# **前端框架Vue、angular、React的优点**区别**和缺点**

****一、Vue.js：****

其实Vue.js不是一个框架，因为它只聚焦视图层，是一个构建数据驱动的Web界面的库。

Vue.js通过简单的API（应用程序编程接口）提供高效的数据绑定和灵活的组件系统。

****Vue.js的特性如下：****

1.轻量级的框架  
2.双向数据绑定  
3.指令  
4.插件化

****优点：****

1. 简单：官方文档很清晰，比 Angular 简单易学。
2. 快速：异步批处理方式更新 DOM。
3. 组合：用解耦的、可复用的组件组合你的应用程序。
4. 紧凑：~18kb min+gzip，且无依赖。
5. 强大：表达式 & 无需声明依赖的可推导属性 (computed properties)。
6. 对模块友好：可以通过 NPM、Bower 或 Duo 安装，不强迫你所有的代码都遵循 Angular 的各种规定，使用场景更加灵活。

****缺点：****

1. 新生儿：Vue.js是一个新的项目，没有angular那么成熟。
2. 影响度不是很大：google了一下，有关于Vue.js多样性或者说丰富性少于其他一些有名的库。
3. 不支持IE8：

****二、angularJS：****

angularJS是一款优秀的前端JS框架，已经被用于Google的多款产品当中。

****angularJS的特性如下：****

1.良好的应用程序结构  
2.双向数据绑定  
3.指令  
4.HTML模板  
5.可嵌入、注入和测试

****优点：****

1. 模板功能强大丰富，自带了极其丰富的angular指令。
2. 是一个比较完善的前端框架，包含服务，模板，数据双向绑定，模块化，路由，过滤器，依赖注入等所有功能；
3. 自定义指令，自定义指令后可以在项目中多次使用。
4. ng模块化比较大胆的引入了Java的一些东西（依赖注入），能够很容易的写出可复用的代码，对于敏捷开发的团队来说非常有帮助。
5. angularjs是互联网巨人谷歌开发，这也意味着他有一个坚实的基础和社区支持。

****缺点：****

1. angular 入门很容易 但深入后概念很多, 学习中较难理解.
2. 文档例子非常少, 官方的文档基本只写了api, 一个例子都没有, 很多时候具体怎么用都是google来的, 或直接问misko,angular的作者.
3. 对IE6/7 兼容不算特别好, 就是可以用jQuery自己手写代码解决一些.
4. 指令的应用的最佳实践教程少, angular其实很灵活, 如果不看一些作者的使用原则,很容易写出 四不像的代码, 例如js中还是像jQuery的思想有很多dom操作.
5. DI 依赖注入 如果代码压缩需要显示声明.

****三、React：****

React主要用于构建UI。你可以在React里传递多种类型的参数，如声明代码，帮助你渲染出UI、也可以是静态的HTML DOM元素、也可以传递动态变量、甚至是可交互的应用组件。

****React特性如下：****

1.声明式设计：React采用声明范式，可以轻松描述应用。  
2.高效：React通过对DOM的模拟，最大限度地减少与DOM的交互。  
3.灵活：React可以与已知的库或框架很好地配合。

****优点：****

1. 速度快：在UI渲染过程中，React通过在虚拟DOM中的微操作来实现对实际DOM的局部更新。
2. 跨浏览器兼容：虚拟DOM帮助我们解决了跨浏览器问题，它为我们提供了标准化的API，甚至在IE8中都是没问题的。
3. 模块化：为你程序编写独立的模块化UI组件，这样当某个或某些组件出现问题是，可以方便地进行隔离。
4. 单向数据流：Flux是一个用于在JavaScript应用中创建单向数据层的[架构](https://link.jianshu.com/?t=http://lib.csdn.net/base/16" \t "https://www.jianshu.com/p/_blank)，它随着React视图库的开发而被Facebook概念化。
5. 同构、纯粹的javascript：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。
6. \*兼容性好：比如使用RequireJS来加载和打包，而Browserify和Webpack适用于构建大型应用。它们使得那些艰难的任务不再让人望而生畏。

****缺点：****

1. React本身只是一个V而已，并不是一个完整的框架，所以如果是大型项目想要一套完整的框架的话，基本都需要加上ReactRouter和Flux才能写大型应用

区别:vue对比其它框架  
Angular  
选择 Vue 而不选择 Angular，有下面几个原因，当然不是对每个人都适合：  
在 API 与设计两方面上 Vue.js 都比 Angular 简单得多，因此可以快速地掌握它的全部特性并投入开发。  
Vue.js 是一个更加灵活开放的解决方案。它允许以希望的方式组织应用程序，而不是任何时候都必须遵循 Angular 制定的规则。它仅仅是一个视图层，所以可以将它嵌入一个现有页面而不一定要做成一个庞大的单页应用。在配合其他库方面它给了更大的的空间，但相应，也需要做更多的架构决策。例如，Vue.js 核心默认不包含路由和 Ajax 功能，并且通常假定在应用中使用了一个模块构建系统。这可能是最重要的区别。  
Angular 使用双向绑定，Vue 也支持双向绑定，不过默认为单向绑定，数据从父组件单向传给子组件。在大型应用中使用单向绑定让数据流易于理解。  
在 Vue.js 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。  
Vue.js 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。Angular，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.js 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列队更新，所有的数据变化都是独立地触发，除非它们之间有明确的依赖关系。唯一需要做的优化是在 v-for 上使用 track-by。  
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。  
React  
React.js 和 Vue.js 确实有一些相似 —— 它们都提供数据驱动、可组合搭建的视图组件。当然它们也有许多不同。  
首先，内部实现本质上不同。React 的渲染建立在 Virtual DOM 上——一种在内存中描述 DOM 树状态的数据结构。当状态发生变化时，React 重新渲染 Virtual DOM，比较计算之后给真实 DOM 打补丁。  
Virtual DOM 提供了一个函数式的方法描述视图，这真的很棒。因为它不使用数据观察机制，每次更新都会重新渲染整个应用，因此从定义上保证了视图与数据的同步。它也开辟了 JavaScript 同构应用的可能性。  
Vue.js 不使用 Virtual DOM 而是使用真实 DOM 作为模板，数据绑定到真实节点。Vue.js 的应用环境必须提供 DOM。但是，相对于常见的误解——Virtual DOM 让 React 比其它的都快， Vue.js 实际上性能比 React 好，而且几乎不用手工优化。而 React，为了最优化的渲染需要处处实现 shouldComponentUpdate 和使用不可变数据结构。  
在 API 方面，React（或 JSX）的一个问题是，渲染函数常常包含大量的逻辑，最终看着更像是程序片断（实际上就是）而不是界面的视觉呈现。对于部分开发者来说，他们可能觉得这是个优点，但对那些像咱一样兼顾设计和开发的人来说，模板能让自己更好地在视觉上思考设计和 CSS。JSX 和 JavaScript 逻辑的混合干扰了自己将代码映射到设计的思维过程。相反，Vue.js 通过在模板中加入一个轻量级的 DSL (指令系统)，换来一个依旧直观的模板，且能将逻辑封装进指令和过滤器中。  
React 的另一个问题是：由于 DOM 更新完全交给 Virtual DOM 管理，当想要自己控制 DOM 时就有点棘手了（虽然理论上可以做到，但是这样做就本质上违背了 React 的设计思想）。如果应用需要特别的自定义 DOM 操作，特别是复杂时间控制的动画，这个限制就很讨厌。在这方面，Vue.js 更灵活，有许多用 Vue.js 制作的 FWA/Awwwards 获奖站点。  
个人推荐就是，vue的入门简单，也有很多公司在用，不愁没人要，react入门难，函数式编程吓人啊。。但是真很好用。。angular不推荐学了。。