vue

API设计上简单，语法简单，学习成本低

构建方面不包含路由和ajax功能，使用vuex, vue-router

指令（dom）和组件（视图，数据，逻辑）处理清晰

性能好，容易优化

基于依赖追踪的观察系统，并且异步队列更新

独立触发

v-model 实时渲染

适用于：模板和渲染函数的弹性选择

简单的语法及项目搭建

更快的渲染速度和更小的体积

react

利用jsx创建虚拟dom

是一种在内存中描述dom数状态的数据结构

函数式的方法描述视图

使用虚拟dom作为模板

程序片段

不好控制dom

生命周期

服务端渲染：react的虚拟dom的生成可以在任何支持js的环境生成的，所以可以在node环境生成，直接转为string,然后插入到html文件中输出浏览器便可

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

优点

React伟大之处就在于，提出了Virtual Dom这种新颖的思路，并且这种思路衍生出了React Native，有可能会统一Web/Native开发。在性能方面，由于运用了Virtual Dom技术，Reactjs只在调用setState的时候会更新dom，而且还是先更新Virtual Dom，然后和实际Dom比较，最后再更新实际Dom。这个过程比起Angularjs的bind方式来说，一是更新dom的次数少，二是更新dom的内容少，速度肯定快

ReactJS更关注UI的组件化，和数据的单向更新，提出了FLUX架构的新概念，现在React可以直接用Js ES6语法了，然后通过webpack编译成浏览器兼容的ES5，开发效率上有些优势.

React Native生成的App不是运行在WebView上，而是系统原生的UI，React通过jsx生成系统原生的UI，iOS和Android的React UI组件还是比较相似的，大量代码可以复用

维护UI的状态,Angular 里面使用的是 $scope，在 React 里面使用的是 this.setState。 而 React 的好处在于，它更简单直观。所有的状态改变都只有唯一一个入口 this.setState()，Angular 就比较复杂，不知道背后使用了哪些黑魔法。

同构的JavaScript

单页面JS应用程序的最大缺陷在于对搜索引擎的索引有很大限制。React对此有了解决方案。

React可以在服务器上预渲染应用再发送到客户端。它可以从预渲染的静态内容中恢复一样的记录到动态应用程序中。

因为搜索引擎的爬虫程序依赖的是服务端响应而不是JavaScript的执行，预渲染你的应用有助于搜索引擎优化。

缺点

React是目标是UI组件，通常可以和其它框架组合使用，目前并不适合单独做一个完整的框架。React 即使配上 redux 的组合，也不能称之一个完整的框架，比如你想用Promise化的AJAX？对不起没有，自己找现成的库去。而且第三方组件远远不如Angular多。目前在大的稳定的项目上采用React的，我也就只知道有Yahoo的Email。React本身只是一个V而已，所以如果是大型项目想要一套完整的框架的话，也许还需要引入Redux和route相关的东西。而Angular在这方面提供的东西比React多多了.

angular

优点

AngularJS是一套完整的框架，angular有自带的数据绑定、render渲染、angularUI库,过滤器,directive(模板),服务q(defer),http，inject(依赖注入),factory,provider……，等等一系列工具，基本上只要你在做web开发用过的东西，它都有一个。但是这些东西React自身都没有。

Angularjs的架构清晰，分工明确，扩展性良好，model，view，controller谁在什么时候做什么事情说的很清楚，angular能够让程序员真正专注于业务逻辑，而且因为对html侵入不大，非常易于和designer协作。整个框架充满了DI的思路，耦合性非常低，对象都是被inject的，也就是说每个对象都可以轻易被替换而不影响其他对象。

Angular生产效率高，单向数据流什么的想法非常好，但是写起来太麻烦！我只想变更个很简单的数据还要经过action、dispatcher、reduce、view四步，angular里一行代码就搞定的事情在react里却如此麻烦

缺点

性能

双向数据绑定是一把双刃剑。随着组件增加，项目越来越复杂，双向数据绑定带来性能问题。

双向数据绑定是如何影响性能的？在JavaScript（ES5）中，并没有实现当变量或对象改变时发出通知的功能，Angular的实现方法被叫做“Dirty-checking（脏检查机制）”，通过跟踪数据的改变再动态更新用户界面（UI）。

在Angular的作用域中任何操作的执行都会引发Dirty-checking，随着绑定数量的增加性能就会越低。

Angular 2.0推翻重做使得目前不宜采用此框架

Angular 1.x版本其实是个比较旧的东西了，现在看来有些理念过时了，比如依赖注入、自己独特的模块化，这些东西其实在ES6下已经很好地被解决了。

Angular的2.0几乎是一个推翻重做的框架，估计不会有1.X的upgrade方案。所以如果现在新开始的项目采用Angular的话，会是一个很尴尬的时机。同样，如此大的改动似乎也反面印证了1.X并不是那么好。

学习成本高

使用Angular需要学习大量的概念，包括但不限于：

模块

控制器

指令

作用域

模板

链式函数

过滤器

依赖注入

vue,react 共性：

虚拟dom实现快速渲染

轻量级响应式组件

服务端渲染易于集成路由工具，打包工具及状态管理工具

---------------------

作者：我总是喜欢逆光

来源：CSDN

原文：https://blog.csdn.net/llm913114267/article/details/69068414?utm\_source=copy

版权声明：本文为博主原创文章，转载请附上博文链接！

**Angular、React、Vue.js 等 6 大主流前端框架都有什么优缺点？**

2018年01月02日 16:06:42 [愿你我都实现彼此的马斯洛](https://me.csdn.net/Jack_zengzhen) 阅读数：6738 标签： [前端技术](http://so.csdn.net/so/search/s.do?q=%E5%89%8D%E7%AB%AF%E6%8A%80%E6%9C%AF&t=blog)[前端框架](http://so.csdn.net/so/search/s.do?q=%E5%89%8D%E7%AB%AF%E6%A1%86%E6%9E%B6&t=blog)[vue.js](http://so.csdn.net/so/search/s.do?q=vue.js&t=blog)[angular.js](http://so.csdn.net/so/search/s.do?q=angular.js&t=blog)更多

个人分类： [前端技术](https://blog.csdn.net/Jack_zengzhen/article/category/7367864)

无意中看到的一篇文章，分享一下。后期有更深理解还会补充

链接：[oschina.net/translate/web-frameworks-conclusions](http://oschina.net/translate/web-frameworks-conclusions" \t "_blank)

原文：[sitepen.com/blog/2017/11/10/web-frameworks-conclusions/](http://sitepen.com/blog/2017/11/10/web-frameworks-conclusions/" \t "_blank)

译者：凉凉\_, wilde, 透过树叶的光等

**要不要使用框架？**

如果你是一个有着深厚技术和经验的人，确实可以坦诚的不使用框架。但你团队的其他成员呢？你手下的那些人呢？或者当你的决定把你自己陷入困境的时候呢？

这种情况下，我们将会看到一个不用框架的团队在展开冒险，最后他们会发现自己创建了一个需要自己着手维护的框架。接着就会出现寻找人才的问题，他们不需要知道框架是如何工作的，只需要寻找会调用网络平台 API 的高级技能人才或者一些自称有经验的人才，最后却发现缺少利于团队发展的技能深度和经验。

Angular 2+

优势？

Angular 2+ 的最大优势在于它的流行程度。也有人认为它和 Google 密切相关的名字，会影响团队使用它。Angular 1 的迅速流行是因为那些来自其他交互式应用程序开发环境的人会发现对于开发单页面 web 应用程序具有相似的模型-视图模式。

通过对 Angular 1 进行现代化演变和重新构建框架的某些部分，Angular 2+ 已经真正的爆发了，大量的正式的和非正式培训机构数量都让人印象深刻，开发者有很强的市场竞争力。对于用户来说它有一套用于构建用户界面的丰富组件，这也是本系列中少有的几个框架能够做到这点。

弱点和挑战？

Angular 框架着重于在单个页面应用程序中创建用户界面并没有处理构建完整的 web 应用这个更大的关注点，如果不及早确定下来，这将会导致整个项目难以维护，在实际项目中，运行时提供不属于核心框架的技术往往让人觉得不可思议，这大大降低了 TypeScript 对最终开发者的价值。

未来将何去何从？

Angular 5 刚刚发布，这看来是 Angular 已经成功的印证了快速发布版本的承诺，在 Google 的持续支持下，Angular 会越来越成熟。

像许多的大型组织一样，Google 具有多重（分裂）的人格，从外表上看，Angular 团队和那些专注于浏览器标准的团队之间显得很和谐。但我们的观点是，和谐只是一层薄薄的窗户纸。Angular 团队对于 web 组件和渐进式 web 应用没有一个真正解决方案。我们认为，业界普遍认可的标准将会在 Angular 框架中会逐步实现，这将会影响到如何更好的构建 Angular 应用将成为一个中/长期的风险。

何时选择 Angular 2+

如果你需要在一个大型的框架内获取技术资源，框架内的技术通常很容易移植；或者你需要在框架中训练开发人员，并且还要有一定的信心，他们会在短期内获得一定的开发能力，这样的话你可以考虑 Angular 2+ 。需要注意的是 Angular1（angular.js）与 Angular2+ 是截然不同的，其中的应用、技术和经验不能直接移植到 Angular2+ 的开发中去。

如果你的 web 应用能够很好的转化为标准的模型-视图模式，那么你也可以忽略其他直接考虑使用 Angular2+ 。

如果你对 Google Material UX 设计模式满意，那么 Material Angular 是遵循该模式的一种快速、简单且可靠的方式。

PS：2017下半年Angular市场占有率迅猛增加

React + Redux

**优势？**

React 和 Redux 的最大优势在于它们相对简单和专注。做一件事情并把它做好是非常困难的，但这两个库都很有效地完成了它们的目标。虽然对于某些状态容器方法可能是外部的，但大多数开发人员还是可以轻松掌握概念，并了解单向数据体系结构的好处，简化大量的用户界面应用程序。

**弱点和挑战？**

React 和 Redux 最大的弱点不是它们是什么，而是它们不是什么。要构建一个功能丰富的 Web 应用程序，你需要许多功能，一旦脱离React 和 Redux 和其他一些库的核心，你将发现一个非常分散的社区，拥有无数的解决方案和模式，不容易整合在一起。

 因此，虽然 React 和 Redux 都是非常专注的库，但缺乏经验的团队还是会很容易地生成不可维护的解决方案，而不是意识到他们所做的选择会导致性能不佳或错误。 即使有经验的开发人员也可能意识到，一个松散的架构或惯例可能会在未来困扰他们。

 假省钱是一种对自己的欺骗，组织范围内采用 React 和 Redux 将轻松降低无效率问题。 没有其他库和模式的广泛约定和标准化，标准化React + Redux 比较于我们正在采用的 JavaScript 来编写我们的应用程序效率要高。

**未来将何去何从？**

Facebook 和 React 最近从繁琐的附加专利纠纷中抽离，他们认识到，就像其他项目一样，更广泛的社区能够提高自己的声音。 我觉得这有助于 Facebook 意识到他们还不能更好地了解我们，相信我们来引导项目。 希望这将继续贯穿项目的特点和技术方向。

很难预测 React 和 Redux 的未来。 但是，将库集中在一起，确实会显着提高适应性，大多数React + Redux 模式都会促进一个分离的体系结构，从而可以轻松地进行重构和迭代。 两年前，大家喜欢的还是React + Flux，但整个社区很快就拥抱了Redux。 思维或模式的其他重大转变可能很容易被采纳。 这种关键能力可能会持续到未来。

**何时选择 React + Redux ？**

如果你很少需要手把手指导，并且正在寻找更好的库而不是全面的框架，那么 React + Redux 可能是正确的。 在这一过程中，你不仅需要对你的团队和组织的能力保持诚实，还要在你的初始开发过程中，以及在整个应用程序的长期维护过程中保持诚实。

Vue.js

**优势？**

 渐进式构建能力是 vue.js 最大的优势，vue 有一个简洁而且合理的架构，使得它易于理解和构建。vue 有一个强大的充满激情人群的社区，这为 vue.js 增加了巨大的价值，使得为一个空白项目创建一个综合的解决方案变得十分容易。

**弱点和挑战？**

 在模型-视图应用程序和状态容器类型的应用程序之间的互相转换可能会令人感到困惑，即使没有完美包含一个模式到另一个模式的完美转换，但让人感觉希望能维持两个模式的相关性。对于那些期待 vue.js 完美解决方案，并可能导致难以维护不一致的应用程序的人来说，这至少是令人困惑的。

 一个更大的挑战是 vue.js 依赖于一个单独的人，很明显，其他的项目基本是由一个组织提供支持，但这让人感觉更加有意义，虽然它有一个强大文件的社区和许多有创新的新增项目，但是 vue 核心的开发基本落在一个人身上。

 我们很高兴看到 vue 更加容易接受新兴的标准方法，但是它的类似于 web 组件的模式，而不是真正的 web 组件，这可能是 vue 所得不偿失的地方。

**未来将何去何从？**

虽然 vue.js 有相当广泛的应用，但也很难预测在中期发展中这个势头能持续多久，它不是由一个商业组织直接支持并维护，因此，这很大程度上依赖于维护者的生存能力和继续维护下去的愿望来决定。它也表现出了一定程度的语言适应能力，并且随着某些模式的落伍和失宠而继续保持自身语言的现代化和时代性，目前没有迹象表明 vue.js 架构将来无法适应进一步发展。

**何时选择 Vue.js?**

如果你有一个传统的 web 应用程序，并需要一个强壮稳健的应用程序层，那么 vue.js 可能是一个很好的选择，它有清晰的模式，即使没有经验的团队也能正确或者错误的使用它。尽管 vue UX 框架没有开箱即用的功能，但在 vue.js 上也能大量持续性构建应用，这将有利于你的项目。

Dojo 2

**优势？**

 Dojo2 专注于带来更多构建在状态容器体系之上的动态组件的体验模式，填补了 react+redux 等框架的许多空白。Dojo2 也知道它不单单只是在自己的生态圈发展，通过包含 web 组件导入和导出功能，也意识到需要支持不同的应用实例，但它依旧提供了一个结构化和固有的框架价值，Dojo2 的核心基础仍然是专注于提供交互性。

 Dojo2 觉得它提供了大量重要的功能和解决方案，这对于构建完整的 web 应用是十分重要的，对于其他大多数框架而言这并不是重点。提供一个国际化系统和广泛的易接入性的模式也是其中之一，同时也提供一个主题系统和演进模式，用以确保不仅能为Typescript/JavaScript 提供良好的代码开发，也能像 CSS 那样管理资源。

 Dojo2 专注于提供一个结构化和符合人体工程学的开发环境，通过使用 typescript 和其他开发模式，它试图提供安全的防护机制去引导新手开发人员，通过专注于提高框架开发效率和开发安全性，旨在让开发团队能够快速交付更好的 web 应用程序。

**弱点和挑战？**

有争论的是，通过进一步延长 Dojo2 的发布时间的做法是否是在阻碍框架的发展，反观其他项目由于其资源的扩大能够继续发展和快速迭代，导致 Dojo2 目前明确的处在一个拥挤的竞争环境之中。

这也许是一个潜在的发展机遇和挑战，同时希望能够在灵活性和交互性上而不是别的特殊理由去使用 Dojo2 。

**未来将何去何从？**

Dojo2 将是未来优秀 web 框架之一，它将继续努力为构建可扩展性的 web 应用程序提供清晰的模式和指导。随着新标准的不断出现，Dojo2 将进一步努力去在框架中实现新的标准方法，继续尝试扩大框架的开放性和交互性，创造适合更多人使用的解决方案。

**何时选择 Dojo2?**

如果你想采用一个灵活的、现代的、响应式的 web 应用程序架构，并且你需要很多智能化的默认设置，那么 Dojo2 将是一个不错的选择。不用去拼凑和构建一个管道，并且为你提供更高阶的命令模式让你可以更加专注的开发项目，更加确认它是直接为你可以直接生产开发所准备的。另外，如果你了解 typescript 的优势，Dojo2 会十分严谨的使用 typescript 来管理并提供一个稳健的开发者开发环境。

Ember

**优势？**

Ember.js可能是最固执己见的主流框架，这也是其最大的优势。它有创建Ember.js应用程序的正确方法，通常只有一种方法来创建应用程序。Ember.js更类似于一个产品或平台，在那里你会到一个供应商的长期支持和维护。Ember.js提供了对其平台的全面版本管理，升级工具以及对API升级的强大指导和工具。成熟，是对Ember.js的一个很好的总结。

Ember.js多年来已经证明，它可以保持其框架并使其与现代标准保持一致，同时不会过早遗忘传统浏览器。

 Ember.js有一个清晰合理的架构来全面构建Web应用程序。

**有什么弱点和挑战？**  
Ember.js可能是最固执己见的主流框架，这也是它最大的弱点。虽然社区是开放的并且接受投资，但是仍然需要找到一个正确的方式来摆脱下滑的趋势，这可能是具有挑战性的问题。  
拥有一个丰富的第三方社区也可能具有挑战性。由于没有开箱即用的UX组件，这很可能会让你使用第三方套件。你可能会发现，虽然这些套件并不全面，你将需要建立或寻找其他组件。由于Ember.js没有扩展，所以对如何交互和管理DOM，你会发现你有不一致的部件，而且也没有提供一个易于管理的界面。  
  
**未来该何去何从？**  
Ember.js的主要贡献者是JavaScript语言标准委员会TC39的核心参与者。在过去的几年中，Ember.js对JavaScript的方向比任何其他框架都有更直接的影响。我们的观点是，这将在未来继续受影响，并帮助促进JavaScript的特性和模式。这也意味着Ember.js将继续保持与未来标准的紧密结合的关系。  
Ember.js不可能在将来随时消失，尽管他们的创新很可能是通过与Ember.js紧密结合的其他项目来实现的，比如Glimmer，它为Ember.js应用程序提供了一个新的UI框架，该框架基于TypeScript。  
  
**为什么我会选择Ember.js？**  
如果你在框架中寻找成熟度，那么Ember.js很难出错。另外，由于Ember.js提供的内容被理解，并且有广泛的官方和官方认可的培训，以及严格的结构，找到能够建立基于Ember.js的应用程序的人才可能比其他框架更容易。也可以教大型团队如何构建应用程序，并确保整个团队的共同对话和理解。  
  
如果你想要对社区保持信心，并批判性地思考他们平台的变化，那么Ember.js会是一个很好的考虑因素。您可以花更少的时间跟随当前的技术趋势，并更多地关注创建应用程序。

Aurelia

**优势在哪？**

Aurelia有很多关于构建Web应用程序的方法，结构和想法。 这个框架的编写有很多技术上的优点。

**有什么弱点和挑战？**

我们估计最大的挑战就是核心发展的动力和临界物质的缺乏。我们感觉很多的观点和概念都是我们对其他框架的批评性的想法，但是这些愿望都没有完全交付。它似乎就像是一个正在进行的工作一样，就像Dojo 2，但是它已经是一个已发布的框架。

大部分的Aurelia是坐落在一个人的肩膀上，如果这个人的的注意力或可用性改变，那么将会带来挑战。

**未来会如何？**

对于Aurelia来说，有一个很大的机会。如果它能够实现他的愿景，他将要完整的保存这个构建Web应用程序的已有的模板，但会以更健全、更完整的方式交付。我们不知道Aurelia是否能够充分的利用这次机会。

**为什么我会选择Aurelia？**

如果您致力于Web模型视图应用程序模块，并且你和你的团队试图想把一些事做的更好，那么Aurelia会是一个选择。它就像是一个正在寻求一个更大的社区来帮助它的发展和进化的框架。

最后的思考

真心希望这一系列的帖子至少给了你一点思考，你应该很容易有这样的想法那就是不可能有可验证的正确决定。同时，希望你也意识到没有普遍的错误决定，你应该用一些问题和思考来武装自己，帮助你选择框架。

一个框架仅仅是一些模式的体现，一些科技的集成，源码帮助我们更加容易去构建和维护网站应用，如果你是个体开发者，我们能提供的最好的建议是花费尽可能多的时间使用那些你认为可以为你所用的框架。

如果你是公司的管理者或骨干领导要去做决定，请记住特点列表只是决定的一方面，有时候并不是越多越好。挑战你自己活着你的团队使用一个整体的框架，但是首先，列出对你和你的组织重要的列表，尤其是那些技术之外特点。