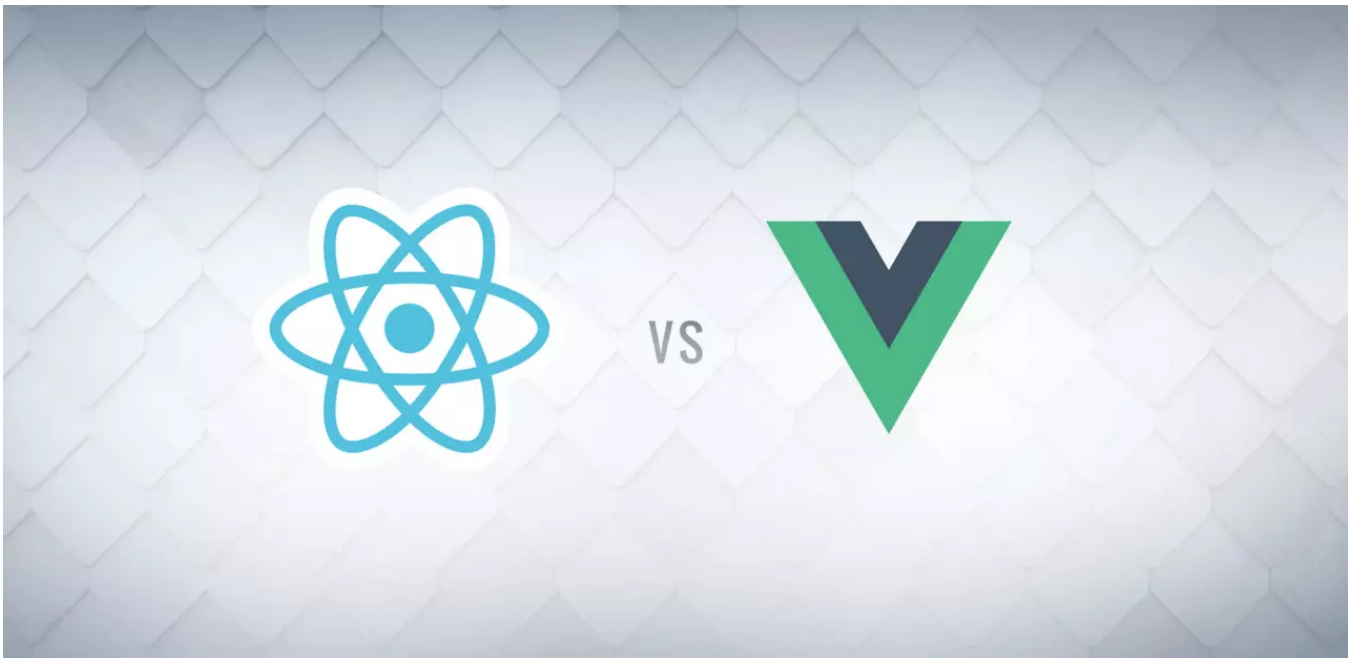
Vue vs React: Javascript 框架之战

阅读 3163 收藏 98 2017-06-27
原文链接：zcfy.cc



正如我们之前提到的，WordPress 的核心团队正争论着为应该将哪款（前端框架）加入现在的架构之中。目前看来，暂时脱颖而出的是React与Vue.js，社区中的很多成员正权衡着这两款框架的利弊。

那到底哪款框架会胜出，哪款框架又会沦为昔日的prototype.js.呢？让我们一起看看吧。

我已经写出了两个几乎一样的Web应用，一个是基于Vue，另一个则基于React，可以方便你在看这篇文章的时候查找相关代码。

- React sample app

简单介绍

除非你最近一直不关注前端的发展，不然你肯定听说过由Facebook创建的JavaScript UI框架——React。它支撑着包括Instagram在内的大多数Facebook网站。React与当时流行的jQuery，Backbone.js和 Angular 1等框架不同，它的诞生改变了JavaScript的世界。其中最大的变化是React推广了Virtual DOM（我们稍后探究）并创造了新的语法——JSX，JSX允许开发者在JavaScript中书写HTML（译者注：即HTML in JavaScript）。WAT?

Vue致力解决的问题与React一致，但却提供了另外一套解决方案。Vue使用模板系统而不是JSX，使其对现有应用的升级更加容易。这是因为模板用的就是普通的HTML，通过Vue来整合现有的系统是比较容易的，不需要整体重构。同时Vue声称它 更容易学习，我最近才接触Vue，能证明所言非虚。关于Vue还需要说的是，Vue主要是由 一位开发者进行维护的，而不像React一样由如Facebook这类大公司维护。

相似之处

React与Vue存在很多相似之处，例如他们都是JavaScript的UI框架，专注于创造前端的富应用。不同于早期的JavaScript框架“功能齐全”，Reat与Vue只有框架的骨架，其他的功能如路由、状态管理等是框架分离的组件。

Virtual DOM

啊哈，人们经常说Virtual DOM是什么呢?

生是基于这么一个概念：改变真实的DOM状态远比改变一个JavaScript对象的花销要大得多。

Virtual DOM是一个映射真实DOM的JavaScript对象，如果需要改变任何元素的状态，那么是先在Virtual DOM上进行改变，而不是直接改变真实的DOM。当有变化产生时，一个新的Virtual DOM对象会被创建并计算新旧Virtual DOM之间的差别。之后这些差别会应用在真实的DOM上。

例子如下，我们可以看看下面这个列表在HTML中的代码是如何写的：

```
<ul class="list">
  <li>item 1</li>
  <li>item 2</li>
</ul>
```

而在JavaScript中，我们可以用对象简单地创造一个针对上面例子的映射：

```
{
    type: 'ul',
    props: {'class': 'list'},
    children: [
        { type: 'li', props: {}, children: ['item 1'] },
        { type: 'li', props: {}, children: ['item 2'] }
    ]
}
```

真实的Virtual DOM会比上面的例子更复杂，但它本质上是一个嵌套着数组的原生对象。

当新一项被加进去这个JavaScript对象时，一个函数会计算新旧Virtual DOM之间的差异并反应在真实的DOM上。计算差异的算法是高性能框架的秘密所在，React和Vue在实现上有点不同。

Vue宣称可以更快地计算出Virtual DOM的差异，这是由于它在渲染过程中，会跟踪每一个组件的依赖关系，不需要重新渲染整个组件树。

而对于React而言，每当应用的状态被改变时，全部子组件都会重新渲染。当然，这可以通过 `shouldComponentUpdate` 这个生命周期方法来进行控制，但Vue将此视为默认的优化。

小结：如果你的应用中，交互复杂，需要处理大量的UI变化，那么使用Virtual DOM是一个好主意。如果你更新元素并不频繁，那么Virtual DOM并不一定适用，性能很可能还不如直接操控DOM。

组件化

被找到:

> 你可以认为组件就是用户界面中的一小块。如果让我来设计Facebook的UI界面，那么聊天窗口会是一个组件，评论会是另一个组件，不断更新的好友列表也会作为一个组件。

在Vue中，如果你遵守一定的规则，你可以使用单文件组件.

```
//PastaItem.vue

<template>
<li class="pasta-dish list-unstyled">
    <div class="row">
        <div class="col-md-3">
            <img :src="this.item.image" :alt="this.item.name" />
        </div>
        <div class="col-md-9 text-left">
            <h3>{{this.item.name}}</h3>
            <p>
                {{this.item.desc}}
            </p>
            <button v-on:click="addToOrderNew" class="btn btn-primary">Add to order</button
        </div>
    </div>
</li>
</template>

<script>

export default {
    name: 'pasta-item',
    props: ['item'],
    data:  function(){
        return{
            orders: 0
        }
    },
    methods: {
        addToOrderNew: function(y){
            this.orders += 1;
            this.$emit('order');
        }
    }
}
```

```
<style src="./Pasta.css"></style>
```

正如上面你看到的例子中，HTML, JavaScript和CSS都写在一个文件之中。你不再需要在 `.vue` 组件文件中引入CSS，虽然这也是可以的。

React也是非常相似的，JavaScript与JSX被写入同一个组件文件中。

```
import React from "react";

class PastaItem extends React.Component {

    render() {
        const { details, index } = this.props;

        return (
            <li className="pasta-dish list-unstyled">
                <div className="row">
                    <div className="col-md-3">
                        <img src={details.image} alt={details.name} />
                    </div>
                    <div className="col-md-9 text-left">
                        <h3>{details.name}</h3>
                        <p>
                            {details.desc}
                        </p>
                        <button onClick={() => this.props.addToOrder(index)} className="btn
                    </div>
                </div>
            </li>
        );
    }
}

export default PastaItem;
```

Props

在上面两个例子中，我们可以看到React和Vue都有'props'的概念，这是properties的简写。props在组件中是一个特殊的属性，允许父组件往子组件传送数据。

```
Object.keys(this.state.pastadishes).map(key =>
```

上面的JSX库组中， `index` ， `key` ， `details` ， `orders` 与 `addToOrder` 都是props，数据会被下传到子组件 `PastaItem` 中去。

在React中，这是必须的，它依赖一个“单一数据源”作为它的“状态”（稍后有更多介绍）。

而在Vue中，props略有不同。它们一样是在组件中被定义，但Vue依赖于模板语法，你可以通过模板的循环函数更高效地展示传入的数据。

```
<pasta-item v-for="(item, key) in samplePasta" :item="item" :key="key" @order="handleOrder(
```

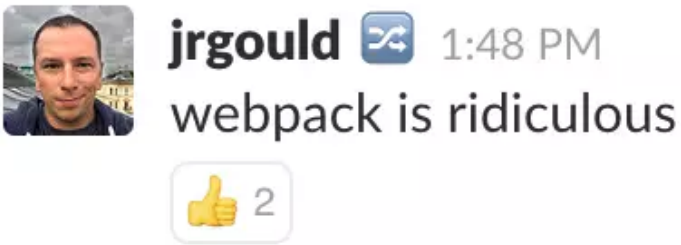
这是模板的实现，但这代码完全能工作，然而在React中展现相同数据会更麻烦一点。

构建工具

React和Vue都有自己的构建工具，你可以使用它快速搭建开发环境。React可以使用Create React App (CRA)，而Vue对应的则是vue-cli。两个工具都能让你得到一个根据最佳实践设置的项目模板。

由于CRA有很多选项，使用起来会稍微麻烦一点。这个工具会逼迫你使用Webpack和Babel。而vue-cli则有模板列表可选，能按需创造不同模板，使用起来更灵活一点。

事实上说，两个工具都非常好用，都能为你建立一个好环境。而且如果可以不配置Webpack的话，我和Jeff认为这是天大的好事。



Chrome 开发工具

React和Vue都有很好的Chrome扩展工具去帮助你找出bug。它们会检查你的应用，让你看到Vue或者React中的变化。你也可以看到应用中的状态，并实时看到更新。

React的开发工具: cdn.deliciousbrains.com/content/upl…

Vue的开发工具: cdn.deliciousbrains.com/content/upl…

配套框架

Vue与React最后一个相似但略有不同之处是它们配套框架的处理方法。相同之处在于，两个框架都专注于UI层，其他的功能如路由、状态管理等都交由同伴框架进行处理。

而不同之处是在于它们如何关联它们各自的配套框架。Vue的核心团队维护着vue-router和vuex，它们都是作为官方推荐的存在。而React的react-router和 react-redux则是由社区成员维护，它们都不是官方维护的。

主要区别

Vue与react有很多的相似之处，但他们也有完全不一致的地方。

模板 vs JSX

React与Vue最大的不同是模板的编写。Vue鼓励你去写近似常规HTML的模板。写起来很接近标准HTML元素，只是多了一些属性。

```
<ul>
    <template v-for="item in items">
        <li>{{ item.msg }}</li>
        <li class="divider"></li>
    </template>
</ul>
```

这些属性也可以被使用在单文件组件中，尽管它需要在在构建时将组件转换为合法的JavaScript和HTML。

```
<ul>
  <pasta-item v-for="(item, key) in samplePasta" :item="item" :key="key" @order="handleOrde
</ul>
```

Vue鼓励你去使用HTML模板去进行渲染，使用相似于Angular风格的方法去输出动态的内容。因此，通过把原有的模板整合成新的Vue模板，Vue很容易提供旧的应用的升级。这也让新来者很容易适应它的语法。

另一方面，React推荐你所有的模板通用JavaScript的语法扩展——JSX书写。同样的代码，用JSX书写的例子如下：

```
<ul className="pasta-list">
    {
        Object.keys(this.state.pastadishes).map(key =>
            <PastaItem index={key} key={key} details={this.state.pastadishes[key]} addToOrd
```

```
    }
</ul>
```

React/JSX乍看之下，觉得非常啰嗦，但使用JavaScript而不是模板来开发，赋予了开发者许多编程能力。

但请记住：

> 能力越大，责任越大。 Ben Parker

JSX只是JavaScript混合着XML语法，然而一旦你掌握了它，它使用起来会让你感到畅快。这可能只是我个人的意见，但我觉得这比Angular 1风格的属性好多了，Angular 1真的难以忍受。

而相反的观点是Vue的模板语法去除了往视图/组件中添加逻辑的诱惑，保持了关注点分离。

值得一提的是，与React一样，Vue在技术上也支持render函数和JSX，但只是不是默认的而已。

状态管理 vs 对象属性

如果你对React熟悉，你就会知道应用中的状态是（React）关键的概念。也有一些配套框架被设计为管理一个大的state对象，如Redux。此外，state对象在React应用中是不可变的，意味着它不能被直接改变（ 这也许不一定正确）。在React中你需要使用 `setState()` 方法去更新状态。

```
addToOrder(key) {
        //Make a copy of this.state
        const orders = { ...this.state.orders };

        //update or add
        orders[ key ] = orders[ key ] + 1 || 1;
        this.setState( { orders } );
}
```


在Vue中，state对象并不是必须的，数据由data属性在Vue对象中进行管理。

```
export default {
  name: 'app',
  data() {
    return {
      samplePasta: samplePasta,
      orders: {}
```

```
...
  methods: {
    handleOrder: function (key) {

      if (!this.orders.hasOwnProperty(key)) {
        this.$set(this.orders, key, { count: 0 });
      }

      this.orders[key].count += 1;
    }
  }
}
```

而在Vue中，则不需要使用如 `setState()` 之类的方法去改变它的状态，在Vue对象中，data参数就是应用中数据的保存者。

对于管理大型应用中的状态这一话题而言，Vue.js的作者尤雨溪曾说过，（Vue的）解决方案适用于小型应用，但对于对于大型应用而言不太适合。

> 多数情况下，框架内置的状态管理是不足以支撑大型应用的，Redux或Vuex等状态管理方案是必须使用的。

有鉴于此，争论你的应用中如何管理状态很可能属于过早优化，并且这很可能只是个人偏好问题。此外，你可能真没必要担心这方面。

React Native vs. ?

React Native能在手机上创建原生应用，React在这方面处于领先位置。使用JavaScript, CSS和HTML创建原生移动应用，这是一个重要的革新。Vue社区与阿里合作开发Vue版的React Native——Weex也很不错，但仍处于开发状态且并没经过实际项目的验证。

那么，谁赢得这场竞赛呢?

TLDR; JavaScript是没前途的, 一起使用TypeScript吧

如果你想在求职市场中有光辉的JavaScript UI框架使用经验，那么React和Vue都是不错的选择。React似乎更加流行，但Vue也变得越来越广为人知。如果你想将现有的应用升级为新的JavaScript框架，那么我建议你选择Vue。实际上，当你想创建一个现代的JavaScript应用时，两者都是很好的选择，取决于你的偏好。

你对于React或者Vue的想法是什么呢？赶快发表评论告诉我们吧！

前端框架 Vue.js React.js

相关热门文章

React Router 4 简介及其背后的路由哲学

rccoder 22

快速掌握react-motion动画库

铁锅 17

技术胖155集前端视频教程-全部免费观看

技术胖 2224 96

laravel5.3 vue 实现收藏夹功能

iMax 16

做个简单的 React-Native application 处理旧书

三毛丶 10

评论

说说你的看法

刘明明 技术负责人

没意思。都是解决实际问题 哪个用起来顺手就用那个。没必要比

▲ 1 评论 6月前

LockLogic 软件开发工程师

感觉上手确实很快，但是实际看到github上别人用这个框架开发的项目，源码读起来就没有那么容易了。

▲ 0 评论 6月前

SuYe 前端工程师 @ 苏叶科技

个人觉得Vue上手不如react简单啊。。

▲ 0 1条评论 6月前

Ly html6 @ 爱我文化传媒

不同的需求，不同的产品，不同的成员都可能不一样，没有好坏之分，只有更适合。