

YAHOO! PRESS

*JavaScript: The Good Parts*

深入挖掘JavaScript精华



# JavaScript

## 语言精粹

(修订版)

*Douglas Crockford* 著

赵泽欣 鄢学鹏 译

O'REILLY®



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
<http://www.phei.com.cn>

## 内 容 简 介

JavaScript 曾是“世界上最被误解的语言”，因为它担负太多的特性，包括糟糕的交互和失败的设计，但随着 Ajax 的到来，JavaScript “从最受误解的编程语言演变为最流行的语言”，这除了幸运之外，也证明了它其实是一门优秀的语言。Douglas Crockford 在本书中剥开了 JavaScript 沾污的外衣，抽离出一个具有更好可靠性、可读性和可维护性的 JavaScript 子集，让你看到一门优雅的、轻量级的和非常富有表现力的语言。作者从语法、对象、函数、继承、数组、正则表达式、方法、样式和优美的特性这 9 个方面来呈现这门语言真正的精华部分，通过它们完全可以构建出优雅高效的代码。作者还通过附录列出了这门语言的毒瘤和糟粕部分，且告诉你如何避免它们。最后还介绍了 JSLint，通过它的检验，能有效地保障我们的代码品质。

这是一本介绍 JavaScript 语言本质的权威书籍，值得任何正在或准备从事 JavaScript 开发的人阅读，并且需要反复阅读。学习、理解、实践大师的思想，我们才可能站在巨人的肩上，才有机会超越大师，这本书就是开始。

978-0-596-51774-8 JavaScript: The Good Part © 2008 by O'Reilly Media, Inc.

Simplified Chinese edition, jointly published by O'Reilly Media, Inc. and Publishing House of Electronics Industry, 2012. Authorized translation of the English edition, 2012 O'Reilly Media Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

本书中文简体版专有出版权由 O'Reilly Media, Inc. 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。

版权贸易合同登记号 图字：01-2009-0879

## 图书在版编目 (CIP) 数据

JavaScript 语言精粹 / (美) 克罗克福德 (Crockford, D.) 著; 赵泽欣, 鄢学鹄译. —修订本. —北京: 电子工业出版社, 2012.8

书名原文: JavaScript: The Good Parts

ISBN 978-7-121-17740-8

I. ①J… II. ①克… ②赵… ③鄢… III. ①JAVA 语言—程序设计 IV. ①TP312

中国版本图书馆 CIP 数据核字(2012)第 171680 号

策划编辑: 张春雨

责任编辑: 付 睿

封面设计: Karen Montgomery

印 刷:

装 订:

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: 11 字数: 258 千字

印 次: 2012 年 8 月第 1 次印刷

定 价: 49.00 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。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

---

# 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

## 致敬

给伙计们：Clement、Philbert、Seymore、Stern 和不可遗忘的 C. Twildo。

---

# 再版译者序

直到重看第 1 版的译者序，我才意识到，不知不觉时间竟然已经过去快 4 年了。

这段不长也不短的时间里，发生了很多事，请容我语无伦次地列举出来。

在这 4 年里，JavaScript 的发展丝毫没有减速的迹象，随着新一轮的浏览器竞赛，HTML5 逐渐得到普及，JavaScript 在 Web 开发领域中的地位日益提高。NodeJS 的出现，更是让 JavaScript 蔓延到服务器端编程领域。还值得一提的是 2 年多以前 CoffeeScript 的诞生，它吸收了 JavaScript 语言的精华，去除了很多本书中提到的毒瘤和糟粕，还添加了很多现代脚本语言的特性，仿佛就是老道（我们私下里这样尊称本书的作者 Douglas Crockford）所想要的 JavaScript 精华子集。老道本人也确实对它推崇有加。

已经 10 年未有重大版本发布的 ECMA，终于在 2009 年年底发布 ECMA-262 的第 5 个版本 (ES5)，这个版本在最后时刻取代了过于激进的 ED4 (JavaScript 2.0)，据说老道的倡议起了重大的作用。谢天谢地，我真不愿意像写 Java 一样写 JavaScript。本书中的一些精华也被 ES5 采纳，比如 JSON 成为 ES5 的标准组件；再比如 ES5 支持“严格”模式，在此模式下，使用未声明的全局变量或者 with 语句，都会抛出错误。下一代的 ES 版本 (ES6) 正在制定中，它的名字是我们无比熟悉的“Harmony (和谐)”，期待更多本书中的优秀思想会出现在 ES6 中。

2009 年，在北京举行的 Qcon 大会上，我和学鵬有幸遇到了老道，并和他合影留念。他比我想象中高大，留着拉风的络腮胡子（后来我看到了他未留胡子的照片，觉得老道还是留胡子的好）。人很安静，看上去有那么一点技术人员的木讷。但站在演讲台上却是侃侃而谈，掷地有声。

于我而言，这 4 年我的人生同样发生了重大的变化。我结了婚，装修了房子，在去年年底，我的孩子也降生了。

.....

就此打住吧，我快要走题了。

这次修订的版本，对照中英文的勘误，修正了 80 余个错误。一些已经过时的经验和数据，我们也尽所能通过译者注进行了标注。作为一名译者，忠实地翻译本书的内容本是职责。

---

在翻译的过程中，我们曾遇到一些与自己的开发经验有冲突的部分。我也在不同场合多次听闻国内外的业界同行，对本书的部分内容提出了不同的看法，有人认为本书一些观点不是 JavaScript 的最佳实践，有人甚至言辞激烈地说有些观点根本就是老道的个人主观看法，而非科学的求证。

在我眼里，我觉得我的孩子是最漂亮的宝宝，天下的父母怕是都无法完全客观地看待自己的孩子。我相信老道视他提炼的 JavaScript 子集，就如同我看自己的孩子。其实，老道在本书的最开头，就已经表明：本书内容是他根据自己的经验提炼的精华子集。也许，每个开发人员对何谓精华、何谓糟粕有自己的看法。但我想，JavaScript 还会不停地发展下去，本书中的内容也许还需要多次修订，但“取其精华，弃其糟粕”的思想是不会过时的。

我的孩子，就遗传了我和太太的精华，如果一定要说有一点是糟粕的话，那就是他没有继承我喜欢安静的优点，每天不闹到精疲力尽就不肯睡去。所以，我只有在孩子晚上熟睡后，才有得闲暇写下断续的文字。

感谢本修订版的策划编辑张春雨老师，他容忍了我的多次跳票，请原谅一个迷恋孩子的父亲。

2012 年 8 月 5 日 码字于杭州

Douglas Crockford 是一位大师。

翻译大师的作品，一边是感到万分的荣幸，一边也是兢兢小心。因为吉尔伯特·海特（美国教育家）曾经说过：写了一本很糟糕的书只是犯错而已，而把一本好书翻译得很糟糕则是犯罪。但这样的大师经典之作，即便是冒着犯罪的风险，也值得翻译出来并推荐给大家。一直到现在，依然有很多资深的开发人员对 JavaScript 存有偏见。秦歌和我，分别负责雅虎口碑网和淘宝网的前端开发组，对此的感受更为深刻。但即便如此，也不得不承认，JavaScript 正日益成为互联网中最普及和最重要的开发语言。

Crockford 曾写过很著名的一篇文章——《JavaScript：世界上最被误解的语言》。建议看到这里的所有读者都找来这篇文章（<http://javascript.crockford.com/javascript.html>）并仔细阅读。早期的商业原因和规范欠缺给 JavaScript 这门语言蒙上了阴影；Copy+Paste 式滥用也让 JavaScript 显得廉价不堪；更糟糕的是，还有大量不负责任的书籍把蹩脚的用例奉为正统，印成了铅字，让新手们从一开始就走上了歧途。不可否认，JavaScript 自身确实存在着不少瑕疵，但瑕不掩瑜。Crockford 为此凭借他广博的学识和丰富的经验提炼出了 JavaScript 的精华子集。开发人员只要在这个子集的范畴中编程，就既能使用 JavaScript 强大的表现力和卓越的动态性，又能免去许多无端的调试烦恼和安全隐忧。

这本书很薄，但承载的内容却非常丰厚和深入。翻译的过程中我也常感汗颜，原来自诩对 JavaScript 颇为了解的我深刻感受到自己知识面的浅薄和不完整，于是翻译的过程也成为自己检讨和学习的过程，收获颇丰。Crockford 在前言中告诫大家，这本书是需要反复阅读的。我们同样推荐所有的读者这样做。

我想每一个热爱技术的开发人员都希望自己有一天成为某个领域的大师。我通过翻译大师的著作也得到了一个启示，“取其精华，去其糟粕”本就是前人告诉我们的学习态度与方法，对日新月异的 IT 领域来说更该如此。当我们面对这些层出不穷的新技术、新理念时，不要匆忙地照单全收或全盘否定。找到最适合工作或自己最感兴趣的技术，并用科学的方法潜下心来坚持学习和研究，我们同样也可以成为大师！

“大师牛人，宁有种乎？”

最后，我要感谢博文视点的编辑赵士威在本书翻译过程中给予我们的莫大帮助。还有周筠



---

老师，她爽朗的笑声让人备感亲切。我还要感谢我的同事，来自美国 NCSU 的晓荷，是博采中外的她给我建议，把 JavaScript 的“好、中、坏”特性翻译为更贴切的“精华、鸡肋、糟粕”。当然，家中的领导（负责接管稿费）是一定要特别感谢的。相信我，如果你身后没有一位善解人意的女人，还是不要去做翻译的好。

译者：赵泽欣（小马），鄢学鹏（秦歌）

2008 年 11 月于杭州城西



---

# 目录

## Table of Contents

前言 .....	xv
第 1 章 精华 .....	1
为什么要使用 JavaScript .....	2
分析 JavaScript .....	2
一个简单的试验场 .....	4
第 2 章 语法 .....	5
空白 .....	5
标识符 .....	6
数字 .....	7
字符串 .....	8
语句 .....	10
表达式 .....	15
字面量 .....	18
函数 .....	19
第 3 章 对象 .....	20
对象字面量 .....	20
检索 .....	21
更新 .....	22
引用 .....	22
原型 .....	22
反射 .....	23
枚举 .....	24
删除 .....	24
减少全局变量污染 .....	25
第 4 章 函数 .....	26
函数对象 .....	26

函数字面量 .....	27
调用 .....	27
参数 .....	30
返回 .....	31
异常 .....	31
扩充类型的功能 .....	32
递归 .....	33
作用域 .....	36
闭包 .....	36
回调 .....	39
模块 .....	40
级联 .....	42
柯里化 .....	43
记忆 .....	43
第 5 章 继承 .....	46
伪类 .....	46
对象说明符 .....	49
原型 .....	50
函数化 .....	51
部件 .....	55
第 6 章 数组 .....	57
数组字面量 .....	57
长度 .....	58
删除 .....	59
枚举 .....	59
容易混淆的地方 .....	60
方法 .....	60
指定初始值 .....	62
第 7 章 正则表达式 .....	64
一个例子 .....	65
结构 .....	69
元素 .....	71
第 8 章 方法 .....	77
Array .....	77
Function .....	83
Number .....	84

Object.....	85
RegExp .....	86
String.....	88
第 9 章 代码风格 .....	94
第 10 章 优美的特性.....	98
附录 A 毒瘤 .....	101
附录 B 糟粕 .....	109
附录 C JSLint .....	115
附录 D 语法图 .....	127
附录 E JSON.....	138
索引 .....	149

---

# 前言

## Preface

要是有所得罪请原谅。本是出自一番好意，  
只是想显点粗浅技艺，那才是我们的初衷。  
——威廉·莎士比亚，《仲夏夜之梦》(*A Midsummer Night's Dream*)

这是一本关于 JavaScript 编程语言的书。它的读者是那些因为偶然事件或好奇心驱使而首次冒险进入 JavaScript 世界的程序员。它也是为那些有着 JavaScript 入门经验但准备更深入地了解这门语言的程序员准备的。JavaScript 是一门强大得令人惊讶的语言。它有时会不按常理出牌，但是作为一门轻量级的语言，它是易于掌握的。

本书的目标是帮助你学习 JavaScript 的编程思想。我将展示这门语言的组成部分，并且让你逐步上手，学会如何组合各个部分。这不是一本参考书，它不会对这门语言和它的怪癖进行全面而详尽的介绍。它不包含你希望知道的一切，那些东西你很容易在网上找到。反之，这本书仅包含那些真正重要的东西。

这本书不是写给初学者的。我希望某天写一本叫《JavaScript: 第一阶段》(*JavaScript: The First Parts*) 的书，但是此书非彼书。这也不是一本关于 Ajax 或 Web 编程的书。本书关注的就是 JavaScript，它只是 Web 开发者必须掌握的语言之一。

这不是一本傻瓜书。这本书虽然薄，但知识点密集，它包括了大量的内容。如果为了理解它而不得不反复阅读，请别沮丧，你的付出将会有所回报。

## 本书的约定

### Conventions Used in This Book

本书使用下列排版约定。

*斜体 (Italic)*

表示专业词汇、链接 (URL)、文件名和文件扩展名。


`等宽字体 (Constant width)`

表示广义上的计算机编码。它们包括命令、配置、变量、属性、键、请求、函数、方

法、类型、类、模块、属性、参数、值、对象、事件、事件处理程序、XML 与 XHTML 标签、宏和关键字。

### 等宽粗体 (Constant width bold)

表示应该由用户按照字面输入的命令或其他文本。

中文版书中切口处的“”表示原书页码，便于读者与原英文版图书对照阅读，本书的索引所列页码为原英文版页码。

## 代码用例

### Using Code Examples

这本书是为了帮助你做好工作。一般来说，你可以在程序和文档中使用本书中的代码。你无须联系我们获取许可。例如，使用来自本书的几段代码写一个程序是不需要许可的。出售和散布 O'Reilly 书中用例的光盘 (CD-ROM) 是需要许可的。通过引用本书和用例代码来回答问题是不需要许可的。把本书中大量的用例代码并入到你的产品文档中是需要许可的。

我们赞赏但不强求注明信息来源。一条信息来源通常包括标题、作者、出版者和国际标准书号 (ISBN)。例如：“*JavaScript: The Good Parts* by Douglas Crockford. Copyright 2008 Yahoo! Inc., 978-0-596-51774-8。”。

如果你感到对示例代码的使用超出了正当引用或这里给出的许可范围，请随时通过 [permissions@oreilly.com](mailto:permissions@oreilly.com) 联系我们。

## Safari® 在线图书



如果你在你最喜爱的技术图书的封面上看到 Safari® 联机丛书图标，那意味着此书也可以通过 O'Reilly Network Safari Bookshelf 在线获取。

Safari 提供了比电子书更好的解决方案。它是一个虚拟图书馆，让您可以轻松搜寻成千上万的顶尖技术书籍、剪切和粘贴代码样本、下载某些章节、在你需要最准确和即时的信息时快速找到答案。免费试用请访问 <http://safari.oreilly.com>。

## 如何联系我们

### How to Contact Us

如果你想就本书发表评论或有任何疑问，敬请联系出版社。

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

我们将为本书提供主页，在其中提供勘误表、示例及其他附加信息。读者可从如下网址访问：

<http://www.oreilly.com/catalog/9780596517748>

如果你想就本书发表评论或提问技术问题，请发送 E-mail 至：

[bookquestions@oreilly.com](mailto:bookquestions@oreilly.com)

关于我们的书籍、会议、资源中心和 O'Reilly 网络的更多信息请登录我们的网址：

<http://www.oreilly.com/>

## 致谢

## Acknowledgments

感谢那些指出我的很多严重错误的审稿者。在生活中，还有什么比有真正聪明的人指出你的过失更好的事情呢？更好的是他们赶在书出版之前做了这个事情。谢谢你们，Steve Souders、Bill Scott、Julien Lecomte、Stoyan Stefanov、Eric Miraglia 和 Elliotte Rusty Harold。

谢谢那些同我一起在 Electric Communities 和 State Software 工作并帮助我发现这门语言实际上有许多精华的人们，特别是 Chip Morningstar、Randy Farmer、John La、Mark Miller、Scott Shattuck 和 Bill Edney。

谢谢雅虎公司（Yahoo! Inc.），因为它给我时间去从事这个项目，并让我在一个如此之棒的地方工作，谢谢过去和现在在 Ajax Strike Force 的所有成员。我也要谢谢 O'Reilly Media, Inc.，尤其是使事情变得如此顺利的 Mary Treseler、Simon St.Laurent 和 Sumita Mukherji。

特别感谢 Lisa Drake 教授所做的所有事情。同时，我要谢谢那些一直为使 ECMAScript 成为一门更好的语言而奋斗的 ECMA TC39<sup>译注 1</sup> 中的伙计们。

最后，谢谢 Brendan Eich，这位世界上最被误解的编程语言<sup>译注 2</sup>的设计者，没有他，这本书也就没有必要了。

---

译注 1： TC39 是研究 JavaScript 语言进化的技术委员会的名字。详情见 <http://www.ecma-international.org/memento/TC39.htm>。

译注 2： 本书作者曾写过一篇文章 *JavaScript: The World's Most Misunderstood Programming Language* (<http://javascript.crockford.com/javascript.html>)。在 2008 年的 3 月 3 日他又写了一篇 *The World's Most Misunderstood Programming Language Has Become the World's Most Popular Programming Language* (<http://javascript.crockford.com/popular.html>)。



……我不过略有一些讨人喜欢的地方而已，怎么会有什么迷人的魔力？  
——威廉·莎士比亚，《温莎的风流娘儿们》(*The Merry Wives of Windsor*)

当我还是一个初出茅庐的程序员时，我想掌握自己所使用语言的每个特性。我写程序时会尝试使用所有的特性。我认为这是炫耀的好方法，而我也的确出了不少风头，因为我对各个特性了如指掌，谁有问题我都能解答。

最终，我认定这些特性中有一部分特性带来的麻烦远远超出它们的价值。其中，一些特性因为规范很不完善而可能导致可移植性问题，一些特性会导致代码难以阅读或修改，一些特性诱使我追求奇技淫巧但却易于出错，还有一些特性就是设计错误。有时候语言的设计者也会犯错。

大多数编程语言都有精华和糟粕。我发现如果取其精华而弃其糟粕的话，我可以成为一个更好的程序员。毕竟，用坏材料怎么能做出好东西呢？

标准委员会想要移除一门语言中的缺陷部分，这几乎是不可能的，因为这样做会损害所有依赖于那些糟粕的蹩脚的程序。除了在已存在的一大堆缺陷上堆积更多的特性，他们通常无能为力。而且新旧特性并不总是能和谐共处，从而可能产生出更多的糟粕。

但是，你有权力定义你自己的子集。你完全可以基于精华的那部分去编写更好的程序。

JavaScript 中糟粕的比重超出了预料。它一诞生，就在短到令人吃惊的时间里被全世界所接受。它从来没有在实验室里被试用和打磨。当它还非常粗糙时，它就被直接集成到网景的 Navigator 2 浏览器中。随着 Java™ 的小应用程序 (Java™ applets) 的失败，JavaScript 变成了默认的“Web 语言”。作为一门编程语言，JavaScript 的流行几乎完全不受它的质量的影响。

好在 JavaScript 有一些非常精华的部分。在 JavaScript 中，美丽的、优雅的、富有表现力的语言特性就像一堆珍珠和鱼目混杂在一起。JavaScript 最本质的部分被深深地隐藏着，以至于多年来对它的主流观点是：JavaScript 就是一个丑陋的、没用的玩具。本书的目的就是要揭示 JavaScript 中的精华，让大家知道它是一门杰出的动态编程语言。JavaScript 就像一块大理石，我要剥落那些不好的特性直到这门语言的真实本质自我显露出来。我相信我精雕细琢出来的优雅子集大大地优于这门语言的整体，它更可靠、更易读、更易于维护。

这本书不打算全面描述这门语言。反之，它将专注在精华部分，同时会偶尔警告要去避免糟粕部分。这里将被描述的子集可以用来构造可靠的、易读的程序，无论规模大小。通过仅专注于精华部分，我们就可以缩短学习时间，增强健壮性，并且还能拯救一些树木<sup>译注 1</sup>。

或许，只学习精华的最大好处就是你可以不必头痛如何忘记糟粕。忘掉不好的模式是非常困难的。这是一个非常痛苦的工作，我们中的大多数人都会很不愿意面对。有时候，制定语言的子集是为了让学生更好地学习。但在这里，我制定的 JavaScript 子集是为了让专业人员更好地工作。

## 为什么要使用 JavaScript

### Why JavaScript?

JavaScript 是一门重要的语言，因为它是 Web 浏览器的语言。它与浏览器的结合使它成为世界上最流行的编程语言之一。同时，它也是世界上最被轻视的编程语言之一。浏览器的 API 和文档对象模型（DOM）相当糟糕，连累 JavaScript 遭到不公平的指责。在任何语言中处理 DOM 都是一件痛苦的事情，它的规范制定得很拙劣并且实现互不一致。本书很少涉及 DOM，我认为写一本关于 DOM 的精华的书就像执行一项不可能完成的任务。

JavaScript 是最被轻视的语言，因为它不是所谓的主流语言（SOME OTHER LANGUAGE）<sup>译注 2</sup>。如果你擅长某些主流语言，但却在一个只支持 JavaScript 的环境中编程，那么被迫使用 JavaScript 的确是相当令人厌烦的。在这种情形下，大多数人觉得没必要先去学好 JavaScript，但结果他们会惊讶地发现，JavaScript 跟他们宁愿使用的主流语言有很大不同，而且这些不同至为关键。

JavaScript 令人惊异的事情是，在对这门语言没有太多了解，甚至对编程都没有太多了解的情况下，你也能用它来完成工作。它是一门拥有极强表达能力的语言。当你知道要做什么时，它还能表现得更好。编程是很困难的事情。绝不应该在懵懵懂懂的状态下开始你的工作。

## 3 分析 JavaScript

### Analyzing JavaScript

JavaScript 建立在一些非常优秀的想法和少数非常糟糕的想法之上。

那些优秀的想法包括函数、弱类型、动态对象和富有表现力的对象字面量表示法。那些糟糕的想法包括基于全局变量的编程模型。

---

译注 1： 作者这里幽默地暗示这本书只关注精华部分，所以书变薄了，用的纸张少了，就可以少砍伐一些树木。

译注 2： 专指一些主流语言，像 C、C++、Java、Perl、Python 等。

JavaScript 的函数是（主要）基于词法作用域（lexical scoping）<sup>译注 3</sup> 的顶级对象。JavaScript 是第一个成为主流的 Lambda <sup>译注 4</sup> 语言。实际上，相对于 Java 而言，JavaScript 与 Lisp <sup>译注 5</sup> 和 Scheme <sup>译注 6</sup> 有更多的共同点。它是披着 C 外衣的 Lisp。这使得 JavaScript 成为一个非常强大的语言。

现今大部分编程语言中都流行要求强类型。其原理在于强类型允许编译器在编译时检测错误。我们能越早检测和修复错误，付出的代价就越小。JavaScript 是一门弱类型的语言，所以 JavaScript 编译器不能检测出类型错误。这可能让从强类型语言转向 JavaScript 的开发人员感到恐慌。但事实证明，强类型并不会让你的测试工作变得轻松。而且我在工作中发现，强类型检查找到的错误并不是令我头痛的错误。另一方面，我发现弱类型是自由的。我无须建立复杂的类层次，我永远不用做强制造型，也不用疲于应付类型系统以得到想要的行为。

JavaScript 有非常强大的对象字面量表示法。通过列出对象的组成部分，它们就能简单地被创建出来。这种表示法是 JSON 的灵感来源，它现在已经成为流行的数据交换格式 <sup>译注 7</sup>。（附录 E 中将会有更多关于 JSON 的内容。）

原型继承是 JavaScript 中一个有争议的特性。JavaScript 有一个无类型的（class-free）对象系统，在这个系统中，对象直接从其他对象继承属性。这真的很强大，但是对那些被训练使用类去创建对象的程序员们来说，原型继承是一个陌生的概念。如果你尝试对 JavaScript 直接应用基于类的设计模式，你将会遭受挫折。但是，如果你学会了自如地使用 JavaScript 原型，你的努力将会有所回报。

JavaScript 在关键思想的选择上饱受非议。虽然在大多数情况下，这些选择是合适的。但是有一个选择相当糟糕：JavaScript 依赖于全局变量来进行连接。所有编译单元的所有顶级变量被撮合到一个被称为全局对象（*the global object*）的公共命名空间中。这是一件糟糕的事情，因为全局变量是魔鬼，但它们在 JavaScript 中却是基础。幸好，我们接下来会看到，JavaScript 也给我们提供了缓解这个问题的处理方法。

在某些情况下，我们可能无法忽略糟粕，还有一些毒瘤难以避免，当涉及这些部分时，我会将它们指出来。它们也被总结在附录 A 中。但是我们将成功地避免本书中提到的大多数糟粕，它们中的大部分被总结在附录 B 中。如果你想学习那些糟粕，以及如何拙劣地使用

---

译注 3： JavaScript 中的函数是根据词法来划分作用域的，而不是动态划分作用域的。具体内容参见《JavaScript 权威指南》中译第 5 版相关章节——“8.8.1 词法作用域”。

译注 4： Lambda 演算是一套用于研究函数定义、函数应用和递归的形式系统。它对函数式编程有巨大的影响，比如 Lisp 语言、ML 语言和 Haskell 语言。更多详细内容请参见 [http://zh.wikipedia.org/wiki/λ\\_演算](http://zh.wikipedia.org/wiki/λ_演算)。

译注 5： Lisp（全名 LISt Processor，即链表处理语言），是由约翰·麦卡锡在 1960 年左右创造的一种基于 λ 演算的函数式编程语言。更多详细内容请参见 <http://zh.wikipedia.org/wiki/Lisp>。

译注 6： Scheme，一种多范型的编程语言，它是两种 Lisp 主要的方言之一。更多详细内容请见 <http://zh.wikipedia.org/wiki/Scheme>。

译注 7： 本书作者也是 JSON（JavaScript Object Notation）的创立者。官方网站中文版网址是 <http://json.org/json-zh.html>。

它们，请参阅任何其他 JavaScript 书籍。

4 《ECMAScript 编程语言》第 3 版定义了 JavaScript( 又称 JScript) 的标准, 它可以从 <http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf> 获得。本书所描述的是 ECMAScript 的一个特定的子集。本书并不描述整个语言, 因为它排除了糟粕的部分。这里排除得也许并不彻底, 回避了一些极端情况。你也应该这样, 走极端只会带来风险和苦恼。

附录 C 描述了一个叫 JSLint 的编程工具, 它是一个 JavaScript 解析器, 它能分析 JavaScript 问题并报告它包含的缺点。JSLint 提出了比一般的 JavaScript 开发更严格的要求。它能让你确信你的程序只包含精华部分。

JavaScript 是一门反差鲜明的语言。它包含很多错误, 而且多刺, 所以你可能会疑惑: “为什么我要使用 JavaScript?” 有两个答案。第一个是你没有选择。Web 已变成一个重要的应用开发平台, 而 JavaScript 是唯一一门所有浏览器都可以识别的语言。很不幸, Java 在浏览器环境中失败了, 否则想用强类型语言的人就有其他选择。但是 Java 确实失败了, 而 JavaScript 仍在蓬勃发展, 这恰恰说明 JavaScript 确有其过人之处。

另一个答案是, 尽管 JavaScript 有缺陷, 但是它真的很优秀。它既轻量级又富有表现力。并且一旦你熟练掌握了它, 就会发现函数式编程是一件很有趣的事。

但是为了更好地使用这门语言, 你必须知道它的局限。我将会无情地揭示它们, 不要因此而气馁。这门语言的精华足以弥补它的糟粕。

## 一个简单的试验场

### A Simple Testing Ground

如果你有一个 Web 浏览器和任意一个文本编辑器, 那么你就有了运行 JavaScript 程序所需要的一切。首先, 请创建一个 HTML 文件, 起个名字, 比如 *program.html*:

```
<html><body><pre><script src="program.js">
</script></pre></body></html>
```

接下来, 在同一个文件夹内, 创建一个脚本文件, 比如起名为 *program.js*:

```
document.writeln('Hello, world!');
```

下一步, 用你的浏览器打开你的 HTML 文件查看结果。本书贯彻始终都会用到一个 `method` 方法去定义新方法。下面是它的定义:

```
Function.prototype.method = function (name, func) {
    this.prototype[name] = func;
    return this;
};
```

我会在第 4 章解释它。

# 优美的特性

## Beautiful Features

我让你的脚玷污我的嘴唇，让你的肖像玷污我的眼睛，让你的每一部分玷污我的心，等候着你的答复。你的最忠实的……

——威廉·莎士比亚，《空爱一场》（*Love's Labor's Lost*）

去年我被邀请为 Andy Oram 和 Greg Wilson 的 *Beautiful Code*<sup>译注 1</sup>（O'Reilly 出版）一书写一篇文章，这是一本以计算机程序的表达之美为主题的选集。我负责的章节将介绍 JavaScript，通过那一部分来证明 JavaScript 不虚其名，它的确是抽象、强大且有用的。然而，我想避开不谈浏览器和其他适合使用 JavaScript 的地方。我想要强调其更有分量的内容，以显示它是值得尊敬的语言。

我立即想到 Vaughn Pratt 的自顶向下的运算符优先级解析器<sup>译注 2</sup>（Top Down Operator Precedence parser），我在 JSLint 中运用了它。在计算机的信息处理技术中，解析是一个重要的主题。一门语言是否具备为其自身编写一个编译器的能力，仍然是对这门语言完整性的一个测试。

我想把用 JavaScript 编写的 JavaScript 解析器的全部代码都包含在文章中。但是我的章节仅是 30 章或 40 章之一，我感觉被束缚在那几页短短的篇幅里。更大的困难是大部分读者没有使用 JavaScript 的经验，我也不得不介绍这门语言和它的特色。

所以，我决定提炼这门语言的子集。这样，我就不必解析整个语言，并且也就不需要描述整个语言了。我把这个子集叫做精简的 JavaScript（Simplified JavaScript）。提炼子集并不难：它包括的就是我编写解析器所需要的特性。我在 *Beautiful Code*（《代码之美》）一书中是这样描述的。

精简的 JavaScript 里都是好东西，包括以下主要内容。

函数是顶级对象

在精简 JavaScript 中，函数是有词法作用域的闭包（lambda）。

---

译注 1: *Beautiful Code* 是 O'Reilly 2007 年 6 月出版的书籍，参见 <http://oreilly.com/catalog/9780596510046/>。

译注 2: 详细内容请参阅作者的文章 *Top Down Operator Precedence parser*——<http://javascript.crockford.com/tdop/tdop.html>。

对象是无类别的。我们可以通过普通的赋值给任何对象增加一个新成员属性。一个对象可以从另一个对象继承成员属性。

#### 对象字面量和数组字面量

这对创建新的对象和数组来说是一种非常方便的代表法。JavaScript 字面量是数据交换格式 JSON 的灵感之源。

子集包含了 JavaScript 精华中最好的部分。尽管它是一门小巧的语言，但它很强大且非常富有表现力。JavaScript 有许多本不应该加入的额外特性，正如你在随后的附录中会看到的那样，它有大量的会带来负面价值的特性。在子集中没有丑陋或糟糕的内容，它们全部都被筛选了。

精简的 JavaScript 不是一个严格的子集。我添加了少许新特性。最简单的是增加了 `pi` 作为一个简单的常量。我这么做是为了证明解析器的一个特性。我也展示了一个更好的保留字策略并证明哪些保留字是多余的。在一个函数中，一个单词不能既被用做变量或参数名，又被用做一个语言特性。你可以让某个单词用在其中之一上，并允许程序员自己选择。这会是一门语言易于学习，因为你没必要知道你不会使用的特性。并且它会使这门语言易于扩展，因为它无须保留更多的保留字来增加新特性。

我也增加了块级作用域。块级作用域不是一个必需的特性，但没有它会让有经验的程序员感到困惑。包含块级作用域是因为我预期解析器可能会被用于解析非 JavaScript 语言，并且那些语言能正确地界定作用域。我编写这个解析器的代码风格不关心块作用域是否可用。我推荐你也采用这种方式来写。

当开始构思本书的时候，我想进一步地发展这个子集，我想展示除了排除低价值特性外，如何通过不做任何改变来获得一个现有的编程语言，并且使它得到有效的改进。

我们看到大量的特性驱动的产品设计，其中特性的成本没有被正确计算。对于用户来说，某些特性可能有一些负面价值，因为它们使产品更加难以理解和使用。我们发现人们想要的产品其实只要能工作即可。事实证明产生恰好可以工作的设计比集合一大串特性的设计要困难得多。

特性有规定成本、设计成本和开发成本，还有测试成本和可靠性成本。特性越多，某个特性出现问题，或者和其他特性相互干扰的可能性就越大。在软件系统中，存储成本是无足轻重的，但在移动应用中，它又变得重要了。它们抬高了电池的效能成本，因为摩尔定律并不适用于电池。

特性有文档成本。每个特性都会让产品指南变得更厚，从而增加了培训成本。只对少数用户有价值的特性增加了所有用户的成本。所以在设计产品和编程语言时，我们希望直接使用核心的精华部分，因为是这些精华创造了大部分的价值。

&lt;100



在我们使用的产品中，总能找到好的部分。我们喜欢简单，追求简洁易用，但是当产品缺乏这种特性时，就要自己去创造它。微波炉有一大堆特性，但是我只会用烹调和定时，使用定时功能就足够麻烦的了。对于特性驱动型的设计，我们唯有靠找出它的精华并坚持使用，才能更好地应对其复杂性。

如果产品和编程语言被设计得仅留下精华，那该有多好。