# React介绍

1. react简介
   1. React简介

Angular google

Vue

React是facebook在2013年开源在github上的javascript库。

Facebook认为MVC无法满足他们的扩展需求，由于他们非常巨大的代码库和庞大的组织，使得MVC很快变得非常复复杂，每当需要添加一项新的功能或特性时，系统的复杂度就成级数增长，致使代码变得脆弱和不可预测，结果导致他们的MVC正在土崩瓦解。认为MVC不适合大规模应用，当系统中有很多的模型和相应的视图时，其复杂度就会迅速扩大，非常难以理解和调试，特别是模型和视图间可能存在的双向数据流动。

* 1. 专注视图层view

现在的应用已经变得前所未有的复杂，因而开发工具也必须变得越来越强大，和angular等框架不同，react并不是完整的mvc/mvvm框架，他专注与提供清晰，简洁的view层解决方案。而又与模板引擎不同，react不仅专注于解决view层的问题，又是一个包括view和controller的库。对于复发的应用，可以更具应用场景自行选择业务层框架，并根据需要搭配flux、redux等来使用。

* 1. Virtual DOM

React把真实dom树转换成javascript对象树，也就是virtual DOM。

传统的web应用，操作DOM一般是直接更新操作的，但是我们知道DOM更新通常是比较昂贵的。

而React为了尽可能减少对DOM的操作，提供了一种不同的而又强大的方式来更新DOM，代替直接的DOM操作。就是Virtual DOM,一个轻量级的虚拟的DOM，就是React抽象出来的一个对象，描述dom应该什么样子的，应该如何呈现。通过这个Virtual DOM去更新真实的DOM，由这个Virtual DOM管理真实DOM的更新。

为什么通过这多一层的Virtual DOM操作就能更快呢？ 这是因为React有个diff算法，更新Virtual DOM并不保证马上影响真实的DOM，React会等到事件循环结束，然后利用这个diff算法，通过当前新的dom表述与之前的作比较，计算出最小的步骤更新真实的DOM。

每次数据更新后 ，重新计算virtual DOM，并和上一次生成的virtual DOM做对比，对发生变化的部分做批量更新。

我们说virtual DOM提升了react的性能，但这并不是唯一的亮点，此外，virtual DOM的渲染方式也比传统DOM操作好一些，但并不明显，因为对比DOM节点也是需要计算资源的。

它最大的好处其实还在于方便和其他平台集成，比如react-native是基于virtual DOM渲染出原生控件，因为react组件可以映射为对应的原生控件。在输出的时候，是输出web DOM还是Android控件，还是ios控件，就由平台决定了。因此，react-native有一个口号：learn Once，Write Anywhere.

* 1. 函数式编程

函数式编程，对应的是声明式编程，它是人类模仿自己逻辑思考方式发明出来的。

试想当我们操作数组的每个元素并返回一个新数组是，如果是计算机的思考方式则需要一个新数组，然后遍历原数组，并计算赋值； 如果是人类的思考方式，则是构建一个规则，这个过程就变成构建一个f函数作用在数组上，然后返回新数组。这样，计算可以被重复利用。

当回到ui界面上，我们产品经理又想出一个新点子时，我们时抱怨呢，还是思考怎么解决这个问题，react把过去不断重复构建UI的过程抽象成了组件，并且在给定参数的情况下约定渲染对应的UI界面，react能充分利用很多函数式方法去减少冗余代码。此外它本身就是简单函数，所以易于测试。可以说，函数式编程才是React的精髓。

* 1. Jsx由来和语法

这里的jsx语法，意在通过加强语法，使得JavaScript变得更快、更安全、更简单。

其实，虚拟元素可以理解为真实元素的对应，他的构建与更新都是在**内存中**完成的，并不会真正渲染到DOM中去。

在react中创建的虚拟元素可以分为**两类**，**DOM元素**（DOM element），与**组件元素**（Component element）,分别对应着原生DOM元素与自定义元素，而jsx与创建元素的过程有着莫大的关联。

1. dom元素

web页面时由一个个html元素嵌套组合而成的。当使用JavaScript1来描述这些元素时，这些元素可以被简单的表示成纯粹的JSON对象，例如：

<button class='button btn-blue'>

<em>我是子元素</em>

</button>

其中包括了元素的类型和属性，如果转成JSON对象，那么依然包括这些。

{

type: 'button',

props: {

className: 'btn btn-blue',

children: [{

type: 'em',

props: {

children: '我是子元素'

}

}]

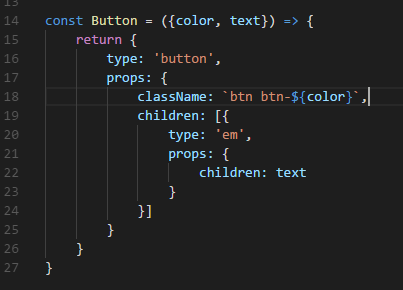
}

}

这样我们就可以在javascript中创建virtual DOM了

1. 组件元素

我们可以很方便的封装上述button元素，得到构建按钮的公共方法：

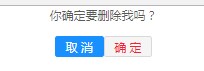


自然，当我们生成DOM元素中具体的按钮时，就可以方便的调用Button（{color：‘blue’, text: ‘我是子元素’}）来创建按钮。

通过上述案例，Button方法其实可以作为元素而存在（现在是个方法）。方法名对应 DOM元素类型，参数对应了DOM元素属性，那么它就具备了元素的两大必要条件（元素类型，dom元素属性），这样构建的元素就是自定义类型的元素，或称为组件元素

这也是react核心思想之一。因为有公共的表达方法，我们就可以让元素们彼此嵌套或混合。这些层层封装的组件元素，就是所谓的React组件，最终我们可以用递归渲染的方式构建出完全的DOM元素树。

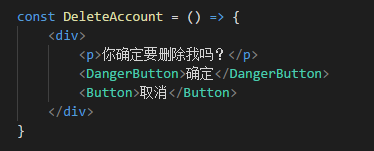
例如：如下div结构，用虚拟dom如何表示







如果让我们用虚拟dom编写dom结构，如果场景复杂，则力不从心了。这时我们便会想到html书写结构的畅快，jsx语法为此应运而生，假设我们使用jsx语法来重新表达以上组件，只需要这么写：



上述的例子并不是真实转换，如需要自己构建虚拟元素，原理是一样的。

在react中使用creatElement实现



Jsx基本语法

1. xml基本语法
2. 定义标签时，只允许被一个标签包裹
3. 标签一定要闭合
4. 元素类型
5. DOM元素
6. 组件元素
7. 注释，事实上，jsx还是javascript，在组件子元素位置使用注释要用{}包起来
8. DOCTYPE
9. 元素属性
10. JSX中不论DOM还是组件，都有属性，DOM元素的属性是标准规范属性，但有两个例外：class和for，因为这两个在js中是关键词：

Class属性改为className

for属性改为htmlfor

1. boolean属性，省略boolean属性会导致jsx认为bool值设为了true,要传false时，必须使用属性表达式，常用于表单元素，比如disabled, required, checked, readOnly等
2. 展开属性



在不知道要设置哪些属性时，可以使用rest/spread特性来提高效率



1. 自定义html属性

在jsx中往dom元素中传入自定义属性，react时不会渲染的，如果要使用html自定义属性，要使用data-前缀，这与html标准也是一致的

然而，在自定义标签中任意的属性都是被支持的

不论组件是用什么方法来写，我们都需要知道，组件的最终目的是输出虚拟元素也就是需要被渲染到界面的结构，其核心渲染方法，或称为组件输出方法，就是render方法。他是react生命周期的一部分，也是最核心的函数之一。

1. javascript属性表达式

属性值要使用表达式，只要用{}替换“”即可，组件也可以作为表达式使用

1. Html转义

React会将所有要显示到DOM的字符串转义，防止XSS攻击，所以，如果jsx中含有转义后的字符，比如：$copy; 最后都不会正确显示。解决办法有以下几个：

1. 直接使用UTF-8字符；
2. 使用对应字符的Unicode编码查询编码；
3. 使用数组组装<div>{{‘cc’, <span>&nbsp;</span>, ‘2015’}}</div>
4. 直接插入原始的html

此外，react提供了dangerouslySetInnerHTML属性，他的作用就是避免react转义字符，在确定必要的情况下使用他：



虚拟元素有两类，一个是