更轻更快的 Vue.js 2.0 与其他框架对比

2016-10-23 野狗

✦✦✦✦✦✦✦✦

01-

更轻更快的Vue.js 2.0

崭露头角的JavaScript框架Vue.js 2.0版本已经发布，在狂热的JavaScript世界里带来了让人耳目一新的变化。

Vue创建者尤雨溪称，Vue 2.0 在性能上有显著的提升，同时保持轻量的文件下载：

> 渲染层基于一个轻量级的Virtual DOM实现进行了重写，该Virtual DOM实现fork自snabbdom。新的渲染层相比v1带来了巨大的性能提升，也让Vue 2.0成为了最快速的框架之一。

根据1.0到2.0迁移指南，“大约90%的API是相同的”。不同于一些JavaScript框架，从一个版本到下一个版本会有巨大的变化，在Vue 2.0中“核心概念没有改变”。

除了核心库，vue-router和vuex库已经更新到2.0版本，vue-cli已经默认生成了 2.0 的脚手架。

在Vue 2.0第一次宣布发布时，尤雨溪说过滤器要离开了。然而随着时间的推移，这个立场改变了，Vue 2.0中仍然保留了文本插入过滤器。的确，在使用指南过滤器这一章节也指明了“Vue 2.x中过滤器仅能在mustache内部绑定使用。要在指令绑定中实现相同的行为，应该使用计算属性。”框架不再提供任何内置过滤器，并且数组过滤器（削减了基于标准的数组）也已经取消了。

因为Vue主要不是由大型技术公司创建的，尤雨溪采取了不同的融资方式。尤雨溪能够在这个项目全职工作要感谢Patreon，在Patreon上尤雨溪设置的赞助方式是每月保证一定数量的金额。截至发稿时，Patreon每月为Vue的可持续发展赞助8853美元。除了尤雨溪之外，Vue.js核心团队还有18个人。

社区里大多是积极的回应，“Vue比ng2更简洁”以及“我发现Vue 2.0介于Angular 1和React之间。它是现代化的并且很容易学习和使用”。

Vue在官网上发布了对比指南（http://vuefe.cn/guide/comparison.html），展示了它与其他框架的不同之处，本文摘录其与如Angular和React对比。

02-
对比 React

React 和 Vue 有许多相似之处，它们都有：

- 使用 Virtual DOM
- 提供了响应式（Reactive）和组件化（Composable）的视图组件。
- 将注意力集中保持在核心库，伴随于此，有配套的路由和负责处理全局状态管理的库。

由于有着众多的相似处，我们会用更多的时间在这一块进行比较。这里我们不只保证技术内容的准确性，同时也兼顾了平衡的考量。我们需要指出 React 比 Vue 更好的地方，像是他们的生态系统和丰富的自定义渲染器。

React社区为我们准确进行平衡的考量提供了非常积极地帮助，特别感谢来自 React 团队的 Dan Abramov 。他非常慷慨的花费时间来贡献专业知识，帮助我们完善这篇文档，最后我们对最终结果都十分满意。

有了上面这些内容，我们希望你能对下面这两个库的比较内容的公正性感到放心。

性能简介

到目前为止，针对现实情况的测试中，Vue 的性能是优于 React 的。如果你对此表示怀疑，请继续阅读。我们会解释为什么会这样（并且会提供一个与 React 团队共同约定的比较基准）。

渲染性能

在渲染用户界面的时候，DOM 的操作成本是最高的，不幸的是没有库可以让这些原始操作变得更快。

我们能做到的最好效果就是：

1. 尽量减少 DOM 操作。Vue 和 React 都使用虚拟 DOM 来实现，并且两者工作的效果一样好。

2. 尽量减少除 DOM 操作以外的其他操作。这是 Vue 和 React 所不同的地方。

在 React 中，我们设定渲染一个元素的额外开销是 1，而平均渲染一个组件的开销是 2。那么在 Vue 中，一个元素的开销更像是 0.1，但是平均组件的开销将会是 4，这是由于我们需要设定响应系统所导致的。

这意味着：在典型的应用中，由于需要渲染的元素比组件的数量是更多的，因此 Vue 的性能表现将会远优于 React。然而，在极端情况下，比如每个组件只渲染一个元素，Vue 就会通常更慢一些。当然接下来还有其他的原因。

Vue 和 React 也提供功能性组件，这些组件因为都是没有声明，没有实例化的，因此会花费更少的开销。当这些都用于关键性能的场景时，Vue 将会更快。为了证明这点，我们建立了一个简单的参照项目（https://github.com/chrisvfritz/vue-render-performance-comparisons），它负责渲染 10,000 个列表项 100 次。我们鼓励你基于此去尝试运行一下。然而在实际上，由于浏览器和硬件的差异甚至 JavaScript 引擎的不同，结果都会相应有所不同。

如果你懒得去做，下面的数值是来自于一个 2014 年产的 MacBook Air 并在 Chrome 52 版本下运行所产生的。为了避免偶然性，每个参照项目都分别运行 20 次并取自最好的结果：

	Vue	React
Fastest	23ms	63ms
Median	42ms	81ms
Average	51ms	94ms
95th Perc.	73ms	164ms
Slowest	343ms	453ms

更新性能

在 React 中，你需要在处处去实现 shouldComponentUpdate，并且用不可变数据结构才能实现最优化的渲染。在 Vue 中，组件的依赖被自动追踪，所以当这些依赖项变动时，它才会更新。唯一需要注意的可能需要进一步优化的一点是在长列表中，需要在每项上添加一个 key 属性。

这意味着，未经优化的 Vue 相比未经优化的 React 要快的多。由于 Vue 改进过渲染性能，甚至全面优化过的 React 通常也会慢于开箱即用的 Vue。

开发中

显然，在生产环境中的性能是至关重要的，目前为止我们所具体讨论的便是针对此环境。但开发过程中的表现也不容小视。不错的是用 Vue 和 React 开发大多数应用的速度都是足够快的。

然而，假如你要开发一个对性能要求比较高的数据可视化或者动画的应用时，你需要了解到下面这点：在开发中，Vue 每秒最高处理 10 帧，而 React 每秒最高处理不到 1 帧。

这是由于 React 有大量的检查机制，这会让它提供许多有用的警告和错误提示信息。我们同样认为这些是很重要的，但是我们在实现这些检查时，也更加密切地关注了性能方面。

HTML & CSS

在 React 中，它们都是 JavaScript 编写的，听起来这十分简单和优雅。然而不幸的事实是，JavaScript 内的 HTML 和 CSS 会产生很多痛点。在 Vue 中我们采用 Web 技术并在其上进行扩展。接下来将通过一些实例向你展示这意味的是什么。

JSX vs Templates

在 React 中，所有的组件的渲染功能都依靠 JSX。JSX 是使用 XML 语法编写 Javascript 的一种语法糖。这有一个通过React社区审核过的例子：

```
render () {
    let { items } = this.props
    let children
    if ( items.length > 0 ) {
        children = (
            <ul>
                {items.map( item =>
                    <li key={item.id}>{item.name}</li>
                )}
            </ul>
        )
    } else {
        children = <p>No items found.</p>
    }
    return (
        <div className = 'list-container'>
            {children}
        </div>
    )
}
```

JSX 的渲染功能有下面这些优势：

- 你可以使用完整的编程语言 JavaScript 功能来构建你的视图页面。
- 工具对 JSX 的支持相比于现有可用的其他 Vue 模板还是比较先进的（比如，linting、类型检查、编辑器的自动完成）。

在 Vue 中，由于有时需要用这些功能，我们也提供了渲染功能 并且支持了 JSX。然而，对于大多数组件来说，渲染功能是不推荐使用了。

在这方面，我们提供的是更简单的模板：

```
<template>
    <div class="list-container">
        <ul v-if="items.length">
            <li v-for="item in items">
                {{ item.name }}
            </li>
        </ul>
        <p v-else>No items found.</p>
    </div>
</template>
```

优点如下：

- 在写模板的过程中，样式风格已定并涉及更少的功能实现。
- 模板总是会被声明的。
- 模板中任何 HTML 语法都是有效的。
- 阅读起来更贴合英语（比如，for each item in items）。
- 不需要高级版本的 JavaScript 语法，来增加可读性。

这样，不仅开发人员更容易编写代码，设计人员和开发人员也可以更容易的分析代码和贡献代码。

这还没有结束。Vue 拥抱 HTML，而不是用 JavaScript 去重塑它。在模板内，Vue 也允许你用预处理器比如 Pug（原名 Jade）。

React 生态也有一个项目(https://wix.github.io/react-templates/)允许你写模板，但是存在一些缺点：

- 功能远没有 Vue 模板系统丰富。
- 需要从组件文件中分离出 HTML 代码。
- 这是个第三方库，而非官方支持，可能未来核心库更新就不再支持。

CSS 的组件作用域

除非你把组件分布在多个文件上(例如 CSS Modules)，要不在 React 中作用域内的 CSS 就会产生警告。非常简单的 CSS 还可以工作，但是稍微复杂点的，比如悬停状态、媒体查询、伪类选择符等要么通过沉重的依赖来重做要么就直接不能用。

而 Vue 可以让你在每个单文件组件中完全访问 CSS。

```
<style scoped>
    @media (min-width: 250px) {
        .list-container:hover {
            background: orange;
        }
    }
</style>
```

这个可选 scoped 属性会自动添加一个唯一的属性（比如 data-v-1）为组件内 CSS 指定作用域，编译的时候 .list-container:hover 会被编译成类似 .list-container[data-v-1]:hover。

最后，就像 HTML 一样，你可以选择自己偏爱的 CSS 预处理器编写 CSS。这可以让你围绕设计为中心展开工作，而不是引入专门的库来增加你应用的体积和复杂度。

规模

向上扩展

Vue 和 React 都提供了强大的路由来应对大型应用。React 社区在状态管理方面非常有创新精神（比如 Flux、Redux），而这些状态管理模式甚至 Redux 本身也可以非常容易的集成在 Vue 应用中。实际上，Vue 更进一步地采用了这种模式（Vuex），更加深入集成 Vue 的状态管理解决方案 Vuex 相信能为你带来更好的开发体验。

两者另一个重要差异是，Vue 的路由库和状态管理库都是由官方维护支持且与核心库同步更新的。React 则是选择把这些问题交给社区维护，因此创建了一个更分散的生态系统。但相对的，React 的生态系统相比 Vue 更加繁荣。

最后，Vue 提供了Vue-cli 脚手架，能让你非常容易地构建项目，包含了 Webpack, Browserify, 甚至 no build system。React 在这方面也提供了create-react-app，但是现在还存在一些局限性：

- 它不允许在项目生成时进行任何配置，而 Vue 支持 Yeoman-like 定制。
- 它只提供一个构建单页面应用的单一模板，而 Vue 提供了各种用途的模板。
- 它不能用用户自建的模板构建项目，而自建模板对企业环境下预先建立协议是特别有用的。

而要注意的是这些限制是故意设计的，这有它的优势。例如，如果你的项目需求非常简单，你就不需要自定义生成过程。你能把它作为一个依赖来更新。如果阅读更多关于不同的设计理念。

向下扩展

React 学习曲线陡峭，在你开始学 React 前，你需要知道 JSX 和 ES2015，因为许多示例用的是这些语法。你需要学习构建系统，虽然你在技术上可以用 Babel 来实时编译代码，但是这并不推荐用于生产环境。

就像 Vue 向上扩展好比 React 一样，Vue 向下扩展后就类似于 jQuery。你只要把如下标签放到页面就可以运行：

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>
```

然后你就可以编写 Vue 代码并应用到生产中，你只要用 min 版 Vue 文件替换掉就不用担心其他的性能问题。

由于起步阶段不需学 JSX，ES2015 以及构建系统，所以开发者只需不到一天的时间阅读指南就可以建立简单的应用程序。

本地渲染

ReactNative 能使你用相同的组件模型编写有本地渲染能力的 APP（IOS 和 Android）。能同时跨多平台开发，对开发者是非常棒的。相应地，Vue 和 Weex 会进行官方合作，Weex 是阿里的跨平台用户界面开发框架，Weex 的 JavaScript 框架运行时用的就是 Vue。这意味着在 Weex 的帮助下，你使用 Vue 语法开发的组件不仅仅可以运行在浏览器端，还能被用于开发 IOS 和 Android 上的原生应用。

在现在，Weex 还在积极发展，成熟度也不能和 ReactNative 相抗衡。但是，Weex 的发展是由世界上最大的电子商务企业的需求在驱动，Vue 团队也会和 Weex 团队积极合作确保为开发者带来良好的开发体验。

MobX

Mobx 在 React 社区很流行，实际上在 Vue 也采用了几乎相同的反应系统。在有限程度上，React + Mobx 也可以被认为是更繁琐的 Vue，所以如果你习惯组合使用它们，那么选择 Vue 会更合理。

03-
对比Angular 1

Vue 的一些语法和 Angular 的很相似（例如 v-if vs ng-if）。因为 Angular 是 Vue 早期开发的灵感来源。然而，Augular 中存在的许多问题，在 Vue 中已经得到解决。

复杂性

在 API 与设计两方面上 Vue.js 都比 Angular 1 简单得多，因此你可以快速地掌握它的全部特性并投入开发。

灵活性和模块化

Vue.js 是一个更加灵活开放的解决方案。它允许你以希望的方式组织应用程序，而不是在任何时候都必须遵循 Angular 1 制定的规则，这让 Vue 能适用于各种项目。我们知道把决定权交给你是非常必要的。
这也就是为什么我们提供Webpack template，让你可以用几分钟，去选择是否启用高级特性，比如热模块加载、linting、CSS 提取等等。

数据绑定

Angular 1 使用双向绑定，Vue 在不同组件间强制使用单向数据流。这使应用中的数据流更加清晰易懂。

指令与组件

在 Vue 中指令和组件分得更清晰。指令只封装 DOM 操作，而组件代表一个自给自足的独立单元 —— 有自己的视图和数据逻辑。在 Angular 中两者有不少相混的地方。

性能

Vue 有更好的性能，并且非常非常容易优化，因为它不使用脏检查。

在 Angular 1 中，当 watcher 越来越多时会变得越来越慢，因为作用域内的每一次变化，所有 watcher 都要重新计算。并且，如果一些 watcher 触发另一个更新，脏检查循环（digest cycle）可能要运行多次。Angular 用户常常要使用深奥的技术，以解决脏检查循环的问题。有时没有简单的办法来优化有大量 watcher 的作用域。

Vue 则根本没有这个问题，因为它使用基于依赖追踪的观察系统并且异步队列更新，所有的数据变化都是独立触发，除非它们之间有明确的依赖关系。

有意思的是，Angular 2 和 Vue 用相似的设计解决了一些 Angular 1 中存在的问题。

04-
对比 Angular 2

我们单独将 Augluar 2 作分类，因为它完全是一个全新的框架。例如：它具有优秀的组件系统，并且许多实现已经完全重写，API 也完全改变了。

TypeScript

Angular 1 面向的是较小的应用程序，Angular 2 已转移焦点，面向的是大型企业应用。在这一点上 TypeScript 经常会被引用，它对那些喜欢用 Java 或者 C# 等类型安全的语言的人是非常有用的。

Vue 也十分适合制作企业应用，你也可以通过使用官方类型或用户贡献的装饰器来支持 TypeScript，这完全是自由可选的。

大小和性能

在性能方面，这两个框架都非常的快。但目前尚没有足够的数据用例来具体展示。如果你一定要量化这些数据，你可以查看第三方参照，它表明 Vue 2 相比 Angular2 是更快的。

在大小方面，虽然 Angular 2 使用 tree-shaking 和离线编译技术使代码体积减小了许多。但包含编译器和全部功能的 Vue2(23kb) 相比 Angular 2(50kb) 还是要小的多。但是要注意，用 Angular 2 的 App 的体积缩减是使用了 tree-shaking 移除了那些框架中没有用到的功能，但随着功能引入的不断增多，尺寸会变得越来越大。

灵活性

Vue 相比于 Angular 2 则更加灵活，Vue 官方提供了构建工具来协助你构建项目，但它并不限制你去如何构建。有人可能喜欢用统一的方式来构建，也有很多开发者喜欢这种灵活自由的方式。

学习曲线

开始使用 Vue，你使用的是熟悉的 HTML、符合 ES5 规则的 JavaScript（也就是纯 JavaScript）。有了这些基本的技能，你可以快速地掌握它(指南)并投入开发 。

Angular 2 的学习曲线是非常陡峭的。即使不包括 TypeScript，它的开始指南中所用的就有 ES2015 标准的 JavaScript，18个 NPM 依赖包，4 个文件和超过 3 千多字的介绍，这一切都是为了完成个 Hello World。而Vue's Hello World就非常简单。甚至我们并不用花费一整个页面去介绍它。

✦✦✦✦✦✦✦✦

作者：David Iffland

译者：任美芒

点击“阅读原文”，看更多

精选文章

↓↓↓

阅读原文