HZ BOOKS
华章科技

EFFECTIVE
系列丛书

PEARSON

JavaScript标准化委员会著名专家撰写，亚马逊五星级畅销书，JavaScript语言之父、Mozilla CTO Brendan Eich联合数位专家鼎力推荐

作者将在JavaScript标准化委员会工作和实践的多年经验浓缩为极具实践指导意义的68个有效方法，深刻辨析JavaScript的特性和内部运作机制，以及编码中的陷阱和最佳实践

Effective JavaScript
68 Specific Ways to Harness the Power of JavaScript

Effective JavaScript

编写高质量JavaScript代码的68个有效方法

（美） David Herman 著
黄博文 喻杨 译

机械工业出版社
China Machine Press

Effective JavaScript
68 Specific Ways to Harness the Power of JavaScript

Effective JavaScript

编写高质量JavaScript代码的68个有效方法

（美） David Herman 著
黄博文 喻杨 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Effective JavaScript：编写高质量 JavaScript 代码的 68 个有效方法 /（美）赫尔曼（Herman, D.）著；黄博文，喻杨译 . —北京：机械工业出版社，2013.11
（Effective 系列丛书）
书名原文：Effective JavaScript: 68 Specific Ways to Harness the Power of JavaScript

ISBN 978-7-111-44623-1

Ⅰ. E… Ⅱ. ①赫… ②黄… ③喻… Ⅲ. Java 语言 – 程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2013）第 257668 号

Effective 系列丛书经典著作，亚马逊五星级畅销书，Ecma 的 JavaScript 标准化委员会著名专家撰写，JavaScript 语言之父、Mozilla CTO Brendan Eich 作序鼎力推荐！作者凭借多年标准化委员会工作和实践经验，深刻辨析 JavaScript 的内部运作机制、特性、陷阱和编程最佳实践，将它们高度浓缩为极具实践指导意义的 68 条精华建议。

本书共分为 7 章，分别涵盖 JavaScript 的不同主题。第 1 章主要讲述最基本的主题，如版本、类型转换要点、运算符注意事项和分号局限等。第 2 章主要讲解变量作用域，介绍此方面的一些基本概念、以及一些最佳实践经验。第 3 章主要讲解函数的使用，深刻解析函数、方法和类，并教会读者在不同的环境下高效使用函数。第 4 章主要讲解原型和对象、分析 JavaScript 的继承机制以及原型和对象使用的最佳实践和原则。第 5 章主要介绍数组和字典，阐述将对象作为集合的用法以及使用数组和字典的一些陷阱。第 6 章介绍库和 API，讲解如何设计良好的 API 的技巧，以清楚、简洁和明确地表达程序，并提高可重用率。第 7 章讲解并发，在技术上讨论一些“约定成俗”的 JavaScript 用法。

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）
责任编辑：陈佳媛
三河市杨庄长鸣印刷装订厂印刷
2014 年 1 月第 1 版第 1 次印刷
186mm × 240mm • 11.25 印张
标准书号：ISBN 978-7-111-44623-1
定 价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

本书赞誉

“这是一本绝不辜负 Effective 软件开发系列期望的编程书籍。对于任何一个想要做到严谨编程的 JavaScript 开发者来说，这本书绝对不容错过。这本书阐述了 JavaScript 内部工作的一些细节，以期帮助读者更好地利用 JavaScript 语言优势。”

——Erik Arvidsson，高级软件工程师

“很少有像 David 这样的编程语言极客能如此舒适、友好地讲解编程语言。他带领我们领会 JavaScript 语法和语义，这个过程既令人陶醉又极其深刻。本书以舒适的节奏额外提供了一些“有问题”的现实案例。当你读完本书后，你会感觉自己获得了一种强大而全面的掌控能力。”

——Paul Irish，Google Chrome 开发主管

“在阅读本书之前，我以为它只是另一本关于如何更好地使用 JavaScript 编程的书籍。然而本书远不止如此，它还会使你更深入地理解 JavaScript 这门语言，这是至关重要的。如果没有这层对 JavaScript 的深入理解，那么你绝不会懂得语言本身的任何东西，只知道其他的程序员是如何编写代码的。”

“如果你想成为一名真正优秀的 JavaScript 开发者，那么请阅读本书。就我来说，我多么希望在第一次开始 JavaScript 编程时就已经阅读了它。”

——Anton Kovalyov，JSHint 开发者

“如果你正在寻找一本正式且极具可读性的并极具洞察力的 JavaScript 语言的书籍，那不用舍近求远了。JavaScript 开发者能够从其中找到珍贵的知识宝藏，甚至技术精湛的 JavaScript 程序员也一定能从中获益。对于有其他语言经验而想一头扎进 JavaScript 世界的从业人员来说，本书是迅速学习 JavaScript 的必读之物。然而，不管你的背景如何，但都不得不承认作者 Dave Herman 在探索 JavaScript 方面做得非常棒——JavaScript 的优美部分、不足部分或介于两者之间的所有内容都囊括于本书之中。”

——Rebecca Murphey，Bocoup 高级 JavaScript 开发者

“对于任何一位理解 JavaScript 并且想要完全掌握它的人员来说，本书是必不可少的读物。Dave Herman 带给了读者深刻的、具有研究和实践意义的 JavaScript 语言理解，通过一个接一个的例子指导并帮助读者达到与他同样的理解高度。这不是一本寻求捷径的书籍，恰恰相反，是一本难得的将经验提炼为指南的书籍。它是一本为数不多让我毫不犹豫推荐的关于 JavaScript 的书籍。”

——Alex Russell，TC39 成员，Google 软件工程师

“很少有人有机会同大师一起学习他们的手艺。这本书弥补了这种遗憾，其对 JavaScript 的研究就像随一位时间旅行哲学家回到公元前 5 世纪与柏拉图一同学习。”

——Rick Waldron，JavaScript 传教士，Bocoup

译 者 序

虽然 JavaScript 在诞生之初由于商业原因及缺乏规范，一直饱受诟病。但是随着时间的推移，人们已经逐渐走出了对这门语言的偏见和误解，开始领略它那强大的语言特性威力。当下 JavaScript 语言大红大紫，研究讨论 JavaScript 的相关书籍早已汗牛充栋，但是这本书作为 Effective 软件开发系列中的一员，却是不可或缺的。

学会编写 JavaScript 程序容易，但要成为专家却实属不易。一方面是由于 JavaScript 语言的设计思想与 Java、C# 等大众语言区别很大，另一方面是由于其设计时的仓促性导致 JavaScript 语言本身精华与糟粕并存。本书的作者 David Herman 作为 JavaScript 标准化的参与者，在书中自然对 JavaScript 的精华和糟粕都进行了深入阐述，并且给出了很多实用的建议。这些建议都来自于第一线的实践经验，无论是初学者还是高级程序员，都可以从中吸收养分，进而快速成长。

本书深入阐述了 JavaScript 语言，通过它可以了解到如何有效地编写出高移植性、健壮的程序和库。本书传承了 Effective 软件开发系列的简明场景驱动风格，通过提示、技术及实用的示例代码解释 JavaScript 中的重要概念。

全书共涉及 68 条关于 JavaScript 程序设计的建议。第 1 章可以让初学者快速熟悉 JavaScript，了解 JavaScript 中的原始类型、隐式强制转换、编码类型等基本概念；第 2 章着重讲解了有关 JavaScript 的变量作用域的建议，不仅介绍了怎么做，还介绍了操作背后的原因，帮助读者加深理解；第 3 章和第 4 章的主题涵盖函数、对象及原型三大方面，这可是 JavaScript 区别于其他语言的核心，读者也不必紧张，在 David Herman 大叔的指引下，你可以轻松掌握这些核心内容，了解到业界最佳实践；第 5 章则阐述了数组和字典这两种容易混淆的常用类型及具体使用时的建议，避免陷入一些陷阱；第 6 章讲述了库和 API 设计；第 7 章讲述了并行编程，这是晋升为 JavaScript 专家的必经之路。

想要深入了解 JavaScript 并获取一线专家的宝贵经验吗？那么，这本书正好适合你。

我和同事喻杨在翻译这本书的过程中投入了不少精力，生怕给这本经典之作留下一些遗憾。感谢华章公司的编辑们对我们的支持。

最后，希望本书能给大家带来一次超凡的阅读体验。

黄博文

序

众所周知，我在 1995 年 5 月用了 10 天时间创建了 JavaScript 语言。迫于现实的压力和冲突管理的势在必行，我将 JavaScript 设计为“看起来像 Java”、“方便初学者”、“在网景浏览器中几乎能控制它的一切”的编程语言。

鉴于极具挑战性的要求和非常短的时间表，我的解决方案除了正确获得了两大特性之外（第一类函数、对象原型），还将 JavaScript 设计为一开始就极具延展性。我知道一些开发者不得不对最开始的几个版本“打补丁”来修正错误，这些先驱的方法比我使用内置库胡乱拼凑的方法更好。举例来说，虽然许多编程语言都限制了可变性，在运行时不能修改或扩展内置对象，或者标准库的名称绑定不能通过赋值被覆盖，但是 JavaScript 几乎允许完全改变每个对象。

总的来说，我相信这是一个很好的设计决定。它明确了某些领域的挑战（如在浏览器的安全边界内，安全地混用可信和不可信的代码）。对于 JavaScript 来说，支持所谓的猴子补丁是很重要的。凭借它，开发者可以编辑标准对象来解决 Bug 和仿效“未来”的功能改造一些旧的浏览器（称为 polyfill 库的 shim，在美式英语中称为 spackle）。

除了这些平凡的应用之外，JavaScript 的可塑性鼓励用户沿着几个更富有创造性的方向形成和发展创新网络。这使得用户仿效其他编程语言，创建了许多工具包和框架库：基于 Ruby 的 Prototype、基于 Python 的 MochiKit、基于 Java 的 Dojo 以及基于 Smalltalk 的 TIBET。随后是 jQuery 库（“JavaScript 的新浪潮”），2007 年我第一次看到它，对我来说，我似乎是一个后来者。它暴风雨般地引领着 JavaScript 的世界，异于其他编程语言，而从旧的 JavaScript 库中学习，开辟了浏览器的“查询和做”（query and do）模型，从根本上简化了浏览器。

这引领着 JavaScript 用户群及其创新网络，促成了一种 JavaScript“自家风格”。这种风格仍在仿效和简化其他的程序库，也促成了现代 Web 标准化的工作。

在这一演变的过程中，JavaScript 保持了向后兼容，当然默认情况下，它是可变的。另外，在 ECMAScript 标准最新的版本中新增了一些方法来冻结对象和封闭对象的属性，以防止对象扩展和属性被覆盖。JavaScript 的演进之旅远未结束。这就像生活的语言和生物系统一样，变化始终存在。在此之前，我无法预见一个单一的横扫其他程序库的“标准库”或编码风格。

前　言

学习一门编程语言，需要熟悉它的语法、形式和结构，这样我们才能编写合法的、符合语义的、具有意义和行为正确的程序。但除此之外，掌握一门语言需要理解其语用，即使用语言特性构建高效程序的方法。后一种范畴是特别微妙的，尤其是对 JavaScript 这样一种灵活而富有表现力的编程语言来说。

这是一本关于 JavaScript 语用学的书。这不是一本入门书籍，我假设你在一定程度上熟悉了 JavaScript 和通常的编程。很多优秀的 JavaScript 入门书籍可供参考，例如，Douglas Crockford 的《JavaScript: The Good Parts》和 Marijn Haverbeke 的《Eloquent JavaScript》。本书的目的是帮助你加深理解如何有效地使用 JavaScript 构建更可预测、可靠和可维护的 JavaScript 应用程序和库。

JavaScript 与 ECMAScript

在深入本书之前澄清一些术语是有必要的。这是一本关于举世皆知的 JavaScript 编程语言的书籍。然而，官方标准定义的规范的描述是一门称该语言为 ECMAScript。历史很令人费解，但这可以归结为版权问题：出于法律原因，Ecma 国际标准化组织不能使用“JavaScript”作为其标准名称。（更糟的是，标准化组织将其原来的名称 ECMA（欧洲计算机制造商协会的英文首字缩写）改为不是全大写的 Ecma 国际标准化组织。彼时，ECMAScript 这个名字大约也是早已注定。）

正式来说，当人们提到 ECMAScript 时，通常是指由 Ecma 国际标准化组织制定的“理想”语言。与此同时，JavaScript 这个名字意味着来自语言本身的所有事物，例如某个供应商特定的 JavaScript 引擎。通常情况下，人们经常交替使用这两个术语。为了保持清晰度和一致性，在本书中，我将只使用 ECMAScript 来谈论官方标准，其他情况，我会使用 JavaScript 指代语言。我还会使用常见的缩写 ES5 来指代第 5 版的 ECMAScript 标准。

关于 Web

避开 Web 来谈 JavaScript 是很难的。到目前为止，JavaScript 是唯一为用于客户端应用程

没有语言不是怪癖的，或者所有语言几乎都是受限地执行常见的最佳实践。JavaScript 是一门怪癖的或者充满限制主义色彩的编程语言。因此，与大多数其他的编程语言相比，想要高效地使用 JavaScript 编程，开发人员必须学习和追求优秀的风格、正确的用法和最佳的实践。当考虑如何做到最高效，我相信避免过度编程和构建刚性或教条式的风格指南是至关重要的。

本书以一种平衡的方式讲解 JavaScript 编程。它基于一些具体的实证和经验，而不迂回于刚性或过度的方法。我认为对于许多寻找以不牺牲表现力，并希望自由采用新思想和编程范式来编写高效 JavaScript 的人来说，它是一位重要的助手和可信赖的向导。它也是一本备受关注、充满乐趣和拥有许多令人叹为观止的示例的读物。

最后，自 2006 年，我有幸结识 David Herman。那是我第一次以 Mozilla 代表的身份邀请他作为特邀专家在 Ecma 标准化机构工作。Dave 深刻、谦逊的专家意见以及他的热情让 JavaScript 在书中的每一页大放异彩。本书绝无仅有！

Brendan Eich

序脚本的所有主流浏览器提供内置支持的编程语言。此外，近年来，随着 Node.js 平台的问世，JavaScript 已经成为一个实现服务器端应用程序的流行编程语言。

不过，本书是关于 JavaScript 而非 Web 的编程。有时，谈论一些 Web 相关的例子和应用程序的概念是帮助读者理解。但是，这本书的重点是 JavaScript 语言的语法、语义和语用，而不是 Web 平台的 API 和技术。

关于并发

JavaScript 一个新奇的方面是在并发环境中其行为是完全不明朗的。ECMAScript 标准（包括第 5 版）关于 JavaScript 程序在交互式或并发环境中的行为只字未提。第 7 章涉及并发，因此，我只是从技术角度介绍一些非官方的 JavaScript 特性。但实际上，所有主流的 JavaScript 引擎都有一个共同的并发模型。尽管在标准中未提及并发，但是致力于并发和交互式的程序是 JavaScript 编程的一个核心概念。事实上，未来版本的 ECMAScript 标准可能会正式地标准化这些 JavaScript 并发模型的共享方面。

致谢

这本书在很大程度上要归功于 JavaScript 的发明者——Brendan Eich。我深深感谢 Brendan 邀请我参与 JavaScript 标准化工作，以及他对我在 Mozilla 的职业生涯中给予的指导和支持。

本书中的大部分材料是受优秀的博客文章和在线论文的启发。我从 Ben“cowboy”Alman、Erik Arvidsson、Mathias Bynens、Tim“creationix”Caswell、Michaeljohn“inimino”Clement、Angus Croll、Andrew Dupont、Ariya Hidayat、Steven Levithan、Pan Thomakos、Jeff Walden，以及 Juriy“kangax”Zaytsev 的博客中学到很多东西。当然，本书的最终资源来自 ECMAScript 规范。ECMAScript 规范自第 5 版以来由 Allen Wirfs-Brock 不知疲倦地编辑和更新。Mozilla 开发者网络仍然是 JavaScript API 和特性最令人印象深刻的、高品质在线资源之一。

在策划和写作这本书的过程中，我有许多顾问。在我开始写作之前，John Resig 就以作者的角度给了我很多有用的建议。Blake Kaplan 和 Patrick Walton 帮我在早期阶段整理我的想法和规划出这本书的组织结构。在写作的过程中，我从 Brian Anderson、Norbert Lindenberg、Sam Tobin-Hochstadt、Rick Waldron 和 Patrick Walton 那里得到了很好的建议。

很高兴能够和 Pearson 的工作人员共事。Olivia Basegio、Audrey Doyle、Trina MacDonald、Scott Meyers 和 Chris Zahn 一直关注我提出的问题，对我的拖延报以耐心，并通融我的请求。我无法想象还能有一个更愉快的写作经历。我对能为 Effective 系列写一本书感到非常荣幸。因为很久

以前我就是《Effective C++》的粉丝，我曾经怀疑我是否有亲自书写一本 Effective 系列书籍的荣幸。

我也简直不敢相信自己有这样的好运气，能够找到梦之队一样的技术编辑。我很荣幸 Erik Arvidsson、Rebecca Murphey、Rick Waldron 和 Richard Worth 同意编辑这本书，他们为我提供了许多宝贵的批评和建议。他们多次纠正了书中一些真正令人尴尬的错误。

写一本书比我预想的要难得多。如果不是朋友和同事的支持，我可能已经失去了勇气。在我怀疑自己的时候，Andy Denmark、Rick Waldron 和 Travis Winfrey 总是给予我鼓励。

我绝大部分时候是在旧金山柏丽附近的神话般的 Java Beach 咖啡厅里写作这本书的。那里的工作人员都知道我的名字，并且我还没点餐之前，他们就知道我想要点什么。我很感谢他们提供了一个舒适的工作场所，并给我提供食物和咖啡。

我的毛茸茸的猫科小友 Schmoopy 为本书做出了它的最大贡献。至少，它不停地跳上我的膝盖，坐在屏幕前（有可能是笔记本电脑比较温暖）。Schmoopy 自 2006 年以来一直是我的忠实伙伴，我不能想象我的生活能离得开这个小毛球。

我的整个家庭对这个项目从开始到结束一直都很支持和激动。遗憾的是，我无法在我的爷爷和奶奶（Frank 和 Miriam Slamar）去世之前和他们分享这本书的成品。但他们会为我感到激动和自豪，而且本书中有一小段我儿时与爷爷 Frank 编写 BASIC 程序的经历。

最后，我要感谢我一生的挚爱 Lisa Silveria，我对她的付出无以为报。

目　录

第 1 章
让自己习惯 JavaScript

JavaScript 最初设计令人感觉亲切。由于其语法让人联想到 Java，并且具有许多脚本语言的共同特性（如函数、数组、字典和正则表达式），因此，具有少量编程经验的人也能够快速学习 JavaScript。新手程序员几乎不需要培训就可以开始编写程序，这要归功于 JavaScript 语言提供的为数不多的核心概念。

虽然 JavaScript 是如此的平易近人，但是精通这门语言需要更多的时间，需要更深入地理解它的语义、特性以及最有效的习惯用法。本书每个章节都涵盖了高效 JavaScript 编程的不同主题。第 1 章主要讲述一些最基本的主题。

第 1 条：了解你使用的 JavaScript 版本

像大多数成功的技术一样，JavaScript 已经发展了一段时间。最初 JavaScript 作为 Java 在交互式网页编程方面的补充而推向市场，但它最终完全取代了 Java 成为主流的 Web 编程语言。JavaScript 的普及使得其于 1997 年正式成为国际标准，其官方名称为 ECMAScript[⊖]。目前许多 JavaScript 的竞争实现都提供了 ECMAScript 标准的各种版本的一致性。

1999 年定稿的第 3 版 ECMAScript 标准（通常简称为 ES3），目前仍是最广泛采用的 JavaScript 版本。下一个有重大改进的标准是 2009 年发布的第 5 版，即 ES5。ES5 引入了一些新的特性，并且标准化了一些受到广泛支持但之前未规范的特性。由于 ES5 目前还未得到广泛支持，所以我会适时指出本书中的条款或建议是否特定于 ES5。

除了 ECMAScript 标准存在多个版本之外，还存在一些 JavaScript 实现支持非标准特性，而其他的 JavaScript 实现却并不支持这些特性的情况。例如，许多 JavaScript 引擎支持 const

⊖ ECMAScript 是一种由欧洲计算机制造商协会（ECMA）通过 ECMA-262 标准化的脚本程序设计语言。——译者注

关键字定义变量，但 ECMAScript 标准并没有定义任何关于 const 关键字的语义和行为。此外，在不同的实现之间，const 关键字的行为也不一样。在某些情况下，const 关键字修饰的变量不能被更新。

```
const PI = 3.141592653589793;
PI = "modified!";
PI; // 3.141592653589793
```

而其他的实现只是简单地将 const 视为 var 的代名词。

```
const PI = 3.141592653589793;
PI = "modified!";
PI; // "modified!"
```

由于 JavaScript 历史悠久且实现多样化，因此我们很难确定哪些特性在哪些平台上是可用的。而令事态更加严峻的事实是 JavaScript 的主要生态系统——Web 浏览器，它并不支持让程序员指定某个 JavaScript 的版本来执行代码。由于最终用户可能使用不同 Web 浏览器的不同版本，因此，我们必须精心地编写 Web 程序，使得其在所有的浏览器上始终工作如一。

另外，JavaScript 并不只是针对客户端 Web 编程。JavaScript 的其他应用包括服务器端程序、浏览器扩展以及针对移动和桌面应用程序的脚本。某些情况下你可能需要一个特定的 JavaScript 版本。对于这些情况，利用特定平台支持的 JavaScript 特定实现的额外特性是有意义的。

本书主要关注的是 JavaScript 的标准特性，但是也会讨论一些广泛支持的非标准特性。当涉及新标准特性或非标准特性时，了解你的应用程序运行环境是否支持这些特性是至关重要的。否则，你可能会面临这样的困境——应用程序在你自己的计算机或者测试环境上运行良好，但是将它部署在不同的产品环境中时却无法运行。例如，const 关键字在支持非标准特性的 JavaScript 引擎上测试时运行良好，但是，当将它部署在不识别 const 关键字的 Web 浏览器上时就会出现语法错误。

ES5 引入了另一种版本控制的考量——严格模式（strict mode）。此特性允许你选择在受限制的 JavaScript 版本中禁止使用一些 JavaScript 语言中问题较多或易于出错的特性。由于其语法设计向后兼容，因此即使在那些没有实现严格模式检查的环境中仍然可以执行严格代码[⊖]（strict code）。在程序中启用严格模式的方式是在程序的最开始增加一个特定的字符串字面量（literal）。

```
"use strict";
```

同样，你也可以在函数体的开始处加入这句指令以启用该函数的严格模式。

⊖ 严格代码是指期望运行于严格模式下的代码。——译者注

```
function f(x) {
    "use strict";
    // ...
}
```

使用字符串字面量作为指令语法看起来有点怪异，但它的好处是向后兼容。由于解释执行字符串字面量并没有任何副作用，所以ES3引擎执行这条指令是无伤大雅的。ES3引擎解释执行该字符串，然后立即丢弃其值。这使得编写的严格模式的代码可以运行在旧的JavaScript引擎上，但有一个重要的限制：旧的引擎不会进行任何的严格模式检查。如果你没有在ES5环境中做过测试，那么，编写的代码运行于ES5环境中就很容易出错。

```
function f(x) {
    "use strict";
    var arguments = []; // error: redefinition of arguments
    // ...
}
```

在严格模式下，不允许重定义arguments变量，但没有实现严格模式检查的环境会接受这段代码。然而，这段代码部署在实现ES5的产品环境中将导致程序出错。所以，你应该总是在完全兼容ES5的环境中测试严格代码。

“use strict”指令只有在脚本或函数的顶部才能生效，这也是使用严格模式的一个陷阱。这样，脚本连接变得颇为敏感。对于一些大型的应用软件，在开发中使用多个独立的文件，然而部署到产品环境时却需要连接成一个单一的文件。例如，想将一个文件运行于严格模式下：

```
// file1.js
"use strict";
function f() {
    // ...
}
// ...
```

而另一个文件不是运行于严格模式下：

```
// file2.js
// no strict-mode directive
function g() {
    var arguments = [];
    // ...
}
// ...
```

我们怎样才能正确地连接这两个文件呢？如果我们以file1.js文件开始，那么连接后的代码运行于严格模式下：