



JavaScript at Scale

# 大型JavaScript应用 最佳实践指南

以一线前沿JavaScript开发者对可扩展性深刻的洞悉力，构建历久弥新的JavaScript应用

[加] Adam Boduch 著  
奇舞团 译



中国工信出版集团



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
www.phei.com.cn

JavaScript at Scale

# 大型JavaScript应用 最佳实践指南

[加] Adam Boduch 著  
奇舞团 译

电子工业出版社  
Publishing House of Electronics Industry  
北京·BEIJING

## 内 容 简 介

本书以介绍扩展 JavaScript 的特殊性，及影响其可扩展性的因素作为开头，逐步深入地介绍了组件的复合与通信、寻址与导航、用户偏好与默认设置、加载时间和响应速度、可移植性和测试、缩小规模、错误处理等大型 JavaScript 应用中的实践经验。本书将教会你如何在真实项目中扩展 JavaScript 应用，设计出灵活的架构。书中的每个主题都涵盖了实践指导，帮助你将知识运用到实际项目中。

Copyright © 2016 Packt Publishing. First published in the English language under the title 'JavaScript at Scale'.

本书简体中文版专有出版权由 Packt Publishing 授予电子工业出版社。未经许可，不得以任何方式复制或抄袭本书的任何部分。专有出版权受法律保护。

版权贸易合同登记号 图字：01-2015-7447

### 图书在版编目（CIP）数据

大型 JavaScript 应用最佳实践指南 / (加) 亚当·博达哈 (Adam Boduch) 著；奇舞团译. —北京：电子工业出版社，2017.2

书名原文：JavaScript at Scale

ISBN 978-7-121-30706-5

I. ①大… II. ①亚… ②奇… III. ①JAVA 语言—程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字（2016）第 311425 号

策划编辑：张春雨

责任编辑：徐津平

印 刷：三河市鑫金马印装有限公司

装 订：三河市鑫金马印装有限公司

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：14.75 字数：285 千字

版 次：2017 年 2 月第 1 版

印 次：2017 年 2 月第 1 次印刷

定 价：65.00 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](mailto:zltz@phei.com.cn)，盗版侵权举报请发邮件至 [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)。

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

# 关于作者

Adam Boduch 在开发大型 JavaScript 应用方面有近 10 年的工作经验。在转型为前端工程师之前，他曾使用 Python 与 Linux 参与了许多大型云计算产品的构建。Adam 拥有非常丰富的开发经验，擅长处理复杂的场景，提高软件的可扩展性。他编写了很多 JavaScript 方面的书籍，其中包括 *Lo-Dash Essentials*，并且，他还热衷于优化用户体验和性能。

Adam 现居住于多伦多，是 Virtustream 的一名高级软件工程师。

---

我想在此感谢我的妈妈和爸爸。

---

# 关于审校者

**August N. Marcello III** 是一位充满激情的软件工程师，在客户端的 Web 应用架构相关的设计、实现、部署方面，有着近 20 年的工作经验。他专注于基于 SaaS 创造良好的用户体验，并将其传播到 Web 生态系统，这无论从个人还是从专业角度来说都极具价值。对新兴通用技术的热爱以及对先进的 JavaScript 平台的专注，驱动着他在技术上精益求精。在工作之余，他会参加越野跑、山地自行车骑行，或者陪伴家人和朋友。他的个人网站为：[www.augustmarcello.com](http://www.augustmarcello.com)。

---

非常感谢 Chuck、Mark、Eric 和 Adam，很荣幸能够跟他们一起工作和学习。谢谢我的家人、朋友，还有我所经历的一切。

---

**Yogesh Singh** 毕业于印度 JSS 技术教育学院。他是一位全栈 Web 开发者，在服务端 Web 开发栈方面（ASP.NET 以及 Node.js）很有经验，而且熟练掌握 HTML、CSS 以及 JavaScript。

Yogesh 热爱 JavaScript 以及相关的库和框架（Backbone、AngularJS、jQuery 和 Underscore）。

他最开始从事的是数据挖掘和数据仓库方面的工作，在数据库开发方面十分专业。他是 MSSQL 的微软认证解决方案成员（MCSA）。

Yogesh 自学能力很强，喜欢学习算法和数据结构，并在斯坦福大学 Coursera 上获得了算法课的结业证明。

他曾就职于 OLX India 和 MAQ Software，目前为 Gainsight 公司的全栈开发者。

业余时间，他喜欢在 <http://mylearning.in> 上写博客。他的 LinkedIn 简历地址为 <https://www.linkedin.com/in/yogesh21>。

---

感谢我的家人、朋友以及同事的支持。

---

**Nikolay Sokolov** 是一名软件工程师，他在云计算、自动化部署和企业软件开发方面有着丰富的经验。现在就职于 Tonomi (<http://tonomi.com/>)，负责基于弹性组件模型分发云应用的自动管理包。

可通过 <https://twitter.com/chemikadze> 随时联系他。

**Serkan Yersen** 是一名洛杉矶的软件开发者。他是一些开源库的作者，例如：[ifvisible.js](#)、[underscore.py](#) 以及 [kwards.js](#)。Serkan 专门从事构建大型 JavaScript 应用，以及为用户广泛的应用创建 UI。2006 年至 2012 年，就职于 <http://www.jotform.com/> 期间，他开发了一个复杂的表单生成器，供上百万用户使用。现在，他就职于 Home Depot 和 Redbeacon (<http://www.redbeacon.com/>)，负责 Web 应用开发。你可以访问他的个人网站：<http://serkan.io/>。

# 关于译者

本书翻译工作由月影领衔的奇舞团翻译小组承担，由王伟华、黄小璐、黄薇负责翻译。

**王伟华**

网名 Aztack，前端技术专家。曾就职百度、奇虎 360 等国内知名互联网公司。拥有丰富的 Web 前端开发经验，擅长 JavaScript、Ruby、Java、C++ 等语言。

个人博客：<https://aztack.wang>

**黄小璐**

毕业于华中科技大学计算机学院。现为奇虎 360 软件开发工程师。曾参与开源项目 [stcjs](<https://github.com/stcjs/stc>) (高性能前端 workflows 系统)。参与翻译了《高性能 HTML5》等书。

**黄薇**

毕业于中山大学，于 2013 年加入奇舞团，近期参与了 Nova.js (Web Component 框架)、声享 (在线制作 PPT) 等项目，对大型 JavaScript 应用有浓厚的兴趣和丰富的开发经验。

以上三位译者曾共同参与《移动 Web 手册》一书的翻译工作。

读者可扫描以下二维码关注奇舞团周刊。



# 前言

能够一直正常运行的应用只是特例，大部分的 JavaScript 应用多多少少都有些问题。而这些问题产生的原因是由于我们总是习惯性地忽略可扩展性。这本书介绍了如何通过扩展前端架构来提高软件质量。扩展 JavaScript 应用是一件有趣的事情，需要考虑很多因素：用户、开发者、开发环境、浏览器环境等。我们的任务就是全面考虑这些因素，从而提供最佳的用户体验。我们要扩展什么？为何要扩展？本书将为大家解答这些问题。

## 本书内容

第 1 章，扩展 JavaScript 应用，介绍了何为可扩展的 JavaScript 应用，以及扩展 JavaScript 应用与扩展其他应用的区别。

第 2 章，可扩展性的影响因素，介绍了如何理解可扩展的需求，设计出更好的架构。

第 3 章，组件组合，介绍了构成架构核心的模式，以及如何以之为蓝图组合组件。

第 4 章，组件的通信与职责，介绍了组件之间的通信是如何制约扩展的。组件的通信模式功能有决定性作用。

第 5 章，寻址和导航，详细介绍了拥有指向不同资源的 URI 的大型 Web 应用，以及如何设计才能应对不断增长的 URI 数量。

第 6 章，用户偏好和默认设置，介绍了设置用户偏好的必要性，以及可配置的组件对扩展应用的重要性。

第 7 章，加载时间和响应速度，介绍了文件数量的增加是如何降低应用效率的。在添加新功能时，要有所舍弃，才能保证 UI 的响应速度。

第 8 章，可移植性和测试，介绍了如何编写不依赖于特定环境的 JavaScript 代码，包



括创建可移植的模拟数据和测试代码。

第 9 章，缩小规模，介绍了移除无用或错误组件对扩展系统其他部分的重要性。

第 10 章，处理错误，介绍了优秀的 JavaScript 架构不会因为某个组件的错误而崩溃。许多时候，在设计时充分考虑对错误的处理是成功扩展的关键。

## 阅读本书的条件

- NodeJS
- 代码编辑器/集成开发环境
- 一个现代 Web 浏览器

## 本书读者

本书的目标读者是对前端架构感兴趣的高级 JavaScript 工程师。阅读本书无须预备框架知识，但本书介绍的大部分概念都来自于框架，例如 Backbone、Angular、Ember。阅读本书需要扎实的 JavaScript 语言知识基础，本书中所有的示例代码都使用 ECMAScript 6 语法编写。

## 约定

在阅读本书时，你会发现许多代表不同类型信息的不同文本样式。这里展示了一些例子及其含义详解。


正文中的代码、数据库表名、文件夹名、文件名、文件后缀、路径、URI 示例、用户输入，以及 Twitter 用户名等均按此格式进行展示：“以 users/31729 为例。路由器应该找到一个模式，能够匹配该字符串，并能够提取出变量 31729”。

代码块按以下格式展示：

```
// 渲染试图的各个部分。
```

```
// 各部分可能有renderer，也可能没有。  
// 但不管有没有renderer，内容都会被返回。
```

 警告和重要提示都会按此格式展示。

 提示和技巧都会按此格式展示。

## 下载示例代码

你可以从 <http://www.broadview.com.cn> 下载所有已购买的博文视点书籍的示例代码文件。

## 勘误表

虽然我们已尽力确保内容的准确性，但错误仍然可能存在。如发现任何错误，可登录博文视点官网 <http://www.broadview.com.cn> 提交勘误信息。一旦勘误信息被本书作者或编辑确认，即可获得博文视点奖励积分，可用于兑换电子书。读者可以随时浏览图书页面，查看已发布的勘误信息。

# 目录

|                          |    |
|--------------------------|----|
| 1 扩展 JavaScript 应用 ..... | 1  |
| 影响扩展的因素 .....            | 2  |
| 对可扩展的需要 .....            | 2  |
| 不断增长的用户 .....            | 3  |
| 添加新功能 .....              | 3  |
| 雇佣更多的开发者 .....           | 4  |
| 架构角度 .....               | 5  |
| 浏览器是一个独特的环境 .....        | 5  |
| 组件设计 .....               | 7  |
| 组件通信 .....               | 7  |
| 加载时间 .....               | 8  |
| 响应性 .....                | 9  |
| 可寻址性 .....               | 9  |
| 可配置性 .....               | 10 |
| 架构性取舍 .....              | 11 |
| 确定不可变内容 .....            | 11 |
| 从开发的便捷性考虑性能 .....        | 11 |
| 性能的可配置性 .....            | 12 |
| 从可替换性考虑性能 .....          | 13 |
| 可寻址性的开发便捷性 .....         | 13 |
| 性能的可维护性 .....            | 13 |
| 减少功能以提高可维护性 .....        | 14 |

|                         |           |
|-------------------------|-----------|
| 利用框架.....               | 15        |
| 框架与类库.....              | 16        |
| 一致地实现模式.....            | 16        |
| 内建的性能.....              | 16        |
| 利用社区智慧.....             | 16        |
| 框架并非天生支持扩展.....         | 17        |
| 小结.....                 | 17        |
| <b>2 可扩展性的影响因素.....</b> | <b>19</b> |
| 扩展用户.....               | 20        |
| 许可证费用.....              | 20        |
| 订阅费用.....               | 21        |
| 消耗费用.....               | 21        |
| 广告支持.....               | 21        |
| 开源.....                 | 22        |
| 与用户沟通.....              | 23        |
| 支持机制.....               | 24        |
| 反馈机制.....               | 25        |
| 提示用户.....               | 26        |
| 用户维度.....               | 26        |
| 扩展用户示例.....             | 27        |
| 扩展功能.....               | 28        |
| 应用价值.....               | 28        |
| “杀手级”功能与“杀死”应用的功能.....  | 29        |
| 数据驱动的功能.....            | 30        |
| 与竞品比较.....              | 30        |
| 修改已有的功能.....            | 31        |
| 支持用户分组和角色.....          | 32        |
| 增加新服务.....              | 32        |

---

|                    |           |
|--------------------|-----------|
| 扩展功能示例.....        | 34        |
| 开发的可扩展性.....       | 34        |
| 寻找开发资源.....        | 35        |
| 开发职责.....          | 36        |
| 资源过多.....          | 36        |
| 扩展开发示例.....        | 37        |
| 影响因素检查表.....       | 37        |
| 用户检查清单.....        | 38        |
| 功能清单.....          | 39        |
| 开发者清单.....         | 41        |
| 小结.....            | 41        |
| <b>3 组件组合.....</b> | <b>43</b> |
| 通用组件.....          | 44        |
| 模块.....            | 44        |
| 路由器.....           | 46        |
| 模型/集合.....         | 50        |
| 控制器/视图.....        | 53        |
| 模板.....            | 55        |
| 应用特定的组件.....       | 56        |
| 扩展通用组件.....        | 56        |
| 识别公用数据、功能.....     | 56        |
| 扩展路由器组件.....       | 57        |
| 扩展模型/集合.....       | 58        |
| 扩展控制器/视图.....      | 59        |
| 将功能映射到组件.....      | 60        |
| 通用功能.....          | 61        |
| 特定功能.....          | 61        |
| 解构组件.....          | 62        |

|                        |           |
|------------------------|-----------|
| 维护和调试组件.....           | 62        |
| 重构复杂组件.....            | 64        |
| 可插拔的业务逻辑.....          | 64        |
| 扩展与配置.....             | 65        |
| 无状态的业务逻辑.....          | 65        |
| 组织组件代码.....            | 66        |
| 小结.....                | 67        |
| <b>4 组件的通信与职责.....</b> | <b>69</b> |
| 通信模型.....              | 69        |
| 消息传递模型.....            | 70        |
| 事件模型.....              | 70        |
| 通信数据结构.....            | 71        |
| 命名约定.....              | 71        |
| 数据格式.....              | 72        |
| 公共数据.....              | 73        |
| 可追踪的组件通信.....          | 74        |
| 订阅事件.....              | 74        |
| 全局事件日志.....            | 74        |
| 事件的生命周期.....           | 77        |
| 通信的开销.....             | 77        |
| 事件的频率.....             | 78        |
| 回调函数执行时间.....          | 80        |
| 事件复杂度.....             | 81        |
| 通信责任区.....             | 82        |
| 后端 API.....            | 82        |
| Web Socket 用于更新状态..... | 83        |
| DOM 更新.....            | 85        |
| 松耦合的通信.....            | 86        |

---

|                     |           |
|---------------------|-----------|
| 替换组件.....           | 86        |
| 应对意外事件.....         | 87        |
| 组件分层.....           | 90        |
| 事件流向.....           | 90        |
| 开发者的职责.....         | 91        |
| 构建代码思维导图.....       | 91        |
| 小结.....             | 92        |
| <b>5 寻址和导航.....</b> | <b>93</b> |
| 实现路由的方法.....        | 93        |
| Hash URI.....       | 94        |
| 传统 URI.....         | 94        |
| 路由是如何工作的.....       | 95        |
| 路由的职责.....          | 95        |
| 路由事件.....           | 96        |
| URI 的结构和模式.....     | 96        |
| 编码信息.....           | 97        |
| 设计 URI.....         | 97        |
| 将资源映射到 URI.....     | 99        |
| 手动创建 URI.....       | 99        |
| 自动生成资源 URI.....     | 99        |
| 触发路由.....           | 103       |
| 用户行为.....           | 103       |
| 重定向用户.....          | 104       |
| 路由配置.....           | 104       |
| 静态路由声明.....         | 105       |
| 注册事件.....           | 105       |
| 禁用路由.....           | 105       |
| 故障排查.....           | 108       |

|                         |            |
|-------------------------|------------|
| 路由器冲突.....              | 108        |
| 记录初始配置.....             | 110        |
| 记录路由事件.....             | 110        |
| 处理非法资源的状态.....          | 110        |
| 小结.....                 | 111        |
| <b>6 用户偏好和默认设置.....</b> | <b>113</b> |
| 偏好类型.....               | 113        |
| 地区.....                 | 113        |
| 行为.....                 | 114        |
| 外观.....                 | 115        |
| 支持地区.....               | 115        |
| 决定支持哪些地区.....           | 115        |
| 维护地区.....               | 116        |
| 设置地区.....               | 116        |
| 选择地区.....               | 117        |
| 存储地区偏好.....             | 117        |
| URI 中的地区.....           | 118        |
| 通用组件配置.....             | 118        |
| 选择配置的值.....             | 119        |
| 存储和硬编码默认值.....          | 119        |
| 对后端的影响.....             | 120        |
| 加载配置值.....              | 121        |
| 配置行为.....               | 122        |
| 启用和禁用组件.....            | 122        |
| 改变数量.....               | 123        |
| 改变顺序.....               | 124        |
| 配置通知.....               | 126        |
| 行内选项.....               | 126        |



---

|                         |            |
|-------------------------|------------|
| 改变外观.....               | 127        |
| 主题工具.....               | 127        |
| 选择一个主题.....             | 128        |
| 单独的样式偏好.....            | 128        |
| 性能影响.....               | 128        |
| 可配置地区的性能.....           | 129        |
| 可配置行为的性能.....           | 129        |
| 可配置主题的性能.....           | 132        |
| 小结.....                 | 132        |
| <b>7 加载时间和响应速度.....</b> | <b>135</b> |
| 组件构件.....               | 135        |
| 组件依赖.....               | 135        |
| 构建组件.....               | 136        |
| 加载组件.....               | 137        |
| 加载模块.....               | 137        |
| 懒惰的模块加载.....            | 138        |
| 模块加载的延迟.....            | 139        |
| 通信瓶颈.....               | 141        |
| 减少间接引用.....             | 141        |
| 分析代码.....               | 143        |
| 组件优化.....               | 145        |
| 维护状态的组件.....            | 145        |
| 处理副作用.....              | 146        |
| DOM 渲染技术.....           | 148        |
| API 数据.....             | 150        |
| 加载延迟.....               | 150        |
| 处理大数据集.....             | 151        |
| 优化运行时组件.....            | 152        |
| 小结.....                 | 153        |