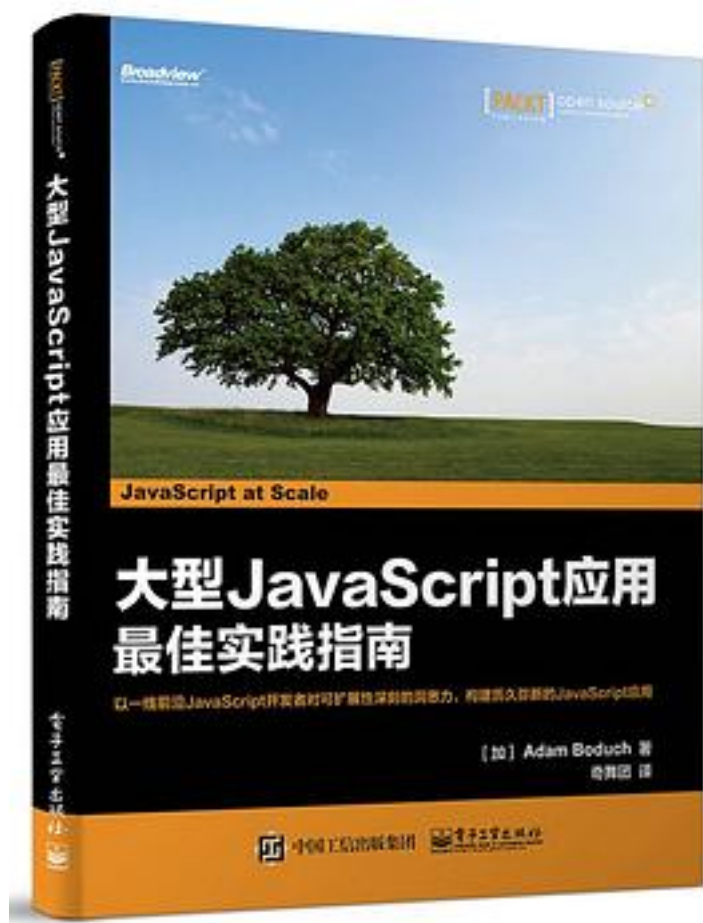


大型JavaScript应用最佳实践指南



[大型JavaScript应用最佳实践指南_下载链接1](#)

著者:【加】Adam Boduch

出版者:电子工业出版社

出版时间:2017-2

装帧:平装

isbn:9787121307065

《大型JavaScript应用最佳实践指南》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。《大型JavaScript应用最佳实践指南》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。

作者介绍:

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。

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。

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>。

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 kwarg.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>)（高性能前端工作流系统）。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。

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
- 2 可扩展性的影响因素 19
- 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
- 3 组件组合 43
- 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
- 4 组件的通信与职责 69
 - 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
- 5 寻址和导航 93
 - 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URI99
 - 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
- 6 用户偏好和默认设置113
 - 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
- 7 加载时间和响应速度135
 - 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
- 8 可移植性和测试 155
 - 与后端解耦155
 - 模拟后端API 155
 - 前端入口 156
 - 模拟工具 157
 - 生成模拟数据集 158
 - 执行操作 159
 - 功能设计过程159
 - 设计API 160
 - 实现模拟数据 160
 - 实现功能 161
 - 协调模拟数据与真实数据 162
 - 单元测试工具163
 - 框架自带的工具 163
 - 独立的单元测试工具 164
 - 工具链和自动化 165
 - 测试模拟场景166
 - 模拟API 和测试固件 166
 - 场景生成工具 167
 - 端到端测试和持续集成168
 - 小结169
- 9 缩小规模171
 - 扩展限制171
 - JavaScript 文件体积 172
 - 网络带宽 173
 - 内存消耗 175
 - CPU 消耗177
 - 后端能力 179
 - 互相矛盾的功能180
 - 重叠的功能 181
 - 冗余的功能 182
 - 用户需求 182
 - 设计失效183
 - 多余的组件 184
 - 低效的数据处理 186
 - 过度创建标记 190
 - 应用组合191
 - 功能的启动 191
 - 新功能的影响 192
 - 重要的库 192
 - 小结193
- 10 处理错误195
 - 快速失效195
 - 使用质量约束 196
 - 提供有意义的反馈 196
 - 当无法快速失效时…… 197
 - 容错198
 - 区分关键行为 198

探测和控制错误行为 199
禁用出错组件 202
优雅地降级功能 203
故障恢复 204
重试失败操作 204
重启组件 207
用户手动干涉 208
当我们无法从故障中恢复…… 209
性能和复杂度 210
异常处理 210
状态检查 211
通知其他组件 211
记录日志和调试 212
有意义的错误日志 212
为潜在故障发出警告 213
通知和指导用户 214
改进架构 214
记录错误场景 215
改进组件分类 215
复杂导致出错 216
小结 216
• • • • • ([收起](#))

[大型JavaScript应用最佳实践指南 下载链接1](#)

标签

JavaScript

前端

计算机科学

javascript

技术

性能

2017

评论

题材、内容都是好的. 只是不知道是翻译的问题还是什么,读起来很不“顺”.

没有细读 主要是读不下去 可能作者确实有很丰富的js经验 不过随着各种js框架的兴起 比如vue.js的兴起 这种大型的js应用很可能不再是这个样子了

[大型JavaScript应用最佳实践指南_下载链接1](#)

书评

[大型JavaScript应用最佳实践指南_下载链接1](#)