



了不起的 JavaScript 工程师

从前端到全端高级进阶

朱德龙◎著

总结多年丰富的前端开发经验
系统地介绍前端开发工作中需要的技能及工具
快速进入“全端”的境界

了不起的 JavaScript 工程师

从前端到全端高级进阶

朱德龙◎著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书讲述了开发者使用 JavaScript 在各种 Web 开发场景下所需要掌握的重点知识和概念。从最基础的开发工具讲起，再到开发思维方式和前端页面开发，然后扩展到小程序开发和开发工具的混合应用，再讲解前后端交互最常用的网络协议及 API 设计，最后讲解了使用 Node.js 开发服务器端应用程序所需要掌握的核心概念。

全书以数据链为线索，对重要概念进行精练的分析和对比，从而帮助读者更好地理解 and 记忆。本书既包括知识技能，又包括设计思想，“道”与“术”并重，让读者“知行合一”，既能“坐而论道”，也能“起而行之”。

本书适合初级和中级前端开发者用来提升技术水平与视野，建立较为完备的开发技能和可迁移的学习能力，帮助读者拥有全端的技术和全局的视野，成为“了不起的 JavaScript 工程师”。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目 (CIP) 数据

了不起的 JavaScript 工程师：从前端到全端高级进阶 / 朱德龙著. —北京：电子工业出版社，2019.8

ISBN 978-7-121-37129-5

I. ①了… II. ①朱… III. ①JAVA 语言—程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字 (2019) 第 150110 号

责任编辑：董 英 特约编辑：田学清

印 刷：三河市双峰印刷装订有限公司

装 订：三河市双峰印刷装订有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱

邮编：100036

开 本：787×980 1/16 印张：21

字数：504 千字

版 次：2019 年 8 月第 1 版

印 次：2019 年 8 月第 1 次印刷

印 数：3000 册 定价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888, 88258888。

质量投诉请发邮件至 zltz@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：(010) 51260888-819, faq@phei.com.cn。

前言

先给出下面三个问题：

- 如何快速成为职场老手，通过提升工作效率加快自己的成长速度？
- 前端技术框架层出不穷，该保持什么样的学习心态？
- 掌握 JavaScript 之后可以做哪些事情，有什么样的晋级路线？

术与道

对于第一个问题，要提升工作效率，大体上可分为两种途径：

- 改变开发习惯，选择好的软硬件环境；
- 积累经验，提升思维能力。

第一种途径是从“术”的层面，在微观、客观上进行提升，非常直接也非常有效，但这种途径也有缺陷，就是效率提升的空间有限。

本书的第 1 章便是教读者一些容易掌握的开发软硬件的知识及如何培养好的开发习惯，因为这些是能够在短时间内提升效率的方法。

第二种途径是从“道”的层面，宏观、主观地进行提升，提升速度缓慢，提升效果也很难快速直接地反映在工作中，但是提升空间可以说是无限的。

本书的第 3 章“高效编写/组织代码的心法”及第 4 章“模块”都属于此类，掌握这些基本原则和思路对前端知识甚至其他语言的学习都会有很大帮助。

JavaScript 是语言工具，是“术”；但是如果能在熟悉 JavaScript 之后进行总结归纳、建立自己的学习模型、通过模型来快速掌握其他高级语言，那么这就是“道”。

关于“术”与“道”的理解，大家可能会有以下两个误区。

1. “道”太难了，把“术”研究透了也能成为高手

一部分读者可能认为“量变引起质变”，只要“术”用得多了，自然能提升“道”。

“量变引起质变”其实是个半真命题，因为通常情况下引起质变是有条件的，这些条件既可能是主观创造的，也可能是客观存在的。

举个例子，比如一个电商网站，每天只有 1000 人使用，但是突然业务暴增到每天一亿人。如果不对系统架构进行调整可能就无法正常运行，这就是客观要求引起的质变。

再举个反例，比如一个开发者做了很多前端项目，“开发经验很丰富”，但是面试的时候很多问题却答不上来。网上的“你是具有 3 年开发经验，还是把 1 年开发经验重复了 3 遍”说的就是这种现象。

对于这种情况虽然有量变，但如果缺乏主动的归纳总结提升，很难引起质变。

有的读者可能会疑惑，如果这位开发者不断地使用新框架，比如从 jQuery 切换到 Vue 算不算质变呢？这就好比一个人之前骑摩托车出行，现在改开汽车出行了，这算是生活的质变吗？

2. “术”不重要，只要潜心向“道”便能成为高手

这种理解也相当理想化。“术”与“道”的关系就好像实践与理论的关系，两者相辅相成。精通“术”之后可以明白“道”，懂了“道”之后又可以用于“术”。

那些理解了“道”的高手很多时候都是从“术”开始学习的。比如《神雕侠侣》中的独孤求败，就是不断地提升剑术来减少对剑本身的依赖的，最后掌握剑道，以至于“不滞于物，草木竹石均可为剑”。

本书一方面会讲解技术重点，比如 HTTP 协议、浏览器，同时也更加注重关于“道”的知识讲解，比如从单页应用开发讲到小程序开发、从 JavaScript 前端讲到其在 Node.js 后端的应用。通过不同知识点重要概念的抽象与对比，最终让读者形成可迁移的、适应性强的学习能力。

危与机

其实开始提出的第二个问题暗示了前端工程师的两个危机。

1. 职业天花板较低

前端工程师相对于算法工程师、数据库管理员等其他开发岗位来说，并不是一个有较高天花板的职位（不一定随着工作年限增加而不断提升个人能力和薪酬待遇）。相反，由于前端技术框架层出不穷，一般情况下，先入行的工程师和后入行的工程师相比，难以有较大的优势。用了几年 jQuery 的开发者 and 入职一年的初级开发者相比，学习使用 Vue 和 React 并不具有太多的优势。技术总监和 CTO 很少有前端工程师出身的。

2. 职业门槛较低

学会 HTML、CSS 这种静态语言就可以实现一个网页效果，JavaScript 上手难度也不算大，同时容错率较高，也就是说，如果代码出现 bug 通常表现在界面和交互上，一般不会对公司造成较大影响，很多控制台错误不容易被客户察觉到。所以，很多 IT 培训机构必开前端培训课程，除了市场需求，门槛低也是一个重要的因素。

那么在这种情况下前端工程师该怎么通过学习和积累来建立自己的优势呢？

至少可以从以下两个方面入手。

1. 提升学习能力

如前面提到的“术”与“道”，就能有效地提升学习能力与学习效率。有了强大的学习能力之后，很多新技术都能快速理解并上手。这样的开发者的能力可以说是无限的，自然也不存在天花板。

2. 多维度学习

本书以前端为起点，紧贴常用的 Web 技术知识（Vue、Angular、React、Node.js、HTTP 等），帮助大家成为能独立完成开发任务的“全端工程师”。全端提供的是另一种可能性，带你了解不同的开发技能点，通过多种技能的组合使你成为真正强大的开发者。

树与叶

我曾经参加过一场线下技术分享沙龙，有一场关于架构师的讨论，一位同学说到自己想成为前端架构师，然后罗列了一大堆技术名词及要学习的点，如看一看 Vue 的源代码、熟练掌握 Webpack 的使用等。这些点看上去都没错，也紧贴热点，但是这位同学以目前的状态可能很难达到目标。因为他没有形成自己的知识框架！

前端的知识点其实很多，也很细碎，对于开发者而言，并不需要把它们都记住，而是要形成自己的知识框架，然后不断地去丰富它。对于新手而言可能比较难，不一定能做到，但是心中一定要有这个意识。

以本书的写作线索为例。一条明线是围绕 JavaScript 这门语言进行编写的，讲述了 JavaScript 的主要使用场景及核心知识，一条暗线便是根据数据流来进行编写的，即从用户端到服务器端；除了这两条纵向的线索，还有一条横向的线索就是从前端到多端（小程序）扩展。这些就是知识框架的体现。

只有先形成框架才能有效地把繁杂的知识点内化成为自己的经验。用罗振宇的话来说就是“把新知识缝合到自己原有的知识体系中”。学无涯而生有涯，知识框架还能帮助我们对知识进行过滤，避免在一些无关的知识点上消耗过多时间。

致谢

个人能力再强，也毕竟是有限的。一个人的成就，往往离不开其他人的帮助与支持。所以本书能够出版要感谢很多人。

感谢家人对我写书的支持。

感谢董英编辑的信任和指点，以及其他编辑的细心校对。

感谢一直支持我的读者，这其中包括本书的读者及博客文章的读者。

感谢学习我慕课网课程学生的支持。

感谢对于我写书提出意见的朋友们。

感谢参考文章和资料的作者。

感谢所有曾经帮助我成长的人。

.....

读者服务

轻松注册成为博文视点社区用户 (www.broadview.com.cn)，扫码直达本书页面。

- **下载资源：**本书如提供示例代码及资源文件，均可在 [下载资源](#) 处下载。
- **提交勘误：**您对书中内容的修改意见可在 [提交勘误](#) 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动：**在页面下方 [读者评论](#) 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/37129>



反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010)88254396；(010)88258888

传 真：(010)88254397

E-mail: dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱 电子工业出版社总编办公室

邮 编：100036



目 录

第 1 章 开发环境	1
1.1 代码编写工具	2
1.1.1 IDE	2
1.1.2 编辑器	3
1.1.3 最佳选择	3
1.2 Docker 容器	5
1.2.1 Docker 概述	5
1.2.2 Docker 重要概念	6
1.2.3 Docker 使用场景	11
1.2.4 Docker 扩展	15
1.3 代码管理	15
1.3.1 什么是代码仓库	16
1.3.2 版本管理的意义	16
1.3.3 版本管理的常用操作	16
1.3.4 分支管理的意义	17
1.3.5 分支管理的常用操作	17
1.3.6 分支管理流程	18
1.4 其他软件	23
1.4.1 文件夹管理软件	23
1.4.2 快速搜索工具	24
1.4.3 终端管理软件	25
1.4.4 Chrome 中的插件	25
1.5 硬件提升	26
1.5.1 提升程序运行速度	26
1.5.2 减少程序切换时间	27

1.6 小结	28
第 2 章 Web 页面与多页应用	29
2.1 Web 页面的运行环境——浏览器	29
2.1.1 渲染引擎	30
2.1.2 JavaScript 引擎	38
2.1.3 数据持久层	40
2.2 HTML	41
2.3 模板	42
2.3.1 模板的意义	42
2.3.2 常用模板分类	42
2.3.3 模板的重要功能	42
2.4 CSS 盒模型	45
2.5 CSS	46
2.6 CSS 布局	47
2.6.1 普通文档流	47
2.6.2 浮动	57
2.6.3 定位	61
2.6.4 弹性盒模型	62
2.7 CSS 预处理	68
2.7.1 预处理的意義	68
2.7.2 预处理的重要功能	69
2.7.3 样式文件规划	72
2.7.4 样式类命名	73
2.8 JavaScript	77
2.8.1 实现功能逻辑	77
2.8.2 操作页面或浏览器	78
2.8.3 进行网络通信	79
2.8.4 第三方 JavaScript 库——jQuery	84
2.8.5 JavaScript 简史	84
2.9 自动化构建工具	85
2.9.1 自动化构建工具的作用	85
2.9.2 常用的自动化构建工具	85
2.10 小结	86

第 3 章 高效编写/组织代码的心法	87
3.1 拆分方式	89
3.1.1 按文件类型拆分	89
3.1.2 按功能类型拆分	90
3.1.3 按关注点拆分	90
3.2 抽象原则	91
3.2.1 第一原则：DRY	91
3.2.2 第二原则：YAGNI	92
3.2.3 第三原则：The Rule of Three	93
3.3 不止于代码	94
3.4 小结	94
第 4 章 模块	96
4.1 模块的意义	96
4.2 ECMAScript 5 中的模块	96
4.2.1 立即执行函数表达式（Immediately-Invoked Function Expression）	97
4.2.2 显式模块声明	97
4.2.3 异步模块定义	98
4.2.4 共同模块定义	99
4.2.5 CommonJS	100
4.2.6 通用模块定义	101
4.3 ECMAScript 6 中的模块	102
4.4 模块打包工具	103
4.5 小结	107
第 5 章 单页应用（SPA）	108
5.1 框架	109
5.2 视图与数据	110
5.2.1 双向数据绑定	111
5.2.2 单向数据流	115
5.3 路由	116
5.3.1 hash	116
5.3.2 history	117
5.4 组件	119

5.4.1	原生组件	119
5.4.2	第三方组件	121
5.5	小结	123
第 6 章	JavaScript 的几个趋势	124
6.1	接口与数据类型	125
6.2	更好的异步解决方案	127
6.3	面向对象与类	132
6.3.1	封装	132
6.3.2	继承	134
6.3.3	多态	135
6.4	模块化	138
6.5	小结	141
第 7 章	小程序概述	142
7.1	常见的 App	142
7.2	JavaScript 开发者的一双翅膀	143
7.3	小程序的发展	144
7.4	小结	144
第 8 章	小程序与 Web 页面	145
8.1	WXML	145
8.1.1	WXML 与 HTML 的相同之处	145
8.1.2	WXML 与 HTML 的不同之处	146
8.2	WXSS	149
8.2.1	WXSS 与 CSS 的相同点	149
8.2.2	WXSS 与 CSS 的不同点	149
8.3	JavaScript 与 WXS	155
8.4	JSON	155
8.4.1	app.json	155
8.4.2	project.config.json	156
8.4.3	page.json	157
8.4.4	component.json	158
8.5	小结	158

第 9 章 小程序与单页应用	159
9.1 路由	159
9.1.1 路由配置	159
9.1.2 路由跳转	160
9.1.3 路由监听	161
9.2 组件	162
9.2.1 组件与页面	162
9.2.2 小程序组件与单页应用组件	164
9.3 web-view	165
9.3.1 作用	165
9.3.2 交互	165
9.4 小结	165
第 10 章 小程序的框架与插件	166
10.1 RxWX	166
10.2 WePY	168
10.3 mpvue	171
10.4 Taro	171
10.5 小结	171
第 11 章 小程序的开发工具与发布	173
11.1 开发者工具组成	173
11.1.1 模拟器	174
11.1.2 调试器	174
11.1.3 编辑器	176
11.1.4 其他功能	176
11.2 发布流程	177
11.2.1 小程序的版本	177
11.2.2 用户身份与权限	178
11.3 小结	178
第 12 章 其他混合应用简介	180
12.1 流应用	180
12.2 桌面应用	181
12.3 小结	182

第 13 章 HTTP 协议与 Web 网站	183
13.1 HTTP 的历史	183
13.2 HTTP 的通信方式	183
13.3 HTTP 的状态	184
13.4 小结	184
第 14 章 HTTP 协议内容	185
14.1 请求行/状态行	186
14.1.1 URL	186
14.1.2 请求方法	188
14.1.3 状态码/状态信息	188
14.2 头部	192
14.2.1 通用头部字段	192
14.2.2 请求头部字段	194
14.2.3 响应头部字段	196
14.2.4 主体头部字段	196
14.2.5 其他头部字段	197
14.3 主体	198
14.4 Cookie	199
14.4.1 Cookie 的分类	199
14.4.2 Cookie 的使用	199
14.4.3 Cookie 的缺陷	200
14.4.4 Cookie 与存储	201
14.5 Cookie 与状态	204
14.5.1 状态存储	205
14.5.2 状态获取	207
14.5.3 基于 token 的认证方式更好	207
14.6 小结	208
第 15 章 HTTP 请求优化	209
15.1 减少连接/请求数	209
15.1.1 减少请求	209
15.1.2 减少连接	210
15.2 缓存数据	210

15.3 减少传输数据量	210
15.4 优化网络链路	210
15.4.1 减少域名	211
15.4.2 使用 CDN	211
15.5 小结	211
第 16 章 HTTP/2 协议	212
16.1 多路复用	212
16.2 压缩	214
16.3 支持 TLS	215
16.4 应用层协议协商	215
16.5 服务器端推送	215
16.6 流控制	216
16.7 小结	216
第 17 章 HTTPS 协议	217
17.1 HTTP 的缺点	217
17.1.1 通信使用明文	217
17.1.2 不验证通信方身份	218
17.1.3 无法证明报文的完整性	218
17.2 理解 HTTPS	219
17.2.1 HTTPS 通信流程	219
17.2.2 密码学基础	221
17.2.3 摘要与签名	221
17.2.4 X.509 与证书	222
17.3 小结	223
第 18 章 WebSocket 协议	224
18.1 WebSocket 与 HTTP	224
18.2 WebSocket 的使用	224
18.2.1 客户端	225
18.2.2 服务器端	226
18.3 小结	227
第 19 章 API	228
19.1 REST	228

19.1.1	REST API 设计	228
19.1.2	REST API 工具	230
19.2	GraphQL	231
19.2.1	GraphQL 设计	232
19.2.2	GraphQL 工具	234
19.3	小结	235
第 20 章	Node.js 概述	237
20.1	为什么要学习 Node.js	237
20.2	什么是 Node.js	238
20.3	Node.js 的历史	239
20.4	Node.js 的结构	240
20.5	Node.js 的运行机制	241
20.5.1	单线程	241
20.5.2	事件循环	242
20.6	学习 Node.js 的三个挑战	244
20.6.1	I/O 回调	244
20.6.2	代码性能	244
20.6.3	多进程协作	245
20.7	小结	245
第 21 章	用 Node.js 编写 Web 服务器端	246
21.1	处理请求	246
21.1.1	创建服务器端	246
21.1.2	解析请求	247
21.2	响应结果	250
21.2.1	状态信息	250
21.2.2	响应头部	251
21.2.3	响应主体	251
21.3	路由解析	253
21.4	I/O 操作	260
21.4.1	文件	260
21.4.2	数据库	264
21.4.3	网络请求	267
21.5	Web 框架	270

21.5.1 Express	271
21.5.2 Koa	271
21.6 小结	271
第 22 章 Node.js 内存控制	273
22.1 内存限制	273
22.2 内存管理	276
22.2.1 内存分配	276
22.2.2 内存回收	276
22.3 内存泄漏	277
22.3.1 缓存	277
22.3.2 不断增长的数组	279
22.3.3 重复的事件监听	279
22.4 大内存处理	281
22.5 小结	283
第 23 章 Node.js 多进程	284
23.1 PM2 模块	284
23.1.1 安装与使用	285
23.1.2 进程通信	286
23.1.3 进程管理	287
23.2 cluster 模块	290
23.2.1 基本使用	291
23.2.2 进程通信	294
23.2.3 进程管理	294
23.3 child_process 模块	298
23.3.1 基本使用	298
23.3.2 进程通信	298
23.3.3 进程管理	302
23.4 worker_threads 模块	303
23.4.1 基本使用	303
23.4.2 线程通信	304
23.4.3 线程管理	305
23.5 小结	306