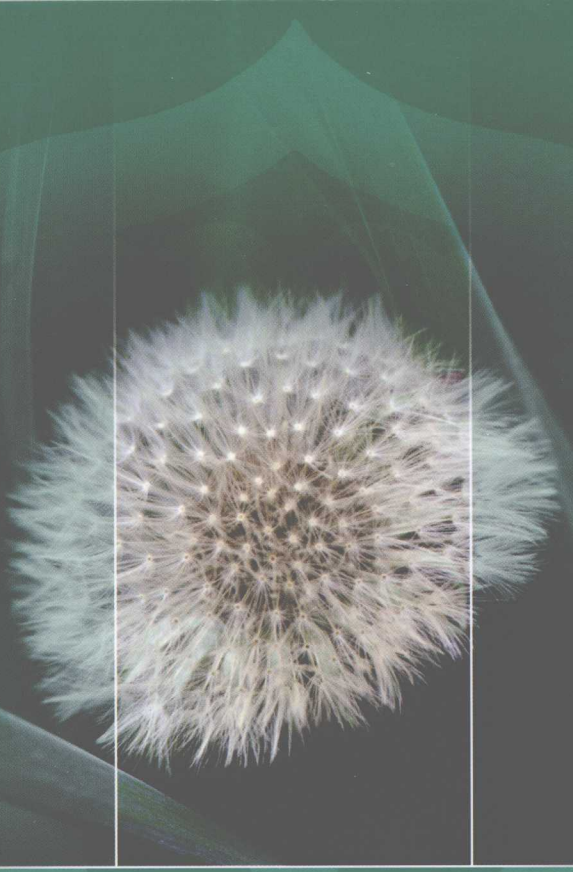
动态语言技术

精品书廊

Broadview
www.broadview.com.cn

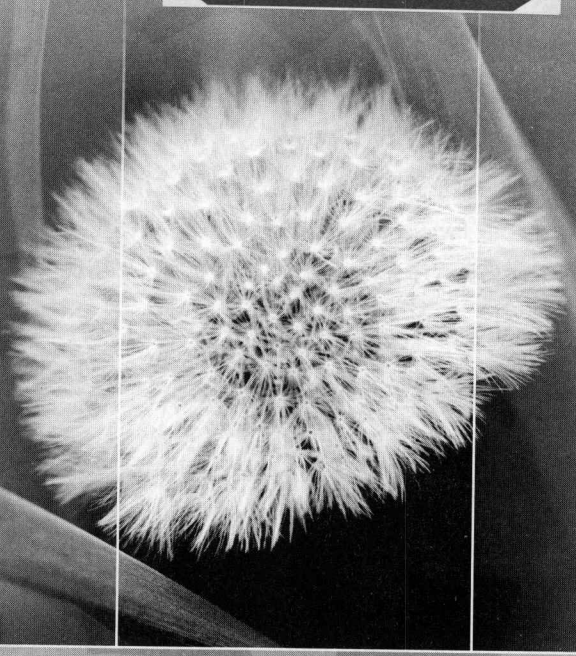


JAVASCRIPT
语言精髓与编程实践

周爱民 著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

动态语言技术
精品书廊



JAVASCRIPT
语言精髓与编程实践

周爱民 著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书详细讲述 JavaScript 作为一种混合式语言的各方面特性，包括过程式、面向对象、函数式和动态语言特性等，在动态函数式语言特性方面有着尤为细致的讲述。本书的主要努力之一，就是分解出这些语言原子，并重现将它们混合在一起的过程与方法。通过从复杂性到单一语言特性的还原过程，读者可了解到语言的本质，以及“层出不穷的语言特性”背后的真相。

本书主要的著述目的是基于一种形式上简单的语言来讲述“语言的本质及其应用”。本书详细讲述了通过框架执行过程来构造一个 JavaScript 扩展框架的方法，并完整地讲述了框架扩展中各种设计取舍，因此可以作为学习研究计算机程序设计语言时的参考，用以展示现实系统如何实现经典理论中的各种编程范型。

图书在版编目（CIP）数据

JAVASCRIPT 语言精髓与编程实践 / 周爱民著.—北京：电子工业出版社，2008.3
（动态语言技术精品书廊）
ISBN 978-7-121-05687-1
I. J… II.周… III.JAVA 语言—程序设计 IV.TP312

中国版本图书馆 CIP 数据核字（2007）第 199169 号

责任编辑：何　艳
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市金马印装有限公司
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：33.5　　字数：715 千字
印　　次：2008 年 3 月第 1 次印刷
印　　数：5000 册　　定价：68.00 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。

代 序

学两种语言

—《我的程序语言实践》节选—

《程序设计语言——实践之路》一书对“语言”有一个分类法，将语言分类为“说明式”与“命令式”两种。Delphi 以及 C、C++、Java、C#等都被分在“命令式”语言范型的范畴；“函数式”语言则是“说明式”范型中的一种。如今回顾自己对语言的学习，其实十年也就学会了两种语言：一种是命令式的 Pascal/Delphi，另一种则是说明式的 JavaScript。当然从语言的实现方式来看，一种是静态的，一种是动态的。

这便是我程序员生涯的全部了。

我毕竟不是计算机科学的研究者，而只是其应用的实践者，因此我从一开始就缺乏对“程序”的某些科学的或学术层面上的认识是很正常的。也许有些人一开始就认识到程序便是如此，或者一种语言就应当是这样构成和实现的，那么可能他是从计算机科学走向应用，故而比我了解得多些。而我，大概在十年前学习编程，以及在后来很多年的实践中，仅被要求“写出代码”，而从未被要求了解“什么是语言”。所以我才会后知后觉，才会在很长的时间里迷失于那些精细的、沟壑纵横的语言表面而不自知。然而一如我现在所见到，与我曾相同地行进于那些沟壑的朋友，仍然在持续地迷惑着、盲目着，全然无觉于沟壑之外的瑰丽与宏伟。

前些天写过一篇 BLOG，是推荐那篇“十年学会编程”的。那篇文章道出了我在十年编程实践之后，对程序语言的最深刻的感悟。我们学习语言其实不必太多，深入一两种就可以了。如果在一种类型的语言上翻来覆去，例如不断地学 C、Delphi、Java、C#……

无非是求生存、讨生活，或者用以装点个人简历，于编程能力的提高是不大的。更多的人，因为面临太多的语言选择而浅尝辄止，多年之后仍远离程序根本，成为书写代码的机器，把书写代码的行数、程序个数或编程年限作为简历中最显要的成果。这在明眼人看来，无过是熟练的砌砖工而已。

我在《大道至简》中说“如今我已经不再专注于语言”。其实在说完这句话之后，我就已经开始了对 JavaScript 的深入研究。在如此深入地研究一种语言，进而与另一种全然有别的语言比较补充之后，我对“程序＝算法＋结构”有了更深刻的理解与认识——尽管这句名言从来未因我的认识而变化过，从来未因说明与命令的编程方式而变化过，也从来未因动态与静态的实现方法而变化过。

动静之间，不变的是本质。我之所以写这篇文章，并非想说明这种本质是什么抑或如何得到，只是期望读者能在匆忙的行走中，时而停下了脚步，远远地观望一下目标罢了。

而我此刻，正在做一个驻足观望的路人甲。

周爱民

前　言

语言

语言是一种交流的工具，这约定了语言的“工具”本质，以及“交流”的功用。“工具”的选择只在于“功用”是否能达到，而不在于工具是什么。

在数千年之前，远古祭师手中的神杖就是他们与神交流的工具。祭师让世人相信他们敬畏的是神，而世人只需要相信那柄神杖。于是，假如祭师不小心丢掉了神杖，就可以堂而皇之地再做一根。甚至，他们可以随时将旧的换成更新或更旧的神杖，只要他们宣称这是一根更有利于通神的杖。对此，世人往往做出迷惑的表情，或者欢欣鼓舞的情状。今天，这种表情或情状一样地出现在大多数程序员的脸上，出现在他们听闻到新计算机语言被创生的时刻。

神杖换了，祭师还是祭师，世人还是会把头叩得山响。祭师掌握了与神交流的方法（如果真如同他们自己说的那样的话），而世人只看见了神杖。

所以，泛义的工具是文明的基础，而确指的工具却是愚人的器物。

计算机语言有很多种分类方法，例如高级语言或者低级语言。其中一种分类方法，就是“静态语言”和“动态语言”——事物就是如此，如果用一对绝对反义的词来分类，就相当于概含了事物的全体。当然，按照中国人中庸平和的观点，以及保守人士对未知可能性的假设，我们还可以设定一种中间态：半动态语言。你当然也可以叫它半静态语言，这个随便你。

所以，我们现在是在讨论一种很泛义的计算机语言工具。至少在眼下，它（在分类概念中）概含了计算机语言的二分之一。当然，限于我自身的能力，我只能讨论一种确指的工具，例如 JavaScript。但我希望你由此看到的是计算机编程方法的基础，而不是某种愚人的器物。JavaScript 的生命力可能足够顽强，我假定它比 C 还顽强，甚至比你我的

生命都顽强。但它只是愚人的器物，因此反过来说：它能不能长久地存在都并不重要，重要的是它能不能作为这“二分之一的泛义”来供我们讨论。

分类法

新打开一副扑克牌，我们总看到它被整齐的排在那里，从 A 到 K 及大小王。接下来，我们将它一分为二，然后交叉在一起；再分开，再交叉……但是在重新开局之前，你是否注意到：在上述过程中，牌局的复杂性其实不是由“分开”这个动作导致的，而是由“交叉”这个动作导致的。

所以分类法本身并不会导致复杂性。就如同一副新牌只有四套 A~K，我们可以按十三牌面来分类，也可以按四种花色来分类。当你从牌盒里把它们拿出来的时候，无论它们是以哪种方式分类的，这副牌都不混乱。混乱的起因，在于你交叉了这些分类。

同样的道理，如果世界上只有动态、静态两种语言，或者真有半动态语言而你又有明确的“分类法”，那么开发人员将会迎来清醒明朗的每一天：我们再也不需要花更多的时间去学习更多的古怪语言了。

然而，第一个问题便来自于分类本身。因为“非此即彼”的分类必然导致特性的缺失——如果没有这样“非此即彼”的标准就不可能形成分类，但特性的缺失又正是开发人员所不能容忍的。

我们一方面吃着碗里，一方面念着锅里。即使锅里漂起来的那片菜叶未见得有碗里的肉好吃，我们也一定要捞起来尝尝。而且大多数时候，由于我们吃肉吃腻了嘴，因此会觉得那片菜叶味道其实更好。所以首先是我们的个性，决定了我们做不成绝对的素食者或肉食者。

当然，更有一些人说我们的确需要一个新的东西来使我们更加强健。但不幸的是，大多数提出这种需求的人，都在寻求纯质银弹[1]或混合毒剂[2]。无论如何，他们要么相信总有一种事物是完美武器，或者更多的特性放在一起就变成了魔力的来源。

我不偏向两种方法之任一。但是我显然看到了这样的结果，前者是我们在不断地创造并特化某种特性，后者是我们在不断地混合种种特性。

更进一步地说，前者在产生新的分类法以试图让武器变得完美，后者则通过混淆不同的分类法，以期望通过突变而产生奇迹。

[1] 参见《人月神话》，美国弗雷德里克•布鲁克斯（Frederick P. Brooks, Jr.）著。
[2] 参见《蓝精灵》，比利时皮埃尔•居里福特（Pierre Culliford，Peyo）著。

二者相同之处，都在于需要更多的分类法。

函数式语言就是来源于另外的一种分类法。不过要说明的是，这种分类法是计算机语言的原力之一。基本上来说，这种分类法在电子计算机的实体出现以前就已经诞生了。这种分类法的基础是“运算产生结果，还是运算影响结果”。前一种思想产生了函数式语言（如 LISP）所在的“说明式语言”这一分类，后者则产生了我们现在常见的 C、C++ 等语言所在的“命令式语言”这一分类。

然而我们已经说过，人们需要更多的分类的目的，是要么找到类似银弹的完美武器，要么找到混合毒剂。所以一方面很多人宣称“函数式是语言的未来”，另一方面也有很多人把这种分类法与其他分类法混在一起，于是变成了我们这本书所要讲述的“动态函数式语言”——当然，毋庸置疑的是：还会有更多的混合法产生。因为保罗·格雷厄姆（Paul Graham）[3]已经做过这样的总结：

> 二十年来，开发新编程语言的一个流行的秘诀是：取 C 语言的计算模式，逐渐地往上加 LISP 模式的特性，例如运行时类型和无用单元收集。

然而这毕竟只是“创生一种新语言”的魔法。那么，到底有没有让我们在这浩如烟海的语言家族中，找到学习方法的魔法呢？

我的答案是：看清语言的本质，而不是试图学会一门语言。当然，这看起来非常概念化。甚至有人说我可能是从某本教材中抄来的，另外一些人又说我试图在这本书里宣讲类似于我那本《大道至简》里的老庄学说[4]。

其实这很冤枉。我想表达的意思不过是：如果你想把一副牌理顺，最好的法子，是回到它的分类法上，要么从 A 到 K 整理，要么按四个花色整理[5]。毕竟，两种或更多种分类法作用于同一事物，只会使事物混淆而不是弄得更清楚。

因此，本书从语言特性出发，把动态与静态、函数式与非函数式的语言特性分列出来。先讲述每种特性，然后再讨论如何去使用（例如交叉）它们。

特性

无论哪种语言（或其他工具）都有其独特的特性，以及借鉴自其他语言的特性。有些语言通体没有“独特特性”，只是另外一种语言的副本，这更多的时候是为了“满足一

[3] 保罗•格雷厄姆是计算机程序语言 Arc 的设计者，著有多本关于程序语言，以及创业方面的书籍。
[4] 这是一本软件工程方面的书，但往往被人看成是医学书籍或有人希望从中求取养生之道。
[5] 不过这都将漏掉了两张王牌。这正是问题之所在，因为如果寻求“绝对一分为二的方法”，那么应该分为“王牌”和“非王牌”。但这往往不被程序员或扑克牌玩家们采用，因为极端复杂性才是他们的毕生目标。

些人使用语言的习惯”。还有一些语言则基本上全是独特的特性，这可能导致语言本身不实用，但却是其他语言的思想库。

我们已经讨论过这一切的来源。

对于 JavaScript 来说，除了动态语言的基本特性之外，它还有着与其创生时代背景密切相关的一些语言特性。直到昨天[6]，JavaScript 的创建者还在小心翼翼地增补着它的语言特性。JavaScript 轻量的、简洁的、直指语言本实的特性集设计，使它成为解剖动态语言的有效工具。这个特性集包括：

- 一套参考过程式语言惯例的语法；
- 一套以原型继承为基础的对象系统；
- 一套支持自动转换的弱类型系统；
- 动态语言与函数式语言的基本特性。

需要强调的是，JavaScript 1.x 非常苛刻地保证这些特性是相应语言领域中的最小特性集（或称之为“语言原子”），这些特性在 JavaScript 中相互混合，通过交错与补充而构成了丰富的、属于 JavaScript 自身的语言特性。

本书的主要努力之一，就是分解出这些语言原子，并重现将它们混合在一起的过程与方法。通过从复杂性到单一语言特性的还原过程，让读者了解到语言的本实，以及“层出不穷的语言特性”背后的真相。

技巧

技巧是“技术的取巧之处”，所以根本上来说，技巧也是技术的一部分。很多人（也包括我）反对技巧的使用，是因为难以控制，并且容易破坏代码的可读性。

哪种情况下代码是需要“易于控制”和“可读性强”的呢？通常，我们认为在较大型的工程下需要“更好的控制代码”；在更多人共同开发的项目代码上要求“更好的可读性”。然而，反过来说，在一些更小型的、不需要更多人参与的项目中，“适度的”使用技巧是否就可以接受呢？

这取决于“需要、能够”维护这个代码的人对技巧的理解。这包括：

- 技巧是一种语言特性，还是仅特定版本所支持或根本就是 BUG；
- 技巧是不是唯一可行的选择，有没有不需要技巧的实现；

[6] 在 JavaScript 2——这种把银弹涂上毒剂以试图用单发手枪击杀恐龙的构想发布之前的“昨天”。

- 技巧是为了实现功能，而不是为了表现技巧而出现在代码中的。

即使如此，我仍然希望每一个技巧的使用都有说明，甚至示例。如果维护代码的人不能理解该技巧，那么连代码本身都失去了价值，更何论技巧存在这份代码中的意义呢？

所以本书中的例子的确要用到许多“技巧”，但我一方面希望读者能明白，这是语言内核或框架内核实现过程中必须的，另一方面也希望读者能从这些技巧中学习到它原本的技术和理论，以及活用的方法。

然而对于很多人来说，本书在讲述一个完全不同的语言类型。在这种类型的语言中，本书所讲述的一切，都只不过是“正常的方法”；在其他类型的一些语言中，这些看起来就成了技巧。例如在 JavaScript 中要改变一个对象方法指向的代码非常容易，并且是语言本身赋予的能力；而在 Delphi/C++中，却成了“破坏面向对象设计”的非正常手段。

所以你最好能换一个角度来看待本书中讲述的“方法”。无论它对你产生多大的冲击，你应该先想到的是这些方法的价值，而不是它对于“你所认为的传统”的挑战。事实上，这些方法，在另一些“同样传统”的语言类型中，已经存在了足够长久的时间——如同“方法”之与“对象”一样，原本就是那样“（至少看起来）自然而然”地存在于它所在的语言体系之中。

语言特性的价值依赖于环境而得彰显。横行的螃蟹看起来古怪，但据说那是为了适应一次地磁反转。螃蟹的成功在于适应了一次反转，失败（我们是说导致它这样难看）之处，也在于未能又一次反转回来。

这本书

你当然可以置疑：为什么要有这样的一本书？是的，这的确是一个很好的问题。

首先，这本书并不讲 Web 浏览器（Web Browser，例如 Internet Explorer）。这可能令人沮丧。但的确如此。尽管在很多人看来，JavaScript 就是为浏览器而准备的一种轻量的语言，并认为它离开了 DOM、HTML、CSS 就没有意义。在同样的“看法”之下，国内外的书籍在谈及 JavaScript 时，大多会从“如何在 Web 页面上验证一个输入框值的有效性”讲起。

是的，最初我也是这样认为的。因为本书原来就是出自我在写《B 端开发》一书的过程之中。《B 端开发》是一本讲述“在浏览器（Browser）上如何用 JavaScript 开发”的书。然而，《B 端开发》写到近百页就放下了，因为我觉得应该写一本专门讲 JavaScript 的书，这更重要。

所以，现在你将要看到的这本书就与浏览器无关。在本书中我会把 JavaScript 提升到与 Java、C#或 Delphi 一样的高度，来讲述它的语言实现与扩展。尽管在本书的最后一部分内容中，讲述了名为“Qomo”的完整的 JavaScript 框架，这有助于你在浏览器上构建大型应用[7]。但是，嗯，本书不讲浏览器，不讲 Web，也并不讲“通常概念下的”AJAX。

JavaScript 是一门语言，有思想的、有内涵的、有灵魂的语言。如果你不意识到这一点，那么你可能永远都只能拿它来做那个“验证一个输入框值的有效性”的代码。本书讲述 JavaScript 的这些思想、核心、灵魂，以及如何去丰富它的血肉。最为核心的内容是在第 2 至 6 章，包括：

- 以命令式为主的一般化的 JavaScript 语言特性，以及其对象系统；
- 动态、函数式语言，以及其他语言特性在 JavaScript 的表现与应用；
- 使用动态函数式特性来扩展 JavaScript 的特性与框架。

在撰述这些内容的整个过程中，我一直在试图给这本书找到一个适合的读者群体，但我发现很难。因为通常的定义是低级、中级与高级，然而不同的用户对自己的“等级”的定义标准并不一样。在这其中，有“十年学会编程”的谦谨者，也有“三天学会某某语言”的速成家。所以，我认为这样定位读者的方式是徒劳的。

如果你想知道自己是否适合读这本书，建议你先看一下目录，然后试读一二章节，可以先选读一些在你的知识库中看来很新鲜的，以及一些你自认为已经非常了解的内容。通过对比，你应该知道这本书会给你带来什么。

不过我需要强调一些东西。这本书不是一本让你“学会某某语言”的书，也不是一本让初学者“学会编程”的书。阅读本书，你至少应该有一点编程经验（例如半年至一年），而且要摈弃某些偏见（例如 C 语言天下无敌或 JavaScript 是新手玩具）。

最后，你至少要有一点耐心与时间。

[7] 如果你试图构建基于 Web 的大型应用（例如基于 AJAX 的工程），那么你可以从 Qomo 中得益良多。它可以成倍地提高你的开发工效，有利于你实现更多的、更有价值的应用特性。（广告结束）

封面说明

本书封面主色为绿色，彰显青春的活力——充满希望和朝气，贴近JavaScript 语言的开源本质。封面底纹为中国的传统艺术——剪纸，该剪纸作品取自袁荃猷先生的《游刃集》（三联书店出版，2002 年），寓意本书是由本土技术专家倾力而作的原创精品图书。

本书的封面素材图案为蒲公英。蒲公英轻盈、透明、复杂而有序，看视平凡、普通，却有着顽强的生命力，更有着了不起的传播力，一阵轻风吹过，就能飞往各处，蓬勃而旺盛地成片生长。本书美编挑选蒲公英作为封面图案，既希望能借此表达出 JavaScript 语言的动态、轻型、有规律的特点，也希望作者的心血之作能借蒲公英蓬勃旺盛的生命力，四处传播，落地生根，为读者全面展示 JavaScript 语言的精髓和编程实践。（责编：何艳）

目 录