链滴

React Vue Web 前端主流技术栈比较

作者：yanjoy

原文链接：https://ld246.com/article/1641802340800

来源网站：链滴

在底层的前端框架领域中，最早是 jquery 称霸互联网，近几年 MVVM 类型的框架慢慢成为主，Vue、React 和 Angular 三大框架并驾齐驱。可以说，目前这三种是开发者用的最多使用最广的底框架，也由此衍生了大量基于这些框架的 I 库。
这些年不断有新的框架冒出来，又不断有旧的框架被淘汰，在这里重点总结一下 Vue、React 各的优缺点以及区别。
<h2 id="前端框架解决了什么问题-">前端框架解决了什么问题？</h2>
框架其实就解决了使用声明语法，描述组件对象的嵌套关系，并自动生成与 DOM 对象的对应关系的问题：

DOM 对象以及他们的从属（同时是传递关系）关系，是通过 html 自动生成的，然而开发者把组件”抽象为 js 对象，由框架实现子组件的自动创建，自动销毁，自动数据传递，自动 render，自事件监听等功能；
解决把 js 组件对象存在它应该在的地方，并且 rerender 的时候能把 js 组件对象和 DOM 节点应起来的过程；
组件复用的问题；
组件重渲染之后，commit 到 DOM 对象上。

在解决 Web 应用开发这些问题时，现代前端框架采用了几种不同的方案：

Vue：是基于数据的 watch 的，组件级别通过 Object.defineProperty 监对象属性的变化，重写数组的 api 监听数组元素的变化，之后进行 dom 的更新。
React：不检查数据变化，React 不会直接渲染数据到 dom，而是中间加一层虚拟 dom，每次都渲染成这个虚拟 dom，然后 diff 下渲染出的虚拟 dom 是否变了，变了的就去更新对应的 dom。所以 React 是需要通过 this.setState()去手动更新数据，它只关心当前组建态。
Angualr：基于脏检查，在每个可能改变数据的逻辑之后都对比下数据是变了，变了的话就去更新 dom。

<h2 id="运行效率问题">运行效率问题</h2>
Vue
vue 是组件级别的数据 watch，当组件内部监听数据变化的地方特别多的时候，一次更新可能计量特别大，计算量大了就可能会导致丢帧，也就是渲染的卡顿。所以当 Vue 组件特别庞大或数据特多时，渲染性能就会明显下降。
解决方案：优化方式就是把大组件拆成小组件，这样每个数据就不会有太多的 watcher。
React
React 并不监听数据的变化，而是渲染出整个虚拟 dom，然后 diff。但是当应用的组件树特别的时候，只是 shouldComponentUpdate 跳过部分组件渲染，依然可能计算量特别大。计算量大了样可能导致渲染的卡顿。
解决方案：通过链表记录下组件路径，通过链表的循环遍历按照时间片分段，让 vdom 的生成再阻塞页面渲染，这就像操作系统对多个进程的分时调度一样。这个通过把组件树改成链表，把 vdom 的生成从递归改循环的功能就是 react fiber。
<h2 id="逻辑-组件复-问题">逻辑、组件复用问题</h2>
Vue
vue 的组件是操作对象的方式，那么逻辑复用方式很自然可以想到通过对象属性的 mixin，vue2 的组件内逻辑复用方案就是 mixin，但是 mixin 很难区分混入的属性、方法的来源，比较乱，代码维性差。
解决方案：通过 mixin 的方式来复用逻辑，也有组件太大的问题，在 vue3 中也采用了 hooks（Vue3 0 function based API）方法。
React
React 的组件是 class 和 function 两种形式，那么类似高阶函数的高阶组件（high order comp nent）的方式就比较自然，也就是组件套组件，在父组件里面执行一部分逻辑，然后渲染子组件。了多加一层组件的 HOC 方式以外，没有逻辑的部分可以直接把那部分 jsx 作为 props 传入另一个组来复用，也就是 render props。但是 HOC 的逻辑复用方式最终导致了组件嵌套太深，而且 class 内生命周期比较多，逻辑都放在一起导致了组件比较大问题。

解决方案：HOC 和 render props 是 react 的 class 组件支持的两种逻辑复用方案。通过 functi
n 组件的 ooks api 解决了 class 组件的逻辑复用方案 Class 组件太大的问题。
<h2 id="-态活跃度的区别">生态活跃度的区别</h2>
| --- | --- |
| <img src="https://ld246.com/images/img-loading.svg" alt="v1.png" data-src="https://b3log
ile.com/file/2022/01/v1-3da2d516.png?imageView2/2/interlace/1/format/jpg"> | <img src="
ttps://ld246.com/images/img-loading.svg" alt="r1.png" data-src="https://b3logfile.com/file/
022/01/r1-6778c44f.png?imageView2/2/interlace/1/format/jpg"> |
从 npmjs.com 上分别看了下，React 和 Vue 之间存在着巨大的安装使用差距，这就意味着世界上大
数团队/开发者在选取前端技术栈时，首先选用的是 React。
| --- |
| <img src="https://ld246.com/images/img-loading.svg" alt="v2.png" data-src="https://b3log
ile.com/file/2022/01/v2-775077bd.png?imageView2/2/interlace/1/format/jpg">|
| Vue 相关库与项目 |
| --- |
| <img src="https://ld246.com/images/img-loading.svg" alt="r2.png" data-src="https://b3log
ile.com/file/2022/01/r2-a7e4c50b.png?imageView2/2/interlace/1/format/jpg">|
| React 相关库与项目 |
这就意味着世界上大多数团队/开发者在选取前端技术栈时，首先选用的是 React。
在知名程序员社区 Github 中搜索两个库可以发现：

Vue 生态支持相对 React 偏弱，最常见的问题就是 React 那边有甚至受欢迎的插件 Vue 没有。R
act 有很多十分受欢迎的开源库，而 Vue 相对匮乏。
Vue 在 TypeScript 语言支持弱，现在虽然 3.x 版本进行了改善，但是由于开发者对于框架的思
惯性原因还是没有多少人认可。

<h2 id="关键特点对比">关键特点对比</h2>
<table>
<thead>
<tr>
<th align="center">非技术性特点</th>
<th>Vue</th>
<th>React</th>
</tr>
</thead>
<tbody>
<tr>
<td align="center">对于初学者的学习曲线*</td>
<td>入门 2-3 天可以尝试开发简单功能；6-8 周独自熟练完成项目需求；对项目框架融会贯通，能
独立解决复杂问题则需要 4-6 个月甚至更长时间</td>
<td>入门 1-2 周可以尝试开发简单功能；6-10 周左右独自熟练完成项目需求；对项目框架融会贯通
能够独立解决复杂问题则需要 4-6 个月甚至更长时间</td>
</tr>
<tr>
<td align="center">库大小</td>
<td>轻巧</td>
<td>较大</td>
</tr>
<tr>
<td align="center">支持团队</td>
<td>个人/社区爱好者</td>
<td>Facebook 公司官方支持</td>
</tr>
<tr>

<td align="center">设计哲学</td>
<td>易用、灵活、高效</td>
<td>声明式、组件化、一次学习随处编写</td>
</tr>
<tr>
<td align="center">适合什么项目*</td>
<td>轻巧，易学且易于编写。由于其熟悉的模板语法和组件的使用，将现有项目集成或迁移到 Vue 得更快，更流畅。因此，Vue 非常适合初创企业，但也可以在大型应用程序中使用。</td>
<td>公司的支持庞大的社区和更大的生态系统，React 非常适合构建复杂的企业级应用程序。因为的成熟性和社区性，或多或少地保证了寻求帮助并获得及时详细的答案</td>
</tr>
<tr>
<td align="center">*对于初学者的学习曲线：“初学者”是指有基本的计算机基础知识，JavaScrip、html、css、浏览器运行原理等 web 开发基础知识扎实。因为个体差异，这里仅为一般情况。对于学者，React 学习曲线相对陡峭，但是一旦入门之后的学习路线两者没有本质差别。另外阿里目前 And Pro V5 版本的 React 脚手架已经大大降低了学习成本。</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
*适合什么项目：Vue 适合小型项目或初创公司这种说法往往来源于目前许多开发者刻板映像，外 Vue 创作者尤雨溪自己也说过：“Vue 从一开始的定位就是尽可能的降低前端开发的门槛，让更的人能够更快地上手开发。 ..... 很多时候我更希望自己做的东西能帮到那些中小型企业和个人开发。举个例子来说，美国传统行业里有很多 small business，它们不像大公司那样有专门的 IT 团队来息化整个流程，很多只能雇一个普通的 contractor 程序员，有些甚至是老板自己兼职研究代码。我到过好几封这样的感谢信，说因为 Vue 让它们多快好省地做了个内部应用，解决了实际问题，这样故事是让我觉得特别爽的。”，在种种原因影响下，Vue 确实不会作为很多公司大型项目优先选择技栈，从而也导致了 Vue 社区并不如 React 繁荣。但是这些并不意味着 Vue 不能拿来构建大型 Web 用。
<h2 id="技术特性对比">技术特性对比</h2>
<table>
<thead>
<tr>
<th align="center">技术特性</th>
<th>Vue</th>
<th>React</th>
</tr>
</thead>
<tbody>
<tr>
<td align="center">超大型 web 应用首屏渲染时间*</td>
<td>较慢</td>
<td>快</td>
</tr>
<tr>
<td align="center">提供了相应式和组件化的视图组件</td>
<td>是</td>
<td>是</td>
</tr>
<tr>
<td align="center">性能优化</td>
<td>更多依赖开发者自身水平</td>
<td>React Fiber</td>

</tr>
<tr>
<td align="center">服务器渲染</td>
<td>支持不好</td>
<td>支持</td>
</tr>
<tr>
<td align="center">数据绑定</td>
<td>双向绑定</td>
<td>单向数据流</td>
</tr>
<tr>
<td align="center">TypeScript 支持</td>
<td>2.x 不支持，3.x 版本改善</td>
<td>支持良好</td>
</tr>
<tr>
<td align="center">组件逻辑复用性</td>
<td>mixin 混入，3.x 版本后改为 hooks</td>
<td>Hooks、HOC</td>
</tr>
</tbody>
</table>
<h2 id="综述">综述</h2>
技术层面从加载速度、运行时性能来说，两者目前综合各种场景应该说是没有什么质的差别，其不存在某项技术更优，只有更适合问题。我们能够根据项目实际情况选择合适的技术栈更为重要。两技术框架其实均能非常好的解决我们日常研发所遇见问题。
React
React 的应用广泛程度、社区活跃度更好，并且 React 是由 Facebook 公司独立团队进行更新护。而且国内很多公司都将 React 作为 web 端应用的主要技术栈，知名的比如：阿里、天猫、华为豆瓣、美团、滴滴、知乎等。这些大厂也开源了一系列基于 React 技术栈的框架，其中最为著名的则 Antd 团队开发的 umi、antd
design。
当我们需要研发大型复杂 web 应用程序时，React 强大的社区、丰富的第三方库能够为我们项提供良好的支持。缺点是 React 的学习曲线可能比较陡峭，编程思维模式较抽象，不利于新人快速掌。
Vue
Vue 从一开始的定位就是尽可能的降低前端开发的门槛，让更多的人能够更快地上手开发，在易性，功能，性能，文档易读上面表现更好。vue3 更是把模版的优势发挥到了极致，在大量数据渲染可以比 React 更快。缺点则是社区规模太小，框架本身也仅仅是由个人开发者在进行维护。在大型复 web 应用开发中，Vue 项目优化比较考验开发者本身技术水平。
最后，使用 Vue 作者尤雨溪自己的总结：
使用场景上来说，React 配合严格的 Flux 架构，适合超大规模多人协作的复杂项目。理论上 Vue 配合类似架构也可以胜任这样的用例，但缺少类似 Flux 这样的官方架构。小快灵的项目上，Vue 和 React 的选择更多是开发风格的偏好。对于需要对 DOM 进行很多自定义操作的项目，Vue 的灵活性优于React。