**React 还是 Vue: 你应该选择哪一个Web前端框架？**

2017年02月07日 14:11:53

阅读数：30412

学还是要学的，用的多了，也就有更多的认识了，开发中遇到选择的时候也就简单起来了。

本文作者也做了总结：

    如果你喜欢用（或希望能够用）模板搭建应用，请使用Vue  
    如果你喜欢简单和“能用就行”的东西，请使用Vue  
    如果你的应用需要尽可能的小和快，请使用Vue  
    如果你计划构建一个大型应用程序，请使用React  
    如果你想要一个同时适用于Web端和原生App的框架，请选择React  
    如果你想要最大的生态圈，请使用React  
    如果你已经对其中一个用得满意了，就没有必要换了

2016年，React在Web端和移动端都实现了迅速的成长，稳稳地领先于其主要竞争对手Augular，巩固了其作为前端框架之王的地位。

但是Vue在这一年里的表现也可谓同样的耀眼。Vue.js 2.0 的发布在整个JavaScript社区都引起了巨大反响，这一点仅从它在Github中涨了25,000颗star，就足以得到证明。

不得不说React和Vue的使用范围是相似的：都是基于组件化的轻量级框架，都专注于构建用户界面的视图层；都既可以用于简单的项目，也适用于使用前沿技术的大规模复杂项目。

因此，很多Web开发者都难免纠结到底要从中选择哪一个框架。这两者中能否分出个高下好坏？或者它们各自有哪些优缺点是需要我们注意的？还是说它们实际上差不多，用谁都一样？

**两种框架，两个拥护者**

在这篇文章中我想用尽可能公平，全面的对比来回答这些疑问。但是问题来了：我是个不折不扣的Vue迷弟，肯定会偏向它。今年我在项目中大量地使用了Vue，在[Medium](https://medium.com/" \t "_blank)上安利它的好处，甚至还在Udemy开设了一门[关于Vue的入门课程](https://www.udemy.com/vuejs-2-essentials" \t "_blank)。

为了平衡一下，我邀请了我的朋友Alexis Mangin一起参与讨论。他是一名优秀的Javascript开发者，且是React的铁粉。与我相似的是，他也频繁地在各种项目中（包括Web端和移动端项目）使用着React。

有一天他问我说：“为什么你这么喜欢用Vue，而不是React？”，当时由于我不太了解React，所以很难给出一个很好的回答。于是我向他提议，我们找一天带上各自的笔记本电脑，一起探讨我们各自喜爱的框架的好处。



Anthony（左）和 Alexis（右）在泰国清迈的Bull and Bear咖啡馆里比较React和Vue

经过大量地讨论和互相学习，我们得出了以下六个关键点：

**如果你喜欢用（或希望能够用）模板搭建应用，请使用Vue**

Vue应用的默认选项是把markup放入HTML文件中。数据绑定表达式使用的是与Angular相似的双大括号（moustache）语法，而指令（特殊的HTML属性）则用于向模板中添加功能。

下面是一个简单的Vue应用示例。它会展示一条消息（message），和一个用来动态反转这条消息的按钮：

1. *// HTML*
2. <div id="app">
3. <p>{{ message }}</p>
4. <button v-on:click="reverseMessage">Reverse Message</button>
5. </div>
6. *// JS*
7. new Vue({
8. el: '#app',
9. data: {
10. message: 'Hello Vue.js!
11. },
12. methods: {
13. reverseMessage: function () {
14. this.message = this.message.split('').reverse().join('');
15. }
16. }
17. });

React应用则与之相反，不使用模板，而是要求开发者借助JSX在JavaScript中创建DOM。下面是用React实现的相同的应用：

1. *// HTML*
2. <div id="app"></div>
3. *// JS (pre-transpilation)*
4. class App extends React.Component {
5. constructor(props) {
6. super(props);
7. this.state = {
8. message: 'Hello React.js!'
9. };
10. }
11. reverseMessage() {
12. this.setState({
13. message: this.state.message.split('').reverse().join('')
14. });
15. }
16. render() {
17. return (
18. <div>
19. <p>{this.state.message}</p>
20. <button onClick={() => this.reverseMessage()}>
21. Reverse Message
22. </button>
23. </div>
24. )
25. }
26. }
27. ReactDOM.render(App, document.getElementById('app'));

对于来自标准Web开发的新开发者来说，模板更容易理解。然而甚至连一些资深的开发者也喜欢使用模板，因为模板能更好地把功能和布局分割开来，还为使用Pug之类的模板引擎提供了可能。

不过使用模板的代价在于你需要学习所有的HTML扩展语法，而渲染函数（render function）只要求会使用标准的HTML和JavaScript。而且比起模板，渲染函数更易于调试和测试。但是，你并不应该因此就错过Vue，因为在Vue2.0中已经提供了使用模板或者渲染函数的选项。

**如果你喜欢简单和“能用就行”的东西，请使用Vue**

一个简单的Vue项目能不需要转译直接使用在浏览器中，这使得在项目中使用Vue可以像使用jQuery一样容易。当然这对于React来说在技术上也是可行的，但典型的React代码是更多地依赖于JSX和诸如class等ES6特性的。而Vue的简单则是更深层地源于其设计的。让我们来比较一下这两个框架是如何处理应用数据的（即“state”）：

React里的状态（state）是不可变（immutable）的，因此你不能直接地改变它，而是要用*setState* API方法：

1. this.setState({
2. message: this.state.message.split('').reverse().join('')
3. });

React是通过比较当前状态和前一个状态的区别来决定何时及如何重新渲染DOM的内容，因此需要使用不可变的状态。

而与此相反的是，Vue中的数据是可变的（mutated），所以同样的数据变量可以用简洁得多的方式修改：

1. *// Note that data properties are available as properties of*
2. *// the Vue instance*
3. this.message = this.message.split('').reverse().join('');

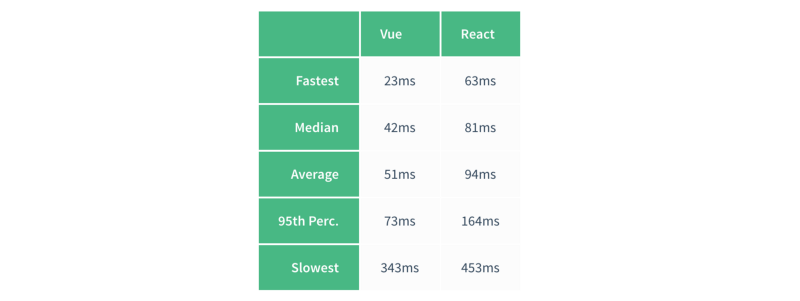
让我们来看看Vue中是如何对状态进行管理的：当你向状态中添加一个新对象时，Vue将遍历其中的所有属性并且将它们转换为getter，setter方法。于是Vue的响应系统开始保持对该状态的跟踪，当该状态的内容发生变化的时候就会自动重新渲染DOM。令人佩服的是，Vue中改变状态的操作不仅更加简洁，而且它的重新渲染系统实际上比React的更快更高效。

不过Vue的响应系统还是有些坑的，比如它无法检测属性的添加和删除，以及某些数组更改。这时候就要用Vue API中的类似于React的set方法的措施来解决。

**如果你的应用需要尽可能的小和快，请使用Vue**

当应用程序的状态改变时，React和Vue都会构建一个虚拟DOM并同步到真实DOM中。两者都有它们各自的优化这个过程的方式。

Vue的核心开发者提供了一个benchmark测试，来表明Vue的渲染系统比React的更快，具体基准的设定以及同其他框架的比较详见[vuejs.org](https://cn.vuejs.org/v2/guide/comparison.html" \t "_blank)。测试方法是将含有10000个项目的列表渲染100次，结果如下图。



从实用的角度来看，这种benchmark只跟边缘情况有关，而大部分应用程序中不会经常进行这种操作，所以这不应该被视作一个重要的比较点。但是，页面大小则是与所有项目有关的，这方面Vue又一次优于React，它目前的版本压缩后只有25.6KB。要用React实现同样的功能，你需要React DOM（37.4KB）和React with Addon库（11.4KB），共计44.8KB，几乎是Vue的两倍大。虽然从React你的确会得到更丰富的API，但双倍的体积并不能带来双倍的功能。

**如果你计划构建一个大型应用程序，请使用React**

像文章开头那样，用Vue和React实现的简单应用程序来比较两者，可能会让一个开发者从一开始就更倾向于Vue。这是因为基于模板的应用程序乍看上去更易理解，而且能很快地写好并跑起来。但是这些最初的便利会引入技术债并阻碍应用扩展到更大的规模。模板容易出现难以注意到的运行时错误，同时也不易于测试，重构和分解。

相比之下，Javascript模板可以组织成经过良好分解，且使用DRY（don't repeat yourself - 避免重复代码）原则的代码的组件，因而具有更强的可重用性和可测试性。Vue也有组件系统和渲染函数，但React的渲染系统可配置性更强，并包含如shallow rendering这样的特性，可结合React的测试工具一起使用，从而大为提高代码的可测试性及可维护性。

虽然React的不可变（immutable）应用状态写起来可能不够Vue简洁，但它在大型应用中仍会大放异彩，因为透明度和可测试性此时变得至关重要。

**如果你想要一个同时适用于Web端和原生App的框架，请选择React**

React Native是一个用于通过Javascript构建移动端原生应用程序的库。 它与React.js相同，只是不使用Web组件，而是使用原生组件。 如果你学过React.js，很快就能上手React Native，反之亦然。

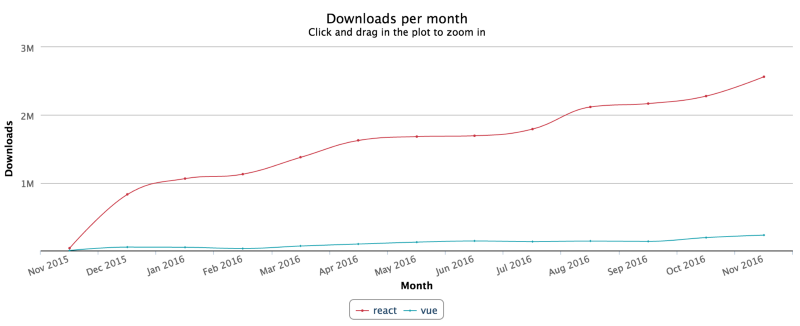
1. *// JS*
2. import React, { Component } from 'react';
3. import { AppRegistry, Text, View } from 'react-native';
4. class HelloWorld extends Component {
5. render() {
6. return (
7. <View>
8. <Text>Hello, React Native!</Text>
9. </View>
10. );
11. }
12. }
13. AppRegistry.registerComponent('HelloWorld', () => HelloWorld);

其意义在于，开发者只需要一套知识和工具就能开发Web应用和移动端原生应用。如果你想同时做Web端开发和移动端开发，学习React对你来说是相当划算的。

阿里的Weex也是一个跨平台UI项目，目前它以Vue为灵感，使用了许多相同的语法，并计划未来实现完全集成Vue，不过集成的时间和具体细节还未确定。由于Vue的设计中将HTML模板作为其核心部分之一，并且现有特性不支持自定义渲染，因此以Vue.js目前的形态来说，很难看到Weex与之的关系将能像React和React Native一样紧密。

**如果你想要最大的生态圈，请使用React**

毫无疑问，React目前比Vue要受欢迎许多——它在NPM上每个月的下载量约为250万次，而Vue只有22.5万次。



人气带来的好处不仅是表面上的名声，还意味着更多相关的技术文章，教程和更多Stack Overflow上的解答和帮助；以及有着更多的工具和插件可以在项目中使用，开发者不用从零开始，可以省很多力气。

这两个框架都是开源的，但是React诞生于Facebook，自带给力的资助，它的开发者和Facebook都承诺会持续维护React。而Vue则是由独立开发者尤雨溪创造，目前也只有他一名全职维护者。虽然也有一些公司资助Vue，但是规模和Facebook和Google没得比。

不过由于Vue的团队的努力，它的小规模和独立性并没有成为劣势。Vue有着固定的发布周期，甚至更令人称道的是，Github上Vue只有54个待解决问题（open issue），已关闭问题（closed issue）则有3456个；与之相比，React的已关闭问题数目相差不多（3447个），待解决问题却有多达530个。

**如果你已经对其中一个用得满意了，就没有必要换了**

总结一下，我们发现，Vue的优势包括：

* 模板和渲染函数的弹性选择
* 简单的语法及项目创建
* 更快的渲染速度和更小的体积

React的优势包括：

* 更适用于大型应用和更好的可测试性
* 同时适用于Web端和原生App
* 更大的生态圈带来的更多支持和工具

而实际上，React和Vue都是非常优秀的框架，它们之间的相似之处多过不同之处，并且它们大部分最棒的功能是相通的：

* 利用虚拟DOM实现快速渲染
* 轻量级
* 响应式组件
* 服务器端渲染
* 易于集成路由工具，打包工具以及状态管理工具
* 优秀的支持和社区

文章标签： [react](http://so.csdn.net/so/search/s.do?q=react&t=blog" \t "_blank)[vue](http://so.csdn.net/so/search/s.do?q=vue&t=blog)

个人分类： [java](https://blog.csdn.net/yzh_2017/article/category/6601748)

相关热词： [react的](https://blog.csdn.net/wanshaobo888/article/details/59090239) [react的if](https://blog.csdn.net/m0_37355951/article/details/72855583) [react](https://blog.csdn.net/fly821760648/article/details/80553574) [与react结合](https://blog.csdn.net/fay462298322/article/details/78361579) [启动react](https://blog.csdn.net/u014744233/article/details/51777674)