O'REILLY®

TURING 图灵程序设计丛书

同构JavaScript应用开发

Building Isomorphic JavaScript Apps

[美] Jason Strimpel & Maxime Najim 著

张俊达 译

中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

同构JavaScript应用开发

Building Isomorphic JavaScript Apps

[美] Jason Strimpel, Maxime Najim 著
张俊达 译

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®
O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社
北 京

图书在版编目（CIP）数据

同构JavaScript应用开发 /（美）杰森·史特林贝尔（Jason Strimpel），（美）马克西姆·纳吉姆（Maxime Najim）著；张俊达译. -- 北京：人民邮电出版社，2017.10
（图灵程序设计丛书）
ISBN 978-7-115-46868-0

Ⅰ. ①同… Ⅱ. ①杰… ②马… ③张… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2017)第223584号

内 容 提 要

本书将向你展示如何构建和维护属于自己的同构 JavaScript 应用。全书分为三部分，第一部分描绘不同种类的同构 JavaScript 的轮廓，第二部分介绍关键概念，第三部分提供业界同行的解决方案案例。通过阅读本书，你将了解到这种应用架构日益流行的原因，并能将其运用于解决关键的业务问题，如页面加载速度和 SEO 兼容性。

本书适合对同构 JavaScript 感兴趣的 Web 开发人员。

◆ 著　[美] Jason Strimpel, Maxime Najim
译　张俊达
责任编辑　朱　巍
执行编辑　夏静文
责任印制　彭志环

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路11号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：11
字数：260千字　2017年10月第1版
印数：1-3 000册　2017年10月北京第1次印刷
著作权合同登记号　图字：01-2017-4856号

定价：49.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

站在巨人的肩上

Standing on Shoulders of Giants

iTuring.cn

站在巨人的肩上

Standing on Shoulders of Giants

iTuring.cn

版权声明

O'Reilly Media, Inc.介绍

O'Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O'Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 *Make* 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版、在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar 博客有口皆碑。”

——*Wired*

“O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——*Business 2.0*

“O'Reilly Conference 是聚集关键思想领袖的绝对典范。”

——*CRN*

“一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——*Irish Times*

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野，并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去，Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——*Linux Journal*

前言

Jason Strimpel

我在多年前就开始了 Web 开发的职业生涯，当时我在加州大学圣地亚哥分校担任行政助理。我的工作职责之一就是维护部门网站。那个年代的 Web 开发还是站点管理员、表格式布局和 CGI 程序的天下，且网景浏览器仍然是浏览器领域中的佼佼者。当时，我的专业知识还有很多不足，也缺乏经验。我清晰地记得，当时我很担心，如果我发送的电子邮件中存在拼写错误，收件人就会看到它们被红色下划线标记出来，正如我看到他们的拼写错误时一样。幸运的是，我的上司很耐心，他说我的邮件中并没有拼写错误，并向我传授了关于网站开发的许多要点。

转眼 15 年过去了，如今我在各大会议上发表演讲，管理开源项目，合著图书，成了这个领域的“专家”。有时候我会问自己：“我是如何取得今天的成就的？”这个问题的答案绝不仅仅是任由时间一天天地流逝——至少不完全是虚度时光。

为什么需要同构JavaScript

到目前为止，我的 Web 开发之路的最后一站是沃尔玛实验室的平台团队，我的同构 JavaScript 探索之旅就是从这里开始的。刚开始在沃尔玛工作时，我被分配到了一个负责开发 Web 新框架的团队。这个框架是从零开始开发的，其目标是支撑面向公众的大型网站。除了满足这类网站的最低要求，支持 SEO 并优化网页加载速度之外，保持 UI 工程师（包括我在内）的开发愉悦感和高效率也非常重要。很明显，为了更好地实现 UI 工程师的目标，我们的首选是基于现有的**单页面应用**（Single-Page Application，SPA，https://en.wikipedia.org/wiki/Single-page_application）技术进行扩展。但 SPA 模型有一个问题，它不能很好地支持我们的最低要求（详情请参见下文中的“完美风暴：一个极其平常的故事”和 1.2.3 节中的“单页面 Web 应用”），所以我们最终选择采用同构的方式。以下是这样做的理由。

- 对于同一个渲染周期，客户端和服务器端可以使用同一套代码。这意味着不需要重复劳动，可以在降低界面开发与维护成本的同时提高团队开发速度。
- 在服务器端渲染一份初始的 HTML 代码后，用户会感觉网页的加载速度更快，这是因为用户可以在浏览器中先看见首屏渲染的内容，而无须等待应用资源的加载和数据抓取完成。在网络延迟比较严重的环境中，这种改进页面预加载的方式尤为重要。
- 同构应用支持 SEO，因为同构应用使用的 URL 不包含 # 号片段。此外，在那些不支持 History API 的浏览器中，它可以（对后续的每次请求）优雅地降级为服务器端渲染的方式。
- 在支持 History API 的浏览器中，同构应用使用了 SPA 模型的分布式渲染，因此后续请求可以减轻服务器负载。
- 无论是在服务器端还是在客户端，UI 工程师都可以完全掌控界面（https://www.nczonline.net/blog/2013/10/07/node-js-and-the-new-web-front-end/），而且同构应用在前后端之间划分了明确的界限，这有助于降低操作成本。

以上是我们团队走上同构 JavaScript 之路的主要原因。但在详细介绍同构 JavaScript 之前，先交代一些背景知识是很有必要的，这可以帮助你理解我们今天所处的具体环境。

平台的演进

Web 技术的快速演进令人难以想象。当初，CSS 和 JavaScript 技术被引入浏览器的目的是提供一种交互模型和关注点分离。你还记得曾有无数的文章提倡结构、样式和行为分离吗？即便在引入了这些技术之后，应用的架构也并没有发生太大变化。文档通过 URI 的形式进行请求，浏览器解析返回内容后，再进行渲染。唯一的不同之处就是 JavaScript 让界面变得更丰富了一点。直到微软公司引入一项被称为 `XMLHttpRequest`（https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest）的新技术，Web 才在这项技术的催化下演进为应用平台。

Ajax：应用平台的崛起

尽管很多前端工程师对微软公司及其 IE 浏览器嗤之以鼻，但他们仍然应当对微软怀有一份感激之情。如果没有微软，前端工程师未必能达到今天的职业高度。如果 `XMLHttpRequest` 技术没有出现，Ajax 技术也就不会诞生；如果没有 Ajax，就不会有这么多修改页面内容的需求，我们也就没有必要使用 jQuery（https://jquery.com/）。你猜接下来会发生什么呢？我们就不会有大量的前端 MV* 库可供选择，单页面应用的模式也不会出现，进而 History API 也不会出现。所以，下一次抱怨 IE 浏览器给你带来麻烦时，请你务必对微软作出客观评价，毕竟微软改变了历史进程，为今天的 Web 应用奠定了基础，还为你提供了一个锻炼思维的场所。

Ajax：技术债的积累

虽然 Ajax 技术影响了 Web 平台的发展进程，但它也以技术债的形式造成了一些破坏性的影响。Ajax 模糊了以前清晰定义的模型。过去，当用户导航到一个新页面或者提交表单数据时，浏览器只需依次发送请求、取得响应并解析完整的文档流。当 Ajax 成为 Web 开发的主流技术后，这种方式完全改变了。现在，工程师不需要向服务器请求重新获取整份文档，只需要根据用户的请求来判断是加载更多数据还是返回另一个视图。这意味着应用可以仅更新页面中的某个区域。这种特性大大优化了客户端和服务器端的性能，同时明显改善了用户体验。不幸的是，客户端的应用架构几乎变得不存在了，而本来负责处理视图层的那些工程师并没有应对这种范式转变的经验。日积月累，这些因素将应用的维护工作变成了噩梦。由于模型定义变得模糊，Web 开发从此经历了一段艰难成长的时期。

完美风暴：一个极其平常的故事

想象一下如下情景：你所在的团队负责维护一个商业应用的商品页面。在该页面的右侧有一个用于显示用户点评的轮播组件，用该组件可以翻页。当用户点击翻页按钮时，客户端会改变 URI 并向服务器发起请求以重新获取整个商品页面。这种低效的做法会让身为工程师的你感到非常苦恼。你认为没有必要刷新整个页面并调用一个重新渲染页面的数据请求，真正需要获取的只是下一页评论的 HTML 内容。好在你一直在跟踪行业内最新的技术发展，准备尝试使用最近学习的 Ajax 来解决这个问题。你收集了一些关于 Ajax 的概念证明，并向上司推荐了这项技术。这时候的你看起来就像巫师一样。像其他技术一样，这项技术最终会正式投入到生产环境中。

你对目前的成果感到非常满意，直到后来了解到一种新的数据交换格式。这种称为 JSON 的格式受到了 JavaScript 的非官方代言人 Douglas Crockford（http://www.crockford.com/）的极力推荐。你马上就觉得目前的实现不够完美了。第二天，你使用一种称为**微模板**（micro-templating，http://ejohn.org/blog/javascript-micro-templating/）的技术编写了一份新的概念证明，并再次向上司推荐。这项技术同样受到了好评并再次投入到生产环境中。

此时普通的工程师已经将你奉为神明。这时候，代码在审查阶段被发现有 bug。上司找到你并让你修复 bug，因为这段逻辑是你实现的。你审查了代码，并向上司宣称 bug 在服务器端的渲染中。然后你需要对使用两套渲染方案的原因进行一番解释。在解释完为什么 Java 不能在浏览器中运行后，你还得向上司保证这种实现是值得的，因为这样可以大大提升用户体验。这个问题像烫手的山芋那样被传来传去，直到最后才得以解决。

尽管违背了 DRY（don't repeat yourself，不要重复你自己）原则，但你依然被誉为专家。你的实现模式逐渐被大家学习、模仿，进而充斥着整个代码库。但随着这种模式的渗透，意料之外的事情发生了。bug 的数量开始不断上升，开发人员开始害怕因修改代码而造成

的回归问题。目前欠下的技术债甚至比国家的财政赤字还要严重。工程经理和开发人员开始互相推卸责任。应用变得非常脆弱，公司也因此难以应对市场的快速变化。你感到一种强烈的罪恶感。幸好，你发现了一种叫作单页面应用的新模式……

客户端架构的救赎

近段时间，你阅读了一些文章，内容主要是人们对于前端架构的缺失而感到沮丧。这些人通常会将责任归咎于 jQuery，尽管这个库本来只是对 DOM 操作的封装（façade）。好在业界有人遇到了和你一样的困境，并且没有在其他不明真相的人的评论中停止自己的脚步。其中一个人就是 Backbone（http://backbonejs.org/）框架的作者 Jeremy Ashkenas（https://github.com/jashkenas）。

你开始了解 Backbone，阅读相关文章，并且深感兴趣。Backbone 将应用逻辑从数据检索中抽离出来，并将界面代码整合为单一语言和运行时，因此可以有效减少服务器端的压力。“找到了！”你在心中欢欣鼓舞地喊道。这个框架将解决我们遇到的所有问题。你又提出了一份新的概念证明，并开始实施。

在我们访问时发生了什么

你很快就被称为救世主。这种新的 SPA 模式在公司范围内被广泛接受。bug 的数量开始减少，工程师又重拾了信心。交付代码时的恐惧感几乎已经消失。这个时候，负责产品的同事找到你，并告知你自从实现 SPA 模式之后，网站的访问量下降了。你得想办法处理 # 号片段带来的问题了。经过一番详尽的研究后，你确定问题出在搜索引擎没有考虑 URI 中的 `window.location.hash` 部分，而 `Backbone.Router` 用这部分来创建可跳转、可收藏书签、可分享的页面视图。因此，当搜索引擎爬取这个应用时，没有任何可以收录的内容。现在你面临的形势更加严峻了，因为这个问题会对商品销量产生直接影响。因此，你再次开启了调研与开发的循环。结果你发现有两种方案可供选择：第一种方案是运转新的服务器，模拟 DOM 操作以运行客户端应用，并将搜索引擎重定向到这些新服务器上；第二种方案是付费让其他公司为你提供解决方案。除了 SPA 实现给公司带来的损失外，这两种方案都需要支付额外成本。

同构JavaScript：一个美好的新世界

上述这个故事综合了我的个人经历以及我从其他工程师那里目睹或听说的故事。如果你也曾为某个 Web 应用付出很多时间，我相信你也会有类似的经历和感受。故事当中的某些问题已经成为了历史，但部分问题依然存在。此外还有一些问题没有明说，例如页面加载速度有待优化以及缺少感知渲染。如果将路由的响应与渲染的生命周期合并为一个通用的代

码库，并同时支持在客户端和服务器端运行，应该就可以解决上述这些问题以及其他潜在问题。这就是同构 JavaScript 的意义所在。同构 JavaScript 应用是整合两种架构，以创建易于维护的、更好的用户体验。

未来的路

本书的主要目的是提供实现同构 JavaScript 所需的基础知识，帮助你理解业界现有的同构 JavaScript 解决方案。本书旨在提供足够多的信息，让你在实际中判断同构 JavaScript 是否为可行的解决方案，同时介绍业内最先进的解决方案，避免你重复造轮子。

第一部分是对这个主题的介绍。首先，详尽地介绍现有的几种 Web 应用架构，内容涵盖同构 JavaScript 的基本原理和用例，如 SEO 支持和提升页面的感知加载速度。然后，概述不同种类的同构 JavaScript 应用，如实时应用与类似 SPA 模式的应用。此外，还介绍了同构应用方案的组成部分，其中包括提供环境 shim 和抽象的实现，以及真正与环境无关的实现。该部分将为第二部分奠定代码基础。

第二部分将主题分解为关键概念，这些概念在大部分同构 JavaScript 解决方案中被普遍使用。每种概念都无须依赖现有的库［如 React（https://facebook.github.io/react/）、Backbone（https://facebook.github.io/react/） 或 Ember（https://facebook.github.io/react/）］ 即可实现。这样做是为了避免将概念和某种特定的解决方案混淆。

第三部分将会介绍业内的专家是如何在他们的解决方案中作出权衡的。

排版约定

本书使用了下列排版约定。

- **黑体**
  表示新术语或重点内容。
- 等宽字体（`constant width`）
  表示程序片段，以及正文中出现的变量、函数名、数据库、数据类型、环境变量、语句和关键字等。
- 加粗等宽字体（**`constant width bold`**）
  表示应该由用户输入的命令或其他文本。
- 等宽斜体（*`constant width italic`*）
  表示应该由用户输入的值或根据上下文确定的值替换的文本。

该图标表示提示或建议。

该图标表示一般注记。

该图标表示警告或警示。

代码示例

补充材料（代码示例、练习等）可以从 https://github.com/isomorphic-javascript-book 下载。

本书是要帮你完成工作的。一般来说，如果本书提供了示例代码，你可以把它用在你的程序或文档中。除非你使用了很大一部分代码，否则无须联系我们获得许可。比如，用本书的几个代码片段写一个程序就无须获得许可，销售或分发 O'Reilly 图书的示例光盘则需要获得许可；引用本书中的示例代码回答问题无须获得许可，将书中大量的代码放到你的产品文档中则需要获得许可。

我们很希望但并不强制要求你在引用本书内容时加上引用说明。引用说明一般包括书名、作者、出版社和 ISBN。比如："*Building Isomorphic JavaScript Apps* by Jason Strimpel and Maxime Najim (O'Reilly). Copyright 2016 Jason Strimpel and Maxime Najim, 978-1-491-93293-3"。

如果你觉得自己对示例代码的用法超出了上述许可的范围，欢迎你通过 permissions@oreilly.com 与我们联系。

Safari® Books Online

Safari Books Online（http://www.safaribooksonline.com）是应运而生的数字图书馆。它同时以图书和视频的形式出版世界顶级技术和商务作家的专业作品。技术专家、软件开发人员、Web 设计师、商务人士和创意专家等，在开展调研、解决问题、学习和认证培训时，都将 Safari Books Online 视作获取资料的首选渠道。

对于组织团体、政府机构和个人，Safari Books Online 提供各种产品组合和灵活的定价策略。用户可通过一个功能完备的数据库检索系统访问 O'Reilly Media、Prentice

Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology 以及其他几十家出版社的上千种图书、培训视频和正式出版之前的书稿。要了解 Safari Books Online 的更多信息，我们网上见。

联系我们

请把对本书的评价和问题发给出版社。

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网页，你可以在那儿找到本书的相关信息，包括勘误表、示例代码以及其他信息。本书的网页地址是：

http://shop.oreilly.com/product/0636920042846.do

对于本书的评论和技术性问题，请发送电子邮件到：

bookquestions@oreilly.com

要了解更多 O'Reilly 图书、培训课程、会议和新闻的信息，请访问以下网站：

http://www.oreilly.com

我们在 Facebook 的地址如下：

http://facebook.com/oreilly

请关注我们的 Twitter 动态：

http://twitter.com/oreillymedia

我们的 YouTube 视频地址如下：

http://www.youtube.com/oreillymedia

致谢

Jason Strimpel

首先，我要感谢妻子 Lasca 的耐心与支持。我每天都被你的智慧、幽默、慈悲和爱所包围。感谢你选择了和我一起生活，这是我莫大的荣幸。是你让我成为了更加优秀的人，感谢你对我的爱。

其次，我想感谢与我合著本书的同事 Maxime。没有你的激情、知识、想法和专业知识，我的努力将大打折扣。你对于软件架构的洞察力每天都能给予我灵感。非常感谢你。

感谢我的编辑 Allyson。你提出的观点、疑问和修改意见让我的写作增色不少。感谢你。

最后，我要感谢第三部分中所有内容的贡献者，感谢你们在百忙之中抽空和读者分享你们的故事。这一部分展示了同构 JavaScript 的深度以及解决方案的多样性。你们用独特的解决方案证明了每一位工程师都是富有创造力的。感谢你们。

Maxime Najim

首先要感谢我的家人——我的妻子 Nicole，以及我的孩子 Tatiana 和 Alexandra，感谢他们在本书的编写和出版过程中给予我的支持与鼓励。

我也永远感激 Jason 邀请我和他合著本书，能与你合作让我感到非常荣幸。这是一个千载难逢的机会，是你的智慧、知识和努力把这个机会变成了现实。我对你感激不尽。同样，非常感谢我们的编辑 Allyson 在整个过程中提供了宝贵的支持和建议。

最后，我要特别感谢第三部分内容的贡献者在百忙中抽空与我们分享他们的故事和经历。非常感谢你们！

电子书

扫描如下二维码，即可购买本书电子版。

目录

第二部分 构建第一个应用