

在 2016 年学 JavaScript 是一种什么样的体验？

程序猿 2016-10-07

来自：方应杭的文章 - 知乎专栏

链接：zhuanlan.zhihu.com/p/22782487（点击尾部阅读原文前往）

原文：<https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f>

嘿，我最近接到一个 Web 项目，不过老实说，我这两年没怎么接触 Web 编程，听说 Web 技术已经发生了一些变化。听说你是这里对新技术最了解的 Web 开发工程师？

准确地说，我是一名「前端工程师」。不过你算是找对人了。我对今年的技术别提多熟了，前端可视化、音乐播放器、能踢足球的无人机，你尽管问吧。我刚去 JS 大会和 React 大会逛了一圈，没有什么新技术是我不知道的。

厉害。是这样的，我要开发一个网页，用来展示用户的最新动态。我想我应该通过后端接口获取数据，然后用一个 table 来展示数据，用户可以对数据进行排序。如果服务器上的数据变化了，我还需要更新这个 table。我的思路是用 jQuery 来做。

可别用 jQuery！现在哪还有人用 jQuery。现在是 2016 年了，你绝对应该用 React。

哦，好吧，React 是什么？

React 是一个非常厉害的库，Facebook 的牛人写的。它能让页面更可控，性能极高，而且使用起来很简单。

听起来确实不错。我能用 React 展示服务器传来的数据吗？

当然可以，你只需要添加两个依赖，一个是 React，一个是 React DOM

额，等下，为什么是两个库？

React 是我说的库，React DOM 是用来操作 DOM 的。因为这些 DOM 是用 JSX 写的，所以需要有一个专门的库来操作。

JSX？JSX 是什么？

JSX 是对 JS 的扩展，它看起来跟 XML 差不多，可以用来写 HTML，你可以认为 JSX 是一种更优雅的 HTML 写法。

为什么不用 HTML 了.....？

现在可是 2016 年啊，没有直接写 HTML 的。

对哦。好吧，加了这两个依赖，是不是就可以开始用 React 了？

不行哦。你需要添加 Babel，然后才能用 React。

Babel 是另一个库？

嗯，Babel 是一个转译工具，Babel 能把你写的 JS 转译成任意版本的 JS。你不一定非要用 Babel，但是如果你不用的话，你就只能写 ES5 的语法了。你知道的，现在是 2016 年，你怎么能不使用 ES2016+ 的语法呢？ES2016+ 多么酷啊。

ES5 是啥？ES2016+ 又是啥？我有点晕。

ES5 就是 ECMAScript 5。大部分人都会使用 ES5，因为大部分浏览器都支持 ES5。

ECMAScript 是啥.....

你晓得的，JS是1995年诞生的，而JS的标准是1999制定出来的。那时候 JavaScript 还叫做 Livescript，只能运行在网景的浏览器里。那时真是混乱的年代，现在好了，我们有了 JS 的 7 个版本的规范。

7 个版本？那 ES5 和 ES2016+ 是？

分别是第 5 个版本和第 7 个版本。

诶，那第六个版本呢？

你说的是 ES6。每个版本都是上一个版本的超集，所以你直接使用最新的 ES2016+ 就好了。

对哦。为什么不用 ES6 呢？

好吧，你可以用 ES6，但是你就用不到 `async` 和 `await` 这么酷的语法了。用 ES2016+ 比较好。用 ES6 的话你就只能用 `generator` 来控制异步任务流了。

不知道你在说什么.....你说了太多我听不懂的名词了。我只是想从服务器取点数据，我以前用 jQuery 挺好的，从 CDN 引入 jQuery，我就能用 AJAX 获取数据了，现在不能这样做吗？

大哥，都 2016 年了，没人用 jQuery 好吗。所有人都知道用 jQuery 只会造出「意大利面条」一样的代码（不可维护）

好吧，所以我现在要加载三个库才能获取并展示数据。

对的，其实你可以用「模块管理器」把这三个库「打包」成一个文件。

哦，什么是模块管理器.....

不同平台的模块管理器不同啦。前端的模块管理器一般指管理 AMD 或者 CommonJS 模块的东西。

好.....吧，什么是 AMD 和 CommonJS?

是两个定义。我们有很多方式来描述 JS 中多个库或类的交互方式，比如 exports 和 requires。你可以按照 AMD 或者 CommonJS 的 API 来书写 JS，然后用 Browserify 将它们打包。

听起来很有道理。不过，什么是 Browserify?

是一个工具，用来将 CommonJS 形式的 JS 文件打包起来，放到浏览器里运行。用 npm 仓库的人发明了 CommonJS。

npm 仓库是什么.....

是一个公开的仓库，用于放置可依赖的模块。

就像一个 CDN 么?

不太一样。它更像是一个数据库，每个人都能在上面发布代码，也能下载上面的代码。你可以在开发的时候将这些代码下载到本地来使用，必要的时候也能上传到 CDN。

听起来像是 Bower!

是的，不过现在是 2016 年了，没有人用 Bower 了.....

好吧，我知道了，所以我应该用 npm 来安装依赖。

对的。我举个例子吧，如果你要使用 React，你直接用 npm 安装 React，然后在代码里导入 React 就可以了。大部分 JS 库都能这么安装。

嗯，Angular 也可以。

Angular 是 2015 年的事情了。不过今年 Angular 还没死，还有 VueJS 和 RxJS 等等，你想学一学么？

还是用 React 吧。我刚才已经学了够多东西了。所以我用 npm 安装 React 然后用 Browerify 来打包就好了？

是的。

这么做看起来有点过于复杂啊。

确实。这就是为什么你应该使用 Grunt、Gulp 或者 Broccoli 这样的任务管理工具，它们能自动运行 Browserify。不对，你现在可以用 Mimosa。

你在说什么.....

任务管理工具。不过我们现在已经不用了。去年我们还在用，后来改成了 Makefiles，但是现在我们用的都是 Webpack。

我以为只有 C/C++ 项目才会用 Makefiles。

是的，不过显然我们做 Web 开发的，喜欢先把事情搞复杂，然后回归到最朴素的状态。每年我们都是这么搞的。你就看着吧，过不了两年，我们就可以在网页上写汇编了。

唉，你刚才说的 Webpack 是什么？

另一种模块管理工具，同时也是一个任务管理工具。你可以认为它是 Browserify 的加强版。

哦，好吧，为什么 Webpack 是加强版？

额，可能并没有加强吧。Webpack 告诉你应该如何管理你的依赖，Webpack 允许你使用不同的模块管理器，不只是 CommonJS，甚至支持 ES6 模块。

这都是哪跟哪啊，我都被绕晕了。

大家都被绕晕了，不过等 SystemJS 出来了就好了。

天呐，又一个 JS 库，这是什么鬼？

呵呵，不像 Browserify 和 Webpack 1.x，SystemJS 是一个动态的模块加载器。

等下，刚才不是说应该把所有依赖打包成一个文件吗？

话是这么说，但是等 HTTP/2 普及之后，不打包反而更好。

那为什么我们不直接在页面里添加 React 的三个依赖文件呢？

不行。你可以从 CDN 加载这些文件，但是你还想要在本地图用 Babel 转译。

唉，这么糙？

是的，你不能在生产环境上运行 babel，你应该在发布到生产环境之前，运行一系列的任务，包括压缩、混淆、内联化CSS、延迟加载script.....

我懂了我懂了。既然我不能直接用 CDN，那么我应该怎么做？

我会考虑用 Webpack + SystemJS + Babel 来转译 Typescript。

Typescript？我们不是在说 JavaScript 吗？！

Typescript 也是 JavaScript 呀，它比 JS 更好用，是 JS 的超集，它是基于 ES6 的，就是我们刚才谈论的 ES6，你还记得吧。

ES2016+ 已经是 ES6 的超集了，怎么又冒出来一个 Typescript？

是这样的，Typescript 能让我们写出「强类型」的 JS，从而减少运行时的错误。2016年，我们应该让 JS 支持强类型了。

显然 Typescript 可以做到。

Flow 也可以做到，区别是 Typescript 需要编译，而 Flow 只是检查语法。

唉，Flow 是？

是一个静态类型检查器，就是 Facebook 的人写的。使用 OCaml 写的，函数式编程很叼的。

OCaml？函数式编程？

如今大牛都用这些东西，都2016年了，你懂的，函数式编程、高阶函数、柯里化、纯函数这些概念。

不知道你在说什么。

一开始大家都不知道。这么说吧，你只需要知道函数式编程比面向对象编程厉害，2016 年我们就指着函数式编程了。

等下，我大学里学过面向对象编程，当时我觉得它还不错。

Java 在被 Oracle 买下来之前也挺不错啊。我的意思是，面向对象以前是不错，现在依然有人用它，但是现在所有人都发觉状态变换是很难维护的，所以大家都开始用「不可变对象」和函数式编程了。Haskell 的人已经用这套东西用了很久了，不过幸运的是 Web 开发领域里有 Ramda 这样的库，让我们用 JS 就可以进行函数式编程了。

你刚刚是不是又抛出了几个名词？Ramnda 又是什么？

不是 Ramnda，是 Ramda，跟 Lambda 表达式有点像。是 David Chambers 写的库。

谁？

David Chambers，大神一个。blablabla

我不得不打断你一下了。这些东西看起来都不错，但是我觉得它们都太复杂，而且没必要。我只是想获取数据然后展示，我很确定这种情况下我不需要掌握这些知识。

回到 React 吧，用 React 我怎么从服务器获取数据？

额，React 没有提供这个功能，你只能用 React 展示数据。

服了啊。那我怎么获取数据？

你用 Fetch API 就可以了。

啥玩意？这个 API 的名字很烂啊。

我也觉得是啊。Fetch API 是浏览器提供的异步请求接口。

哦，那不就是 AJAX。

AJAX 只是使用 XMLHttpRequest 对象，但是 Fetch API 可以让你用 Promise 风格来发起异步请求，帮你摆脱「回调地狱」。

回调地狱？

是的，每次你发起一个异步请求，就得等待它响应。这时你就得在函数里使用一个函数，这种嵌套调用就是回调地狱。

好吧。Promise 解决了这个问题么？

是的。用 Promise 来管理回调，你就可以写出更易读的代码，更容易测试的代码。甚至可以同时发起多个请求，然后等待它们全部返回。

Fetch 也能做到吗？

是的。但前提是你的用户使用了新版的浏览器，不然的话你就需要加一个 Fetch 的「polyfill」，或者使用 Request、Bluebird 或者 Axios 这些库。

天呐我到底需要多少个库？

这是 JS，同一件事情有上千个库在做。我们了解库，而且我们有最好的库，我们有海量的库，

要什么有什么。

你刚才说的几个库都是干什么的？

这几个库操作 XMLHttpRequest 然后返回 Promise 对象。

好像 jQuery 的 ajax 方法做的是同样的事吧.....

从 2016 年起我们就不用 jQuery 了。用 Fetch，大不了加个 Polyfill，要不然用 Bluebird、Request 或者 Axios 都行。然后用 await 和 async 管理 Promise，这样才能控制好异步任务。

这是你第三次说 await 了，那是什么东西？

await 能让你拦住一个异步调用，让你更好地控制异步返回的数据，大大增强了代码的可读性。await 非常好用，你只需要在 Babel 里添加 stage-3 配置，或者添加 syntax-async-functions 和 transform-async-to-generator 插件就可以了。

听起来像是疯了。

没疯。为了使用 await，把 Typescript 编译之后再 用 Babel 转译一道的人才是疯了。

啥玩意？Typescript 不支持 await？

下个版本就支持了。

我已经无话可说了。

你看其实很简单。用 Typescript 写代码，用 Fetch 发起异步请求，所有代码编译成 ES6，然后用上 Babel 的 stage-3 配置项，把 ES6 转译成 ES5。所有代码用 SystemJS 加载。如果你用

不了 Fetch，就加个 polyfill，或者用 Bluebird、Request 或者 Axios，这样你就可以用 await 来处理 Promise 了。

看了我们对于「简单」的理解是不同的。好吧，有了这些，我终于可以获取数据然后用 React 展示数据了，对吧？

你的网页需要处理状态变更吗？

唔，不用吧。我只是想展示数据。

那就好，不然我就得跟你解释 Flux，以及 Flux 的一些实现，比如 Flummox、Alt、Fluxible。不过说真的你应该用 Redux。

你说的这些我就当耳旁风了。再说一次，我只想展示数据。

这样啊，如果你只是想展示数据，其实你不需要 React。你只需要一个模板引擎。

你逗我呢？

我只是告诉你你可以用什么技术。

别说了，真的。

我想说，即使只是用一个模板引擎，我还是会用 Typescript + SystemJS + Babel 的。

我只是想在页面上展示数据，你就告诉我用哪个模板引擎就好了。

有很多，你用过哪一个？

额，太久没用了，不记得了。

jTemplates、jQote 还是 PURE?

额，不记得，还有别的么？

Transparency? JSRender? MarkupJS? KnockoutJS? 这一个支持双向绑定。

还有吗？

PlatesJS? jQuery-tmpl? Handlebars? 还有些人在用。

有点像。有哪些跟最后一个比较像的？

Mustache, underscore? 我记得连 Lodash 都有一个模板引擎，不过这是 2014 年的事情了。

额，也许是再新一点的库？

Jade? DustJS?

没用过

DotJS? EJS?

没用过。

Nunjucks? ECT?

没用过。记不起来了，要是你的话，你用哪个？

我应该会用 ES6 原生的模板字符串

我猜猜，只有 ES6 支持。

对的。

需要用 Babel

对的。

需要用 npm 安装

对的。

需要用 Browserify 或者 Webpack，或者 SystemJS

对的。

如果没用 Webpack 的话，我还需要一个任务管理工具。

对的。

但是由于我要用函数式编程和强类型语言，所以我首先要用上 Typescript 或者 Flow。

对的。

如果我要用 await，那我就必须用 Babel 转译。

对的。

然后我就能用上 Fetch、Promise 和各种炫酷的东西。

嗯，别忘了加上 Fetch 的 Polyfill，因为 Safari 不支持 Fetch。

你猜怎么着，我们就聊到这吧。我不做了，我不做 Web 了，我也不想再碰 JS 了。

没事，过不了几年，我们都会用 Elm 或者 WebAssembly 了。

我要回后端去了，我受不这些变动、版本更新、编译和转译了，JS 社区如果觉得有人能跟上它的脚步，那这个社区就是疯了。

我理解你。我建议你去 Python 社区。

为什么？

听说过 Python 3 吗？

完。

译者注：最后一句「听说过 Python 3 吗？」是讽刺 Python 3 发布已经 8 年了，Python 社区却依然在使用 Python 2.7。而 JS 社区正好相反，把还没有实现的语言特性都用到生成环境中了！

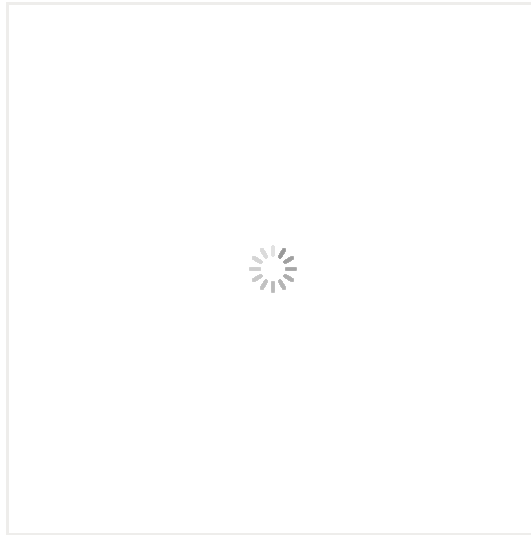
●本文编号1976，以后想阅读这篇文章直接输入1976即可。

●本文分类“[前端](#)”，搜索分类名可以获得相关文章。

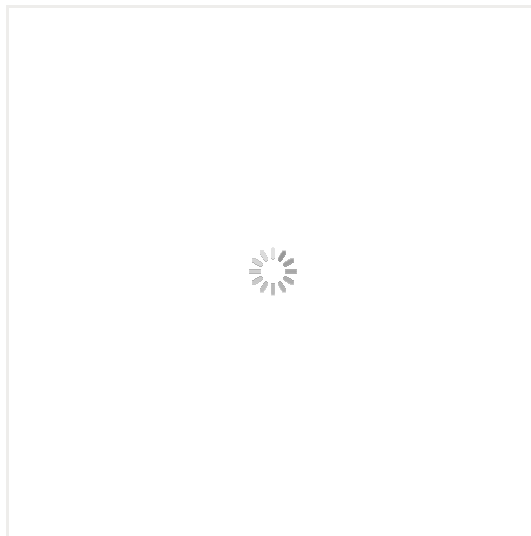
●输入m可以获取到文章目录

本文内容的相关公众号推荐

前端开发↓↓↓



Web开发↓↓↓



更多推荐《**15个技术类公众微信**》

涵盖：程序人生、算法与数据结构、黑客技术与网络安全、大数据技术、前端开发、Java、Python、Web开发、安卓开发、iOS开发、C/C++、.NET、Linux、数据库、运维等。

[阅读原文](#)