进行动态计算。

除了map函数，还可以使用其他高阶函数，如reduce\filter等函数。值得注意的是，与map函数相似但不返回调用结果的forEatch函数不能这么使用

Props可以传入组件以及节点

1. 组件props

不仅可以使用this.props.children定义，也可以将子组件以props的形式传递。



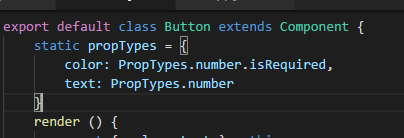
一般我们会用这种方式让开发者定义组件的某一个prop，让其具备多种类型，来做到简单配置和自定义配置组合在一起的效果。

这就是component props

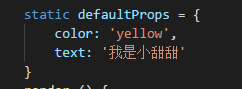
1. Function props与父组件的通信
2. PropTypes

javascript不是强类型语言，在没用保证的环境下写javascript已经习以为常了。强类型还是弱类型，正是开发时的约束问题，react对此做了妥协，便有了propTypes.

propTypes用于规范props的类型与必须的状态，如果定义了propTypes, 那么在开发环境下，就会对组件的props值的类型做检查，如果传入的props不能与其匹配，react则会在控制台报warning，在生产环境下，这是不会进行检查的。



Ps: 因为function和boolean在JavaScript中是关键词，所以propTypes中的用func和bool表示。



ClassNames插件