黑客派

《从零构建前后分离的 web 项目》：前端了解过关了吗？前端基础架构和硬核介绍

作者：pkwenda

原文链接：https://hacpai.com/article/1536056426888

来源网站：黑客派

前端准备 ： 前端了解过关了吗？前端基础架构和硬核介绍

技术栈的选择

首先我们构建前端架构需要对前端生态圈有一切了解，并且最好带有一定的技术**前瞻性**，好的技术架可能日后会方便的扩展，减少重构的次数，即使重构也不需要大动干戈，我通常选型技术栈会参考以三点：

一、提出自身业务的需求

- SEO 是否非常重要?
- 主要面向：移动端还是 pc 端?
- 是否有开发 app 的规划?

有了这样的问题我们可以带着问题去重点选型一些这写问题技术方案比较成熟的技术栈。

二、自身是否成熟，文档是否友好

这里举一个以前开发过程中实际遇到的，当时为了优化用户体验，节省开发效率 选型了一款 MVVM 量框架，可惜当时没有决定权，CTO 选型了 avalon

当时之所以没有选择 backbone ，主要是因为没有成熟的中文文档，考虑到团队的流动性和上手性时没做考虑，最终选择了 司徒正美的 avalon 当时来说还是比较前卫的，也有一些以去哪网为首的大司都在用。我们当时用的时候 avalon2 刚出不久，直接用的 2.0，使用过程也出现了一点问题：文档散，这一块那一块，存在后置性，生态少，扩展性价比不高 ，有时候遇到匪夷所思的 bug 寻找原因了几遍 demo 、文档 可能会找到答案，没有重点标识。当然就当时来说确实是给我们提升了部分开效率，但是我可能当时更偏向 Angular 或 vue 的。因为他们有无以伦比的生态圈和各种问题的技术案以以及完善的开发者文档，值得一提的是 avalon 的作者是兼职维护的，如果全栈运营的话，我相远比现在更好，看一看 avalon 的源码也会对自己有不少的提升。对于生产的技术选型要更加谨慎。

三、了解其生态系统

上文提到了生态系统，以我比较常用的 vue 来举例，vue 发展至今仅官方为我们提供了以 **vuex、 vue-router、 vue-loader、 vue-cli、 vuepress、 vue-devtools、 vue-ssr** 为首的 89 个开源项目包括无数的 vue 相关的 UI 库，vue 插件 甚至是近两年淘宝提供的 Hybrid： **weex** 的支持

截止今天 github 开源的 与 vue 相关的项目多达 167,752 个，与 angular 相关的多达 416,811 个，与 react 相关的 多达 594,272 个。

统计时间 2018-09-01

我想有了这样的生态支持，完全可以满足我们中小项目的 95% 以上的需求，至于比较哪个更强是没意义的 。

因为比较熟悉让我斗胆私自选择 vue 作为我们的 SPA 主架构

四、画出我们期望的前端基础架构模型

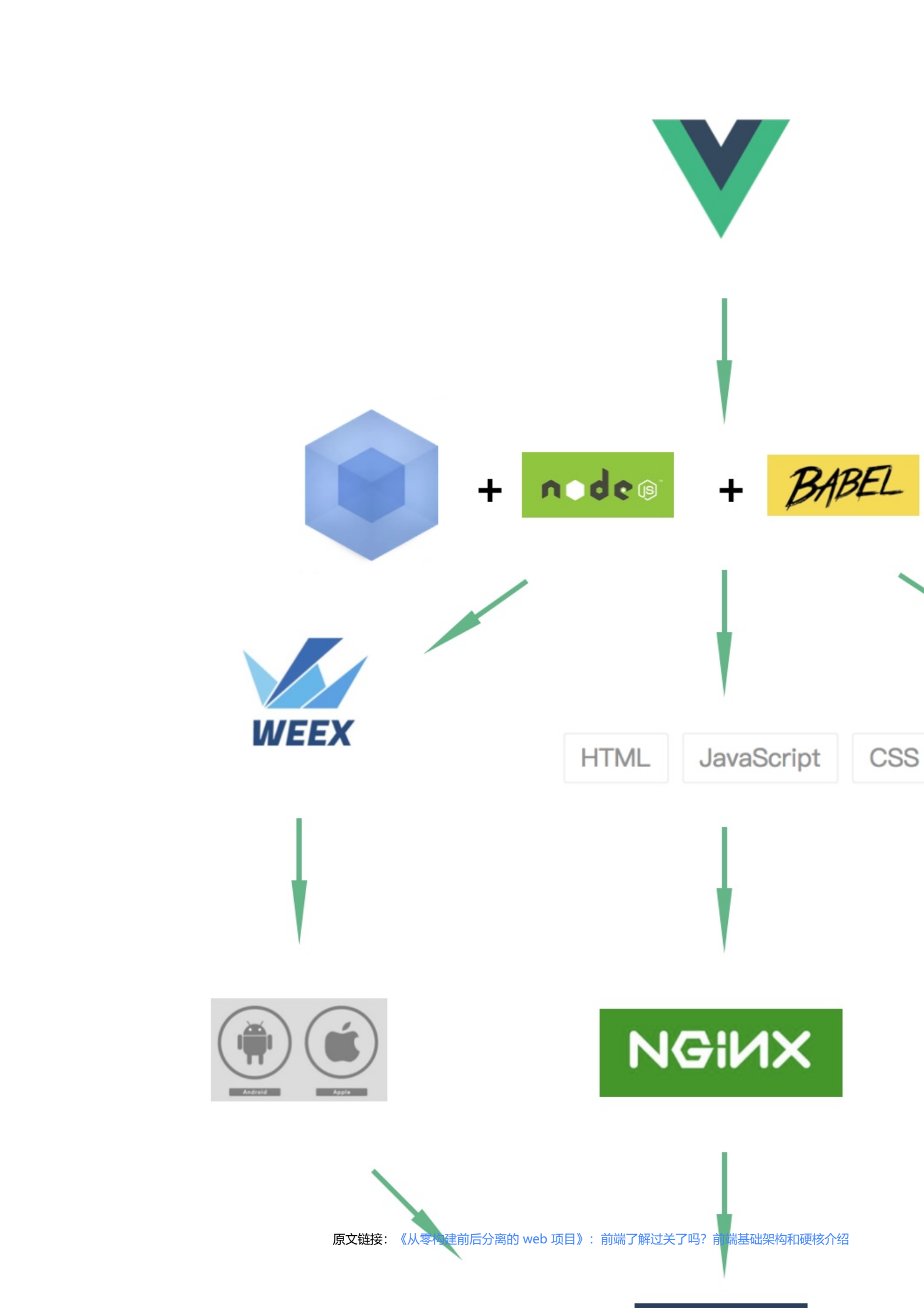
因为我们上一章选型了 Vue，如果只考虑前端我们最初的想法:技术栈大概是这样的：

通过 node 和 webpack 的支持 把 vue 组件 build 打包成传统元素，发布到 http 服务中，请求后端

原文链接：《从零构建前后分离的 web 项目》：前端了解过关了吗？前端基础架构和硬核介绍

务。

随后可能是这样的：



原文链接：《从零构建前后分离的 web 项目》：前端了解过关了吗？前端基础架构和硬核介绍

随着目前主流第三方库的越来越多和技术的尝鲜、客户端的需求、或被动[不得不用]、或主动的去引了 babel less sass *.loader 和 hybrid 等组件库。

再后来的技术栈需要我们根据真正踩坑之后才会逐步完善

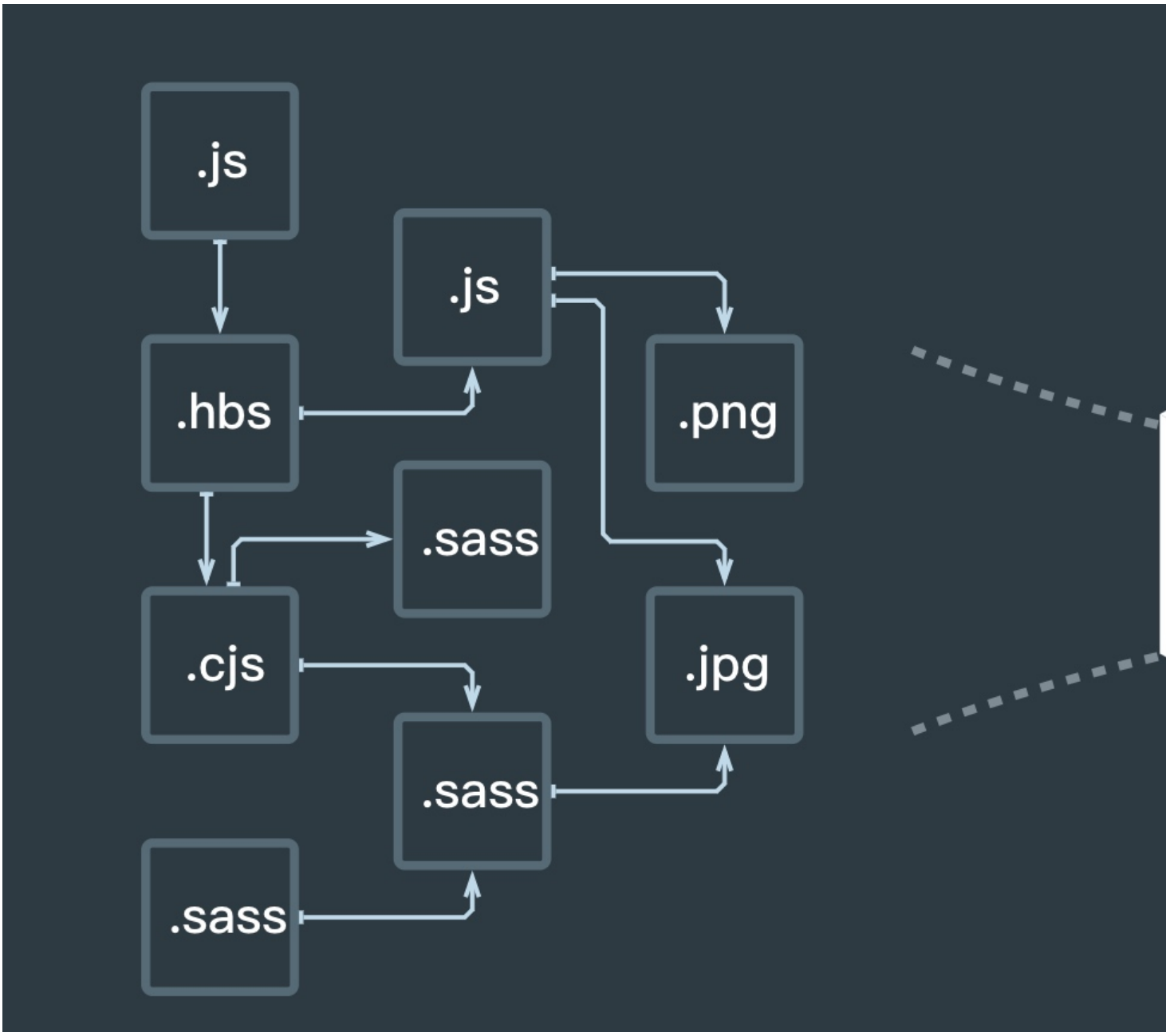
可能是 polyfill 懒加载 xss protobuf 等 针对 **浏览器兼容**、**速度优化**、 **SEO** 、**通信协议** 等具体问题。以，前期可以不用过多考虑，我们只要知道：这个问题我们是可以解决的，但是现在可以先不去考虑有些同学，太过于“**完美主义**”以至于想法不错，但动起手来做了几天就不做了，**完美主义害死人。**

了解 Webpack

WebPack 可以看做是模块打包机器，它可以分析你的项目结构，找到 JavaScript 模块以及其它的一浏览器不能直接运行的拓展语言：Stylus、Scss、less、TypeScript、CoffeeScript 等，并将其转换打包为合适的格式供浏览器使用。比较常用的还可以通过 webpack-dev-server 进行开发模式的**热更新**

WebPack 是一种模块化开发的方案

当 webpack 处理应用程序时，它会递归地构建一个依赖关系图(dependency graph)，其中包含应用序需要的每个模块，然后将所有这些模块打包成一个或多个 bundle



webpack 通过 loader 可以支持各种语言和预处理器编写模块，最后打包为一个（或多个）浏览器可别的 *JavaScript css * 文件

目前支持的 loader 列表

了解 ES6

原文链接：《从零构建前后分离的 web 项目》：前端了解过关了吗？前端基础架构和硬核介绍

官方说法

ECMAScript 6（简称ES6）是于2015年6月正式发布的JavaScript语言的标准，正式名为ECMAScript 2015（ES2015）。它的目标是使得JavaScript语言可以用来编写复杂的大型应用程序.

科普

很多人总是搞不清楚 ES 这些东西，这里大白话讲讲：

他们的先后顺序是：ES5、ES6(ES2015)、ES7、ES8

在 2015 年 6 月 ES6 的第一个版本发布， 正式名称就是 《ECMAScript 2015 标准》（简称 ES2015 算是 2011 年 ECMAScript 5.1 之后的 6.0版本

2016 年 6 月，小幅修订的《ECMAScript 2016 标准》（简称 ES2016）[因为改动小，其实他是 6.1 版本，但总有人愿意叫它 ES7 ， 不标准的]

2017 年 6 月发布 的《ECMAScript 2017 标准》（简称 ES2017） [因为改动小，其实他是 6.2 版本 但总有人愿意叫它 ES8 ， 不标准的]

就像 **Kubernetes** 人们开他起了一个 **K8S** 的名字 (K 和 S 中间有 8 个单词)，他是不标准的

了解 Babel Traceur

Babel、Traceur 是一个编译JavaScript的平台，它可以编译代码帮你达到以下目的：

JavaScript.next-to-JavaScript-of-today compiler

今天就使用未来的 JavaScript

截止发布日期 (2018-09-04)，没有一款完全支持ES6的JavaScript代理（无论是浏览器环境还是服务环境），所以热衷于使用语言最新特性的开发者需要将ES6代码转译为ES5代码。

让你能使用最新的JavaScript代码（ES6，ES7...），而不用管新标准是否被当前使用的浏览器完全支
；

ES7 作者完全没精力看，不过 Bable 逐渐替代了 Google 的 Traceur 成为主流了，我是个俗人，所我选 Bable

了解 Sass Less Stylus

Sass 是不是违反了中国的广告法了？？

Sass 、Stylus 和 Less 之类的预处理器是对原生CSS的拓展，它们允许你使用类似于variables, nestin , mixins, inheritance等不存在于CSS中的特性来写CSS，CSS预处理器可以这些特殊类型的语句转化浏览器可识别的CSS语句。

- **一张表格对比三语言**

语言	实现	特性	赋值	缩进
Sass	Ruby	变量$开头	$var: value	不需要
Less	JavaSript	变量@开头	@var: value	不需要
Stylus	NodeJs	不能使用@开头	var:10	都可以

你现在可能都已经熟悉了，上文讲 WebPack 讲过： webpack 里使用相关 loaders 进行配置就可以用了，以下是常用的CSS 处理loaders:

Less Loader

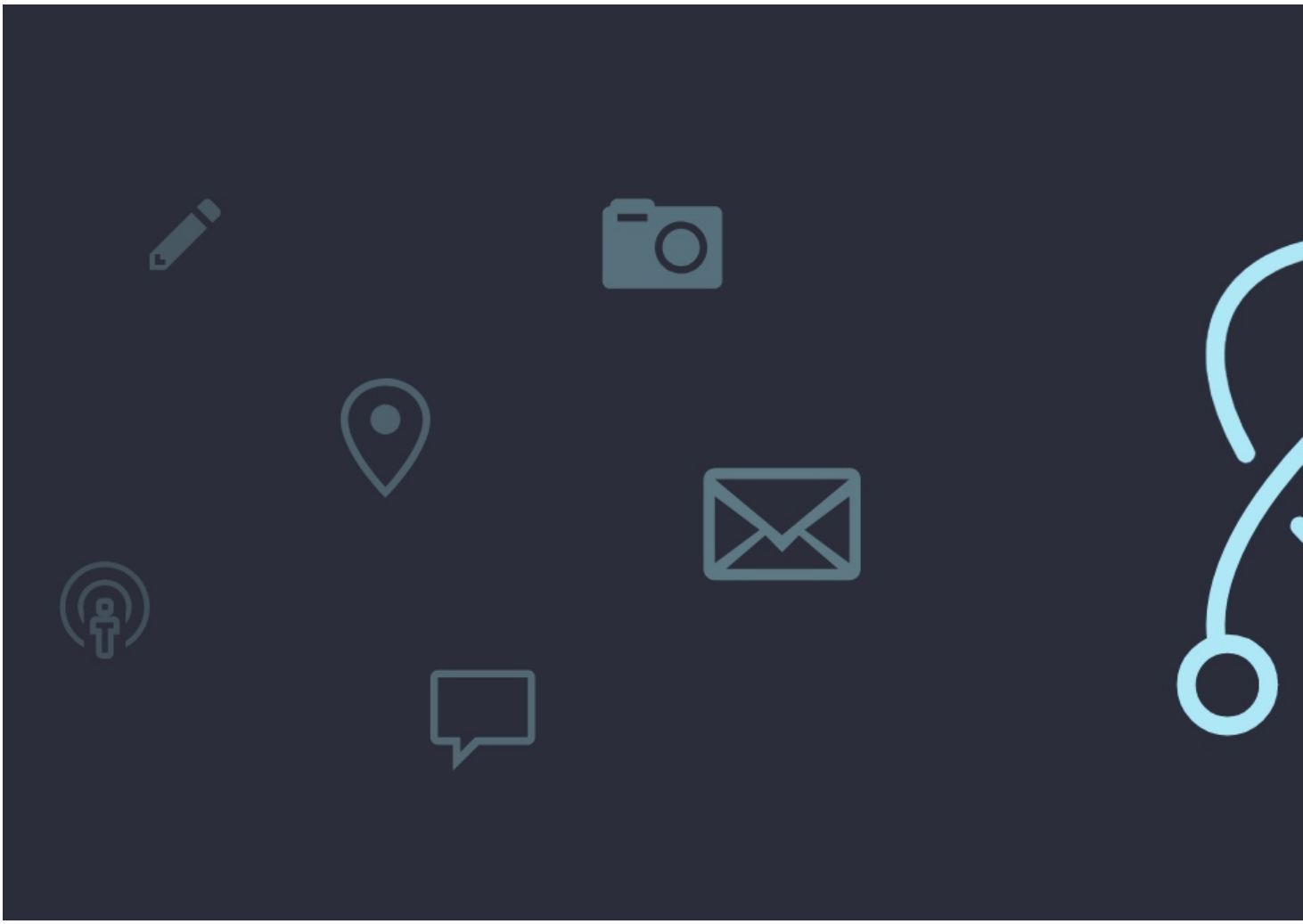
Sass Loader

Stylus Loader

自己去找：loader 列表

像：哪种语言更好、使用的更多、更简单 容易引起争议的 博主不想讨论，看自己喜好

了解 Electron



一个可以使用使用： JavaScript, HTML 和 CSS 构建跨平台的桌面应用的框架，也算 hybrid 的一种主要场景是 PC 端，没啥好说的。

值得一提的是 Visual Studio Code 、Atom、GIthub Desktop 都是基于此构建的，有时候按 CMD + option + i 有惊喜哦

基于 Electron 开发的APP列表

总结

这些也就基本是前端比较常用的主流技术栈组成的骨架了，之后的各种 webpack 插件，各种工具库选型随着项目实战引入更好，现在讲大家也记不住。

别急实战中的前端架构要比现在复杂得多，跟我一起循序渐进的的来。

下一章为大家实战：**《如何快速构建项目并升级为一个规范的前端骨架》**

关于我

- 目前在写 《从零构建前后分离项目》系列，修正和补充以此为准
- 不断更新的 项目实践地址

往期文章

《从零构建前后分离 WEB 项目》 序 - 开源的意义

《从零构建前后分离web项目》：开篇 - 纵观WEB历史演变

《从零构建前后分离web项目》探究 - 深入聊聊前后分离架构

《从零构建前后分离web项目》准备 - 前端了解过关了吗？前端基础架构和技术介绍