2018 前端趋势：更一致，更简单

2018-01-20 JavaScript

2017 是一个前端 Web 开发年。

像 React 和 Angular 这样的框架，继续在社区中享有大规模的支持，但是，新的候选者 Vue ，人气也很旺。Webpack 依旧是构建的首选工具，NPM 仍旧是系统选择包的工具。WebAssembly 以前所未有的速度向 Web 开放了众多新的和令人兴奋的案例。像 GraphQL 等技术，革新了书写和在 web 应用中使用 API 的方式。

于此同时，语言自身也在改进，ECMAScript 标准的 2017 版本增加了异步功能，这大大提高了开发者写异步代码时的经验。现在，它们被所有的主流浏览器支持。另一个值得注意的改进是共享内存和原子操作。

然而， 在暴露出他们出现浏览器侧信道攻击涉及推测执行之后，共享内存在2月5日被所有的主流浏览器暂时禁止 。

预计今年某个时候，当浏览器的开发商找到的阻止漏洞的方法时，共享内存就可以使用了。

库和框架

React

2017年9月，React 16 的发布赚足眼球。这是迄今为止，React 动静最大的一个版本：增加了数据块（fragments，现在可以返回一个数组，而不是将所有的东西都装在一个无用的 `<div>` 元素里）；更佳的容错机制（可以显示错误的范围，出错时，React 就会从根元素卸载或者在特殊的出错范围组件处卸载）；接口（portals，现在你可以在 React DOM 树之外的 DOM 节点中展示 React 子元素），还有数据流（streaming，允许服务器端的 App 向客户端提供数据流，而不必等待整个序列完成之后才进行）。

此外，React 还采用 RFC 模式，让 React 开发团队有机会获得更多有益的想法。任何会影响到 React API 的 RFC 建议，都可以提交。React 开发团队发布了他们的语义修改（context changes）建议作为第一个 RFC 的示范。读起来还挺有意思。

React 粉们已经提供好几个建议，有些功能非常有趣，包括：

- 处理方法的参数——减少代码量，这个建议中，props，state 和 context 都被视为参数。
- setState 返回一个承诺（promise）——如果你需要 setState 同步，并且你在一个异步/等待的环境中，你会发现这对形影不离的鸳鸯对子非常美好。
- 异步-安全静态生命周期钩子——完全抛弃传统的、基于类的 API ，让我们处理起异步数据来更容易，还能节省不必要的处理步骤，向方法组件提供更洁净的升级通道。

当然，并不是所有的建议都会出现在未来的版本中。但要承认，React 开发团队为用户们做了这些安排，还是很不错的。随着 Yarn 和 Ember 等项目的应用展开，RFC 将会变成主流形式。

现代网络开发过程中，设置并协调所有工具相当复杂，所以，Boilerplate 项目在 React 社区内总是受到欢迎。大多数人会建议用户直接克隆项目文件，就地起炉灶。新手常常茫然不知所措。因为，他们总是会看到一个复杂的“白板”（blank slate），竟然会依赖成千上万个类库或软件，而且他们完全不理解那些配置代码是什么意思。

Facebook 的 create-react-app 则不同 —— 它是一个命令行工具，可以将 Webpack、Babel、PostCSS 和 Jest 打包到一起，在零配置情况下的进行开发。自去年以来，它越来越受欢迎。它在 GitHub 中，是一颗闪亮的明星，star 数由 2017 年初的 18k 直接攀升到年底的 40k 。它还提供一个“eject”（弹射）命令，让你跳出 create-react-app 模式。那个模式下，依赖软件自动安装、配置文件自动生成，你只需要手动修改配置文件。有人说，这个命令的面世也是 React 近年来大受欢迎的部分原因。

对于服务器端的 React 应用程序，next.js 是个很流行的选择。它提供了你所需要的“通用的”（universal）网络应用开发工具，安装、配置起来还挺简单。在开发难度日益增加，渐进迭代式网络应用（progressive web app）再度受宠，通用的或者同构（isomorphic）的应用降温的情况下，这一点尤其重要。如果你要新开发一个项目，我郑重地推荐你使用 next.js 。

我认为，React 社区最终会开发出类似 create-react-app 的东西，但针对的是更为复杂的应用。

next.js 与此目标非常接近。但它只是服务器端的应用，这就意味着它不会成为主流。在我看来，还没有哪一个框架已经同时实现即好开发，又好使用。

> “附带电池”（batteries included）的方法将诱惑越来越多的开发者，从而对系统配置的复杂和系统维护所必须花费的时间产生错觉。

Angular

尽管 Angular 最新的版本（版本 5.1.3 ）已于1月3号发布了，但是 AngularJS 项目（也就是 Angular 1.x 版本 ）仍旧处于活跃的开发状态，甚至在 2017年12月18号 发布了版本 1.6.8 。许多大公司仍旧使用旧版本的 Angular ，并由于这个原因重要的速度改进和安全修复都移植到了 AngularJS 上。

尽管谷歌对就项目的支持何时结束还不明确，但是在过去的官方说法中已表明对其的支持，在主要的 web 流量转向 angular.io 而非 angular.org 之前是不会停止的。然而，鉴于旧版本使用的是相当自由的 MIT 协议，尽管官方在2018年不会对其在继续支持，你也可以期待进一步的发展。

近来 Angular 的发布引起了大家的注意，尤其是最新的 v5 版本的发布。通过如对模板的提前（ahead-of-time）编译，以及在打包中简单方便地整合 service worker 这样创新性的功能，其保持着与其竞争者的与众不同。

当这些功能对于任何应用程序都是必备的时候，Angular 的闪光之处在于其集成的工具。Angular CLI 简单易用，并且现在还可以通过 App Shell 提高对快速生成通用的和渐进的 web 应用的支持。

React 社区所秉持的是一种不太固执己见的前端开发哲学。大多数情况下，开发者需要手动安装许多复杂的功能，除非他们使用 multitude of boilerplate projects 项目中的一种。许多开发者倾向于自己动手设置，这样他们可以理解系统的各个方面。

有时 web 社区感觉起来是在固执己见和集中化与非固执己见和非集中化之间的轮回。一件令人不禁思考的事情是 React 社区是否会最终向其他的方向发展。

在完成了几个大型定义开发的 React / Redux / Webpack 项目后，所有的事情都基本为你准备好了，“马上开始工作”（just work）是一种极具吸引力的前景。

通过近来发布的版本，可以有趣的看到 Angular 在新的一年中竟会更加受到欢迎。尽管还很难说有多少，但是当你看到 NPM 的下载量的时候，Angular 并没有看起来增长的那么多。React 已经继续保持领先，尤其是在过去的一年中。它目前每天 NPM 的下载量是其他的三倍。

Vue

Vue 在 2017 年已经成了 React 一个非常受欢迎的可替代选项。它们都利用了虚拟 DOM ，并且都是基于组件且超轻量级的。在 JavaScript 2017 调查的描述中，Vue 被列为Angular 1 和 React 之后第三个最常被使用的前端框架。最值得注意的是它还是那次调查中最“想要去学习”的框架。

Vue 的核心团队计划 2.6 版本的发布会赶在今年的2月份之前，并将专注于错误处理、函数式组件一级服务端渲染。跟随 React 的引领，他们也计划在未来的版本中只支持那些基业长青的浏览器版本。

> Vue 在过去几年 日渐受欢迎, 但要取代 React 当前前端视图库王者的地位，现在看来还很难说。 许多人都写过它对于来自 Angular 领域的开发者们的吸引力, 而我也期望这种吸引力能继续保持。通常的观点是，Vue 不需要你去使用 JSX ，也不像 Angular，它不会强制要求你使用 TypeScript。

它的模板语言也同 Angular 的相当类似。此外，Vue 也有一整套类似 Angular 的联系紧密的包，不过 Vue 在以一种更加分散的方式将它们维护得相当好。

模块打包器

Webpack

Webpack 3 在 2017 年 6 月发布，将作用域的提升（scope hoisting）作为它的旗舰功能。作用域的提升（scope hoisting）将所有模块一同封装在一单个闭包中而不是分拆它们。这可以显著地提升 bundle 的执行时间和 bundle 的体积。 Rollup 是一个显著的特性，另一个捆绑器模块已经成为 Webpack 2 及更高版本中功能的灵感来源。

Webpack 团队已为 Webpack v4 版计划了许多重要的特征，这是为 alpha 版本写的博文，预计将会很快发布。最大的特点是 WebAssembly 模块的支持--目标是使 WASM 模块作为 ECMAScript 模块轻易地运行在 Webpack 上。还计划在生成 CSS 的方式彻底修改 WebPack 。而不是把 CSS 植入 JavaScript 中，Webpack 4 将生成 CSS 资源。

新版本还将专注于构建效率（性能）-- 这是 Webpack 社区投票选出的最优先的 issue 。

在我看来，Webpack 也应该更多地关注文档和配置信息。虽然 Webpack 的过人之处是配置灵活，但它牺牲了用户体验。

一个 Webpack 的 zero-config（零配置）模式已被提出，但它并没有被优先考虑，尽管像 Parcel 这样的模块打包器已经爆炸式地流行。

Parcel

2017 年底，Parcel 大出风头，在不到一个月的时间里斩获 1.4万 多个 star。它的成功，得益于 Webpack 提供的“零配置”的进展缓慢和混沌不清。它提供了几个重要的、跟 Webpack 类似的模块绑定功能，如代码分割和模块热替换。

接下来的开发工作将会集中在补充与 Webpack 类似的小功能上，如进入点（entry point）和一个完备的插件系统。

2018 年我将会密切关注 Parcel 的开发进展。它是否能取代炙手可热的 Webpack ，让我们拭目以待。

> 尽管 Webpack 的最新版本推出了很有价值的功能，新版的用户文档网站也进行了大幅的改进，还是让人感觉到 Webpack 正在走下坡路。

在复杂应用情景下，Webpack 的配置工作仍然是一件头疼的事。

如果能纾解开发人员的痛苦，提供一个不需要多少配置工作的替代方案，Parcel 定会有所成就。

其他工具

Gulp 和 Browserify 仍然被数以千计的项目以各种形式采用，但不再被认为是前端构建工具的前沿技术。它们的持续开发对于现有系统的维护非常重要，并且它们目前仍然可以用于非常具体的新项目用例。然而，过去几年开发者的普遍看法是，它们过于复杂，需要过多的手动设置。在 Webpack 应用越来越广泛占据领先地位的情况下，他们去年的 NPM 下载量都在持续下滑。

工具

TypeScript

TypeScript 有一个版本计划在一月发布，包括新的 ECMAScript 功能，例如数字隔离器和几种涉及对象的文字和类的高级类型系统改进。还有一个改变计划，是提高 TypeScript 的模块系统处理非 ECMAScript 模块的能力。

这将使它更符合 Babel 处理模块互操作性的方式。希望这可以让 TypeScript 更容易使用不同类型的模块，毕竟对新用户来说是一个致命的痛点。此版本还计划通过增加对 ECMAScript 模块自动转换的支持，来改进已经非常棒的重构功能。

> 微软的 TypeScript 显然在对抗 Flow 上已经赢了（对手是来自 Facebook 的类型检查工具）。这有很多原因，但在我看来，仅仅是微软把项目运作得很好。

跟微软每个月的大量的版本发布相比，Flow 就是零星的小的版本。而且使用 TypeScript 的工具也更好，带有 tslint 的卓越的 linter 支持和 Visual Studio Code（以及许多其他编辑器）提供的绝妙的编辑器支持，提供了 Flow 不可能实现的自动转换。

这跟是否是一个更好的类型系统几乎是无关的。——我敢打赌，大多数开发人员更关心的是支持和易用性。

此外，TypeScript 的社区是很大的。通过 DefinitelyType 项目，TypeScript 提供的流行 NPM 包的类型定义与 flow-typed 提供的类型定义相比，要多很多。如果不出意外，这一事实对任何使用 Flow 的项目的长期生存能力构成严重威胁。

移动端

通用 Web 应用程序在 React 出现的时候开始流行起来。这种创新使前端 Web 应用程序能够以增加开发复杂性为代价在服务器上先渲染。虽然它们还很是很流行，但它们绝不是真正的做事方式。

在移动端，当前的开发者已经开始专注于开发所谓的渐进式 Web 应用 - 这是最初由 Google 赞助的一项计划，旨在使 Web 应用对移动端用户更加友好。对于开发者来说，这意味着更加关注速度和移动端用户体验。这可以通过使用像 service workers 来实现离线支持和应用程序清单文件来定制应用在操作系统中的外观等新技术来实现。这可以被看作是响应式网页设计的自然演变。

Google 还赞助了加速移动端页面（Accelerated Mobile Pages，AMP）项目，该项目通过标准化由 Google 提供的缓存式 Web Components 轻量级文档格式来极大地增加了移动设备上的网页加载次数。它已经被网络上的主流内容发布商迅速采用，但关于发布商的广告收入和关于通过在 Google 服务器上托管内容而放弃控制权的担忧这两方面存在持续的争议。

如果我们希望 Web 继续保持为一个充满竞争和吸引力的平台，我们需要与移动端应用竞争。

> 尽管 渐进式 Web 应用不能做移动端应用可以做的所有事情，但它是维持 Web 长期健康状况的重要一步。我希望他们变得更受欢迎，最好在不久的将来成为强制性的。

概括总结

总的来说，前端已趋于将现有项目和 Web 开发中许多不同的部分进行整合。React、webpack、TypeScript 继续变得更受欢迎。Vue 和 Parcel 看起来可能成为各自的领域的领先者的竞争威胁；同时，旧的技术如 Angular 和 Browserify 还在，但以开始缓慢下滑。

一些趋势仍在继续，如基于组件的设计。它绝不是一个新概念，它最近开始复兴并不局限于 Web 开发。我不希望应用程序架构在短期内发生任何根本性的变化。

有一种倾向于开发者友好的“自以为是”的工具。你可以在反对 Webpack 和 React 的生态系统的复杂性上看到它们。简单的确胜过复杂，但是没有复杂度很难满足各种各样的需求。

> 前端发展需要的是更多的共识。人们常常嘲笑它过于复杂，我也有这样的观点。

最近的一个重点是吸引新的开发人员，我认为我们也应该关注一般企业 Web 项目中的复杂性——包括应用程序本身和辅助它的构建工具。

插件: LogRocket, 一款适合 Web 应用的 DVR

LogRocket 是一个前端日志工具，它可以让你像发生在自己的浏览器中那样重现问题。无需猜测错误发生的原因，或者要求用户截图以及日志转储，LogRocket 可以让你重现会话以便快速了解发生了什么错误。无需考虑框架，它适用于任何应用程序，也有插件可以从 Redux、Vuex和@ngrx/tore 上记录额外的上下文。

除了记录 Redux 动作和状态之外，LogRocket 还会记录控制台日志、JavaScript 错误、堆栈信息、带有头+主体的网络请求/响应、浏览器元数据和自定义日志。它还可以指导 DOM 记录页面上的 HTML 和 CSS ，即使是最复杂的单页面应用程序也可以重建像素完美的视频。

译者：Tocy, 大别阿郎, Tot_ziens, 南宫冰郁, 亚林瓜子, 无若, LeoXu, 我是菜鸟我骄傲, wilde
译文：https://www.oschina.net/translate/what-im-looking-for-from-frontend-in-2018