O'REILLY®

/theory/in/practice

JavaScript之美

Beautiful JavaScript

Anton Kovalyov 编
杜春晓 司韦韦 译

中国电力出版社

JavaScript之美

Anton Kovalyov 编

杜春晓 司韦韦 译

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

O'Reilly Media, Inc. 授权中国电力出版社出版

中国电力出版社

图书在版编目（CIP）数据

JavaScript 之美 / (美) 安顿·科瓦诺夫（Anton Kovalyov）编；杜春晓, 司韦韦译. — 北京：中国电力出版社，2017.12

书名原文：Beautiful JavaScript

ISBN 978-7-5198-1364-2

I. ①J… II. ①安… ②杜… ③司… 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288815号

北京市版权局著作权合同登记 图字：01-2017-6362号

出版发行：中国电力出版社
地　址：北京市东城区北京站西街19号（邮政编码100005）
网　址：http://www.cepp.sgcc.com.cn
责任编辑：刘 炽（liuchi1030@163.com）
责任校对：马 宁
装帧设计：Susan Thompson，张 健
责任印制：蔺义舟

印　刷：三河市百盛印装有限公司
版　次：2017年12月第一版
印　次：2017年12月北京第一次印刷
开　本：787毫米×980毫米　16开本
印　张：10.5
字　数：198千字
印　数：0001—3000册
定　价：48.00元

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

译者序

JavaScript是一个传奇。Brendan Eich仅用10天时间就设计出了它的第1个版本。仓促之间，它存在各种各样的缺陷也就不足为奇。出乎意料的是，它的不足给众多开发者留下了广阔的发挥空间。加之编写、运行JavaScript都很简单，只要有文本编辑器和浏览器即可。就这样，无数本没有机会进入程序设计领域的人们加入JavaScript社区，其中很多开发者都是自学成才。一门存在各种不足的语言，一个多数成员靠自学成才的社区，演绎了一段不可思议的神话，JavaScript多年来稳坐Web语言之主的宝座，与HTML、CSS并称Web开发的三剑客。我不由地想起孔子所说的“犁牛之子”，想起他的“吾少也贱，故多能鄙事。”JavaScript和它的开发者正是这种精神的集中体现。

本书由15位JavaScript专家写成，每人各写一章，分享他们认为JavaScript美在何处。跟随大师的脚步，我们将学习到代码复用的不同方法、模板编译、词法和句法分析、异常捕获、Node.js事件循环；还将学到多名开发人员用JavaScript开发，应注意哪些问题；除了这些具体的问题，作者还探讨了不同的抽象层次、编程范式和代码风格等。作者通过对以上内容介绍，帮助我们认识到JavaScript的魅力之所在。这些内容的落脚点在于，帮助我们编写出高效、安全、易于他人理解和维护的高质量代码。

感谢各位作者，名单太长，这里就不一一列出了，其中尤其感谢Graeme Roberts、Marijn Haverbeke和Rick Waldron，感谢三位的答疑解惑。感谢王海霞和邵有生，我向二人请教过问题。最后，感谢家人和朋友对我的支持。

本书由杜春晓（新浪微博：@宜_生）和司韦韦（前敦煌网前端工程师）共同翻译。内容包罗万象，由于时间仓促，书中翻译错误、不当和疏漏之处在所难免，敬请读者批评指正。

译者

目录

前言

函数是第一等公民，句法像Java，继承用原型实现，(+"")等于0，这就是JavaScript。它可以说是世上争议和误解最多的编程语言。开发这门语言只用了10天，因此它存在大量缺陷，有很多不够优雅的地方。自它面世之后，很多开发者就一直尝试取代它作为Web语言之主的地位。但时至今日，该语言和围绕它形成的生态系统仍在蓬勃发展。JavaScript是当今最流行、Web平台开发的真正语言。是什么令JavaScript如此特殊？为什么这门仓促设计的语言取得成功，而其他语言却失败了？

我相信JavaScript（和Web）之所以能够存活下来，是因为它的无处不在，个人计算机没有安装JavaScript解释器，这种情况几乎不可能存在，以及它从混乱中求发展，化压力为动力，努力提升自我的能力。

JavaScript不同于其他语言，它将各种不同的人聚集于Web开发这个大平台。只要装有文本编辑器和Web浏览器，人们就可以动手开发JavaScript，很多人也确实是这么入行的。它的表现力、有限的标准库促使开发者多方尝试，将其发挥到极限。人们不仅用它开发网站和应用，还用它编写各种库，开发能够编译回JavaScript的编程语言。这些库彼此存在竞争关系，解决问题的方法往往相冲突。JavaScript生态系统一片混乱，但却也充满生机。

过去人们用JavaScript编写的很多库和语言，现已被人忘记。然而，开发者的最佳想法（那些证明了他们自己的实力并经得住时间检验的想法）被这门语言所吸收。

它们成功进入 JavaScript 的标准库，固化为它的句法。它们使这门语言更加优秀。

曾经有很多语言和技术以取代 JavaScript 为使命。它们非但没有成功，反而不情愿地充当了 JavaScript 的鞭策者。每当意欲取代 JavaScript 的新语言或系统出现后，浏览器厂商就会想方设法使其变得更快、强大和健壮。这些优秀的想法一次次融入到 JavaScript 语言的新版本之中，不好的想法则被抛弃。与 JavaScript 相竞争的这些技术，不仅没能取代它，反而使其更强大。

如今，JavaScript 的受欢迎程度超乎想象。这一局面会持续下去吗？我无法预测 5 年、10 年之后，它是否依然如此受欢迎，但它以后表现如何真的没那么重要。因为对我而言，统观所有语言，未毁之于自身的缺陷，反逆势为之而终获繁荣，并能将各种不同背景的人引入计算机编程圣境之中的，JavaScript 是一个绝佳的例子。

关于本书

本书作者熟稔 JavaScript。每位作者各写一章，他们均是各自领域的专家。他们根据自己的侧重点，介绍了 JavaScript 不同方面的特点，其中有些特点你只有编写大量代码，在试错的过程中才能发现。随着阅读的深入，你将发现这些挚爱着 JavaScript 的推动者到底喜欢它的哪些点。

阅读本书，你还将学到很多知识。我确实受益颇多。但是，请不要误将本书当作教程，因为它比教程更宏大。某些章节是对人们一贯认识的挑战，作者向我们展示了即使是最令人恐惧的特征，也可以成为很有用的工具。有几位作者展示了 JavaScript 可用来表现自我，俨然是一种艺术形式，而其他作者则分享了成百上千名开发者在代码库中使用 JavaScript 时，所需的注意事项和最佳实践。有些作者分享个人经历，其他作者则着眼于未来。

本书的各章内容在写作上没有固定模式可循，甚至有一章内容极为幽默、诙谐，这一章是我有意安排的。我尽可能地给予作者充分的自由，我想看看他们究竟能写出怎样的佳作。他们果然不负所望，内容之精彩令人难以置信。他们最终写出了一本极具 JavaScript 风格的书，每一章都反映了作者的风貌。

排版约定

本书在排版上遵循以下约定：

斜体（Italic）

表示新术语、URL、邮件地址、文件名和文件扩展名。

`等宽字体（Constant width）`

程序及段落中表示变量、函数名、数据库、数据类型、环境变量、声明和关键字等程序中的元素，使用等宽字体。

小贴士

该元素表示小贴士或建议。

注意

该元素表示一条注意事项。

使用示例代码的注意事项

配套材料（示例代码、练习等）请从 *https://github.com/oreillymedia/beautiful_javascript* 下载。

本书是为了帮助你更好地完成工作。一般来讲，书中的示例代码，你用于自己的项目和文档，无需联系并获得我们的许可，但大量复制我们的代码另议。例如，编写程序，使用书中的多处代码，无需我们授权，但出售或传播用 O'Reilly 图书示例代码制作的 CD-ROM 光盘，则需要我们授权。引用本书内容回答问题，或引用示例代码，无需授权，但在你的产品文档中，大量使用本书的示例代码，则需要授权。

如果你能添加内容的出处，我们将非常感激，当然这不是必须的。出处通常要标明书名、作者、出版社和 ISBN 号信息。例如：“*Beautiful JavaScript*, edited by Anton Kovalyov（O'Reilly）. Copyright 2015 Anton Kovalyov, 978-1-449-37075-6”。

如果你觉得示例代码的使用方式可能不当或在我们上面列出的许可范围之外，请联系我们确认，邮箱是 *permissions@oreilly.com*。

Safari® Books Online

Safari Books Online 是一个按需服务的数字图书馆，以图书和视频形式提供全世界科技和商业领域顶级作者创作的专业内容。

技术专家、软件开发者、Web 设计师、商业和创意人士将 Safari Books Online 作为研究、解决问题、学习和认证培训使用的首要资源。

Safari Books Online 为企业、政府、教育机构和个人提供多种方案和定价策略。

我们向会员开放成千上万本图书、培训视频和待正式出版的手稿。我们用一个具备强大检索功能的数据库存储资源。这些资源来自几百家出版机构，其中包括 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 和 Course Technology。更多信息请访问 *https://www.safaribooksonline.com*。

联系方式

欢迎将你对本书的任何意见和问题寄给我们，地址如下：

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

我们为本书做了一个网页，把勘误信息、示例代码和其他附加信息列在上面。地址是 *https://github.com/oreillymedia/beautiful_javascript*。

你对本书的任何意见或技术方面的问题，都可以将其发送至 *bookquestions@oreilly.com*。

关于我们的图书、课程、会议和新闻的更多信息，请访问我们的网站 *http://www.oreilly.com*。

欢迎关注我们的 Facebook 账号：*http://facebook.com/oreilly*。

欢迎关注我们的 Twitter 账号：*http://twitter.com/oreillymedia*。

欢迎观看我们上传到 YouTube 网站上的视频：*http://www.youtube.com/oreillymedia*。

第 1 章

美丽的 mixin

Angus Croll

> 实际上并不是什么大不了的问题，开发者却喜欢整出一个过于复杂的解决方案。
>
> —— Thomas Fuchs

项目开始后，代码逐渐多起来。于是，我们将其封装为函数以便复用。随着项目的推进，封装的函数多得不能再多了。于是，我们寻找复用函数的方法。JavaScript 开发者往往不遗余力地使用“合适”的复用技术。但有时过于追求去做正确的事情，却会错过近在眼前的美景。

类继承

JavaScript 开发者，若学过 Java、C++、Objective-C 和 Smalltalk，编写 JavaScript 代码时，对于用类这种层级结构组织代码，心存近乎宗教般的虔诚。然而人们并不擅长分类。从抽象的超类反向派生出实际的类和类具有的各种行为，不符合常人的认知过程且存在限制，必须先创建超类，其他类才能对其扩展。而更接近对象本质的类，更具一般性，抽象程度更高，我们对具体的子类有了更多了解之后，再去定义类将更加容易。此外，从一般到具体的派生方法，类之间存在紧密耦合问题，比如一类完全定义在另一类之上，生成的模型可能过于死板、脆弱和荒谬（“Button 是 Rectangle 还是 Control 的子类？我跟你讲，`Button` 继承自 `Rectangle`，`Rectangle` 继承自 `Control`……不，等会……”）。若不尽早理顺各类之间的继承关系，系统将永远受累于一组存在缺陷的关系。出于偶然或凭借天分，我们也许偶尔也能理顺它

们之间的关系，但即使最小化的树结构所表示的心智模式，对我们而言通常也过于复杂，我们无法快速理解它。

类继承适合为现有的、易于理解的层级结构建模，明确地声明 `FileStream` 是一种 `Input Stream`，这没有问题。但如果主要动机是函数复用（通常如此），那么为实现类继承而建立起来的层级结构，很快就会变为充斥着毫无意义的子类型的迷宫，令人感到棘手，并且还会增加代码的冗余程度和维护代码逻辑的难度。

原型

大多数行为是否能映射到客观上“正确”的类别，这点仍值得怀疑。并且，主张类继承的一派也的确遭遇到了一伙狂热分子的挑战，他们同是 JavaScript 的忠诚捍卫者。他们宣称 JavaScript 是一种基于原型而不是面向对象的语言，任何带有“类”字样的方法根本不适用于 JavaScript。但“原型”的含义是什么？原型和类有着怎样的区别？

用通用的编程术语来讲，原型是指为其他对象提供基本行为的对象。其他对象可在此基础上扩展基本行为，加入个性化行为。该过程也称为有差异的继承，有别于类继承的是它不需要明确指定类型（静态或动态），从形式上而言，它也不是在一种类型的基础上定义其他类型。类继承的目的是复用，而原型继承则不一定。

> 一般而言，开发人员使用原型，通常不是为了分类，而是为了利用原型和对象之间的相似性。
>
> —— Antero Taivalsaari，诺基亚研究中心

JavaScript 的每个对象均指向一个原型对象并继承其属性。JavaScript 的原型是实现复用的好工具：一个原型实例可以为依赖于它的无数实例定义属性。原型还可以继承自其他原型，从而形成原型链。

到此为止还好理解。但 JavaScript 仿效 Java，将 `prototype` 属性绑定到构造器，其结果是多个层级的对象继承通常需要链接构造器和原型来实现。在介绍 JavaScript 之美的书里，加入 JavaScript 原型链的标准实现方法，可能会令读者望而生畏，简单来讲，为了定义继承者的初始化属性，而为基础原型创建一个新实例的做法，既不优雅也不直观。另一种方法 [手动复制不同原型的属性，改动构造器（constructor）的属性，以此来伪造原型继承] 则更不可取。