北雁云依的博客
把信拿去吧，你可以假戏真做



2022年12月22日 · 4065 字 · 27 分钟 · 1224 浏览

# Hexo 的表演该落幕了，让它退场吧

教程　杂谈

我在 2019 年开始用 Hexo 写博客。在那之前我用的是 jekyll 和 WordPress。出于身份隔离的需要，我有好几个博客，这个博客的域名是 `stblog.penclub.club`，其中的 `st` 是 `static` 的意思，与 `dynamic` 相对应。那时我没有接触前端技术（事实上还不会编程），在我当时的概念里，网页分为动态网页和静态网页，动态网页需要一个能运行 PHP 之类东西的服务端，静态网页则只需要一个静态文件服务器。

我印象里的 Hexo 是一个使用 ejs 模板生成静态页面的工具。虽然不会编程，但改代码我还是会一点的，于是我修改了我博客的代码。后来我看到了 hexo-theme-icarus，它是我最喜欢的 hexo 主题。然而当我准备如法炮制，给它更换评论系统时，我却发现自己改不动它：它用的是一个被称为 jsx 的全新格式。[1]我转而使用 hexo-theme-volantis，它用的是 ejs，不过它的功能够多，不需要我去魔改。

2020 年底，我下定决心要学网页编程，并且我做到了。我用 1 个月时间从 fullstackopen 学会了用 React 写单页应用（Single Page Applications，SPA，下文会用到这个缩写），之后又花一个学期，学会了 TypeScript 和后端开发。现在我知道，html 里可以没有任何网页上的内容，而只包含几行用于加载 js 的代码，内容则完全由 js 渲染上去。这时我回过头看之前的 `hexo-theme-icarus`，发现也不过如此：我用 React 是写函数式组件的，而它用的是类组件的老式写法。我只写过一两个 JavaScript 项目，之后用的都是 TypeScript，而 `hexo-theme-icarus` 用的仍然是 JavaScript（估计是 `hexo-`

`renderer-inferno` 只支持 js）。其它用 ejs 作为模板引擎的 Hexo 主题则更加落后：论语法简洁度，论类型提示，ejs 都远远不如 jsx。

仅仅半年的时间，我拿来写博客的东西对我就不再神秘，甚至显得有些腐朽。前端领域在近几年的发展实际上也如此，只是稍微慢一点。2013 年，和 Hexo 同龄的 gulp 出现，两年后 Webpack 诞生，**前端工程化**如火如荼地进行，只用了几年，gulp 就已无人问津。前端开发者们的构建工具从像 gulp 那样的自动化脚本，变成了 Webpack、Parcel 和 Rollup 那样的打包器，还引入了像 Babel、SWC 和 ESBuild 那样的**编译器**。

然而，Hexo 和它的主题们显然没有跟上这一进度。绝大多数 Hexo 主题还在用 `<script>` 标签引入以 UMD / IIFE 形式提供的前端库（比如 jQuery），少部分主题加上了 gulp 以实现代码压缩。ES Module、Tree-Shaking 这样的新概念则是闻所未闻，这为它们带来了**极其严重的空间浪费**。ejs 模板缺乏类型提示和语法检查，这导致开发者的心智负担停留在**五年前**前端开发者们的水平。此前我使用的 `hexo-theme-volantis` 主题就在一处少判断了一个可能为空的值，让我的分享图标显示不正常，这个 bug 持续存在了至少两年，直到我写这篇文章时才被我修复，而假如有类型提示，这种错误就不可能发生。这实际上还反映了 yaml/json 配置文件的缺陷，它们的类型提示（JSON Schema）和 TypeScript 的类型提示是两套系统，你需要一些步骤才能把它们对接起来。（尽管绝大多数 Hexo 主题根本没使用 TypeScript）

`hexo-theme-icarus` 在首屏会加载大约 400k 的 CSS（未经 gzip 压缩的体积，下同），其中绝大多数类名没有被使用。它用上了 jsx 和 inferno（类似 React），却因为 Hexo 的限制，没法让 inferno 跑在前端，而是用了 jQuery，为了处理时间和日期，它又引入了 moment.js，这两者加在一起带来了另外 400k 的消耗。

开发时间比 icarus 晚了两年，因此也在 js 上表现好一点的 volantis 没有使用 jQuery 和 moment.js，成功把 js 体积干到了 150k，但却依旧有着大约 700k 的 css 体积——绝大部分由 Font Awesome 贡献。主题作者显然无法删除它，因为主题作者**无法预测**用户会用到 Font Awesome 里的哪些图标，却又无法

**按需加载**。一个办法是把这些分离到配置文件里，让用户改配置文件，从而自行决定要加载哪些图标，但这显然也不是一个足够好的解决方案。

这些在我看来浪费资源的行为，几乎都是因为 Hexo 的架构缺陷。Hexo 通过 Renderer 去处理 markdown 和模板，然后把结果交给 Generator，Generator 再把结果写入文件。这样的架构，使得 Hexo 无法在渲染过程中对前端代码进行优化。例如，在渲染过程中，一个 Hexo 主题无法把用户用到的 Font Awesome 类名提取出来，然后把 css 里的其它类名删掉。这超出了 Hexo 的能力范围。

我不能把我的大脑当成 JavaScript 解释器，去看看全局对象被改成了什么样子；我也不能靠大脑优化 css 的内容。Hexo 能提供的方案（构建之后遍历 html，剔除未使用的 css 类名）对我来说还是不够用，而且显然增加了一个多余步骤——更不用说绝大多数 Hexo 主题也没这么做。2022 年 3 月，我学会了 Vue3，Hexo 的这些缺陷变得更加不可容忍了。这之后，我把我的博客切换到了 VuePress2。

我使用 vuepress-theme-hope 作为我的博客主题，它的作者同时也是 waline 的前端作者和 VuePress2 的核心维护人员。这几个项目都有着极其现代的架构，TypeScript 的类型也很完备，一些项目还有自动化测试，我可以很轻松地阅读、分析、修改，甚至提交贡献，而丝毫不用担心破坏原有的功能。作者还把主题的大部分功能拆成了独立的插件，这样其它主题也可以使用。遗憾的是，VuePress2 的生态并不好，我目前没有见到第二个完成度像 vuepress-theme-hope 这样的主题。

不过，VuePress 是一个 SPA，这在加快了页面切换速度的同时，也让它的 js 体积变得很大（vuepress-theme-hope 文档页面首屏 js 900k），拖慢了它的首屏加载速度。VuePress 更适合拿来给 vue 项目写文档，因为它可以无缝引入 vue 组件，做一个 tree-shaking，删掉没用到的部分，然后在不同页面按需引入特定的模块。但对于博客来说，首屏加载速度比页面切换速度更重要，因为用户更多地是在别处点击链接进来阅读一两篇文章，而不是在频繁地切换页面。

最后，我选择了 Astro。Astro 是一个和 Hexo 那样的静态页面生成器，但与 Hexo 不同的是，它带来了群岛架构、TypeScript、ESM 语法、MDX 和完全自动化的按需加载。Astro 的模板系统使用了类似 jsx

的语法，能像写 jsx 一样，用 esm 的 import 语法去引入一个组件或布局，而不用担心使用了但是没有引入、引入目标不存在或引用了没被使用的东西这样的问题，因为它有完整的 TypeScript 支持。像“漏掉一个大括号”这样的问题则更不可能发生。Astro 的群岛架构使用了 Web Components，并且要求浏览器支持 ES Module，不过我们不能因此说 Hexo 生成的页面的兼容性比 Astro 好——Hexo 没有群岛架构这种东西，不能拿一个不存在的功能去比较兼容性。群岛架构和 MDX 让 Astro 能随意引入使用了 React、Vue、Svelte、Angular 和 Solid 这样的前端框架写的组件——甚至是在 Markdown 里这么干。

例如，这是一个使用 solid.js 实现的计数器，而我在博文里只用一行代码就引入了它。你可以点击下面的两个按钮查看效果。

0s

-1s +1s

再来插一个图标 ，Astro 的图标插件 和 UnoCSS Preset Icons 均支持引入 Iconify 上的任何图标，其中当然包括 Hexo 博客主题常用的 Font Awesome。但使用它们引入的图标是**按需加载**并以 svg 的形式（而非字体）内联的，没用到的图标**永远**不会被加载。

现在我们可以说，除了生态不如 Hexo，Astro 就是 Hexo 的完全上位替代——体积更小、功能更多、开发更便捷。只是因为刚刚出现，Astro 的生态比 VuePress2 的还差。幸好该有的东西全都有，而且很大程度上它能复用已有的现代前端技术成果（比如 tailwindcss 和我正在使用的 UnoCSS）。于是我在上面从零开始写了一个自己的博客主题。我对自己的审美能力有些自知之明，知道自己是开发者而非设计师，因此我照抄了 `hexo-theme-icarus` 的页面布局。在总共约几个小时的 coding 以后，我实现了我想要的大部分功能（RSS、SEO、友链和评论系统），然后把自己的博客换成了它，从而实现了总共只有 35k 的首页加载体积。即使在博文页面（40k），加上 katex 的 23k（css），和 182k 的 waline 评论系

统（css+js）后，体积也依然比 hexo-theme-volantis 更低。比较可惜的地方是，在迁移过程中，我发现老博客有些地方破坏了我的身份隔离，因此我重置了历史记录，并且 `push -f` 了一遍。如果不是这样的话，我将能用 commit history 展示自己从 Hexo 换成 VuePress2 再换成 Astro 的全过程。

自 2013 年 Hexo 发布首个版本至今，已经过去了 9 年。尽管 Hexo 仍然是一个非常优秀的静态页面生成器，它的生态和社区都很完善，文档也很详细，但它的维护和开发成本已经显著高于期望。Hexo 作为旧时代的产物，已经和 jQuery 那样到了该被抛弃的时刻，而 VuePress、Docusaurus、Next.js、Nuxt.js 和 Astro（取决于使用场景）将取而代之。曾经 Vue 的文档是用 Hexo 写的，后来换成了 VuePress，现在则是 vitepress。前端更新换代极快，而时间不等人。如果说博客主暂时还只能在已有的生态里挑选，那么开发者则更应该立即停止为 Hexo 生态继续贡献代码[2]，而是转向 Astro，这将是一场双赢——开发者不再需要头疼 Hexo 固有架构带来的问题，而 Astro 的生态也能得以发展。

——只是，写博客的人已经越来越少了，一个“**夕阳产业**”能有足够的动力吗？这就是技术以外的问题了。

> 如果觉得上文对体积的比较不够客观，可以进入下面几个参考页面，按 F12 抓包，排除掉我博客没有的功能（如看板娘）以后，对照一下体积。
>
> - *hexo-theme-volantis 作者之一的博客*
> - *hexo-theme-icarus 文档页面*
> - *vuepress-theme-hope 作者的博客*
> - *惶心博客 - 该博客使用了较为轻量的 hexo-theme-iris，链接内博文的 html+css（不含评论系统和音乐播放器）共 48k。*
> - *宝硕博客 - 博主是 Hexo 的开发人员之一，~~这个使用 Hexo 和 Next.js 写的博客可以代表 Hexo 博客开发的最高水平~~。与本人确认过，这个博客只是把 Hexo 当成 CMS 来用，网页是由 Next.js 生成的。*
>
> 我选择的页面都是比较短的文章，至少比我这篇文章更短。

1. 后来我知道，Hexo 的 Renderer 层是分离的，可以用任何你想用的模板引擎。 
2. hexo-theme-yun 开发者云游君已经在半年前转向，甚至直接自己搓了一套博客系统 valaxy。看来英雄所见略同。 

教程 杂谈

▶ **分享**

我遭到了来自性侵者的持续骚扰，精神状态亦受影响。为了保护我的精神状态，原匿名评论区无限期关闭。在评论区的发言本就是公开的，故将其在两周左右时间内发布的近万字骚扰言论合订公布。打开阅读前，请务必确保自身精神状况。

在原评论区恢复使用前，还请注册 GitHub 账号以使用临时评论区