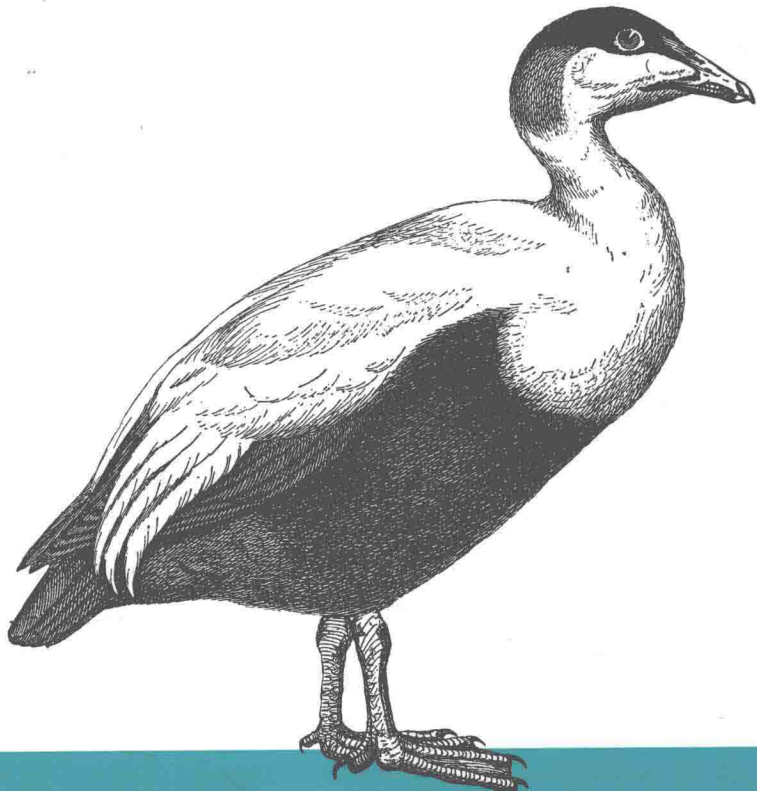
Functional JavaScript



JavaScript 函数式编程

[美] *Michael Fogus* 著
欧阳继超 王妮 译

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

O'REILLY®

JavaScript 函数式编程

[美] Michael Fogus 著
欧阳继超 王 妮 译

人 民 邮 电 出 版 社
北 京

图书在版编目（CIP）数据

JavaScript函数式编程 /（美）佛格斯（Fogus, M.）著 ；欧阳继超，王妮译. -- 北京 ：人民邮电出版社，2015.8
ISBN 978-7-115-39060-8

Ⅰ. ①J… Ⅱ. ①佛… ②欧… ③王… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第125259号

版权声明

LED/OLED 技术与应用丛书

◆ 著　[美] Michael Fogus
译　欧阳继超　王　妮
责任编辑　陈冀康
责任印制　张佳莹　焦志炜

◆ 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
三河市海波印务有限公司印刷

◆ 开本：787×1000　1/16
印张：14
字数：258 千字　2015 年 8 月第 1 版
印数：1-3 000 册　2015 年 8 月河北第 1 次印刷

著作权合同登记号　图字：01-2013-7657 号

定价：49.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315

内容提要

JavaScript 是近年来非常受瞩目的一门编程语言，它既支持面向对象编程，也支持函数式编程。本书专门介绍 JavaScript 函数式编程的特性。

全书共 9 章，分别介绍了 JavaScript 函数式编程、一等函数与 Applicative 编程、变量的作用域和闭包、高阶函数、由函数构建函数、递归、纯度和不变性以及更改政策、基于流的编程、无类编程。除此之外，附录中还介绍了其他和 JavaScript 函数式编程相关的知识。

本书内容全面，示例丰富，适合想要了解函数式编程的 JavaScript 程序员和学习 JavaScript 的函数式程序员阅读。

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

序一

这是一本令人兴奋的书。

尽管开始是想作为嵌入 HTML 文档中的精简版 Java（Java-lite）的脚本语言，由此可以进行一点交互，但 JavaScript 却成为通用编程最灵活的语言之一。

你可以随意根据最适合你的特定方式来写 JavaScript。在 JavaScript 中这样做比其他更刚性的语言更自然，究其原因，是支撑 JavaScript 的核心理念：更大程度上比面向对象语言如 Ruby 和 Java 扩展了一切都是对象（一切都是一个值）的理念。函数既是对象，也是值。任何对象都可以作为其他对象的原型（默认值）。函数只有一种，你可以随意使用它，可以作为一个纯函数、一个突变过程，或作为一个对象的方法。

JavaScript 可以但不会强制使用不同的编程风格。早期，我们倾向于把传统期望和“最佳”实践套用到 JavaScript 的学习中。当然，这会使得 JavaScript 很像没有类型的 Java，或是将类型写到每个方法上面的注释里面。渐渐地，有人开始实验：在运行时生成函数，使用不可变的数据结构，创建不同于面向对象的设计模式，发现链式 API 的魔力，或是扩展内置原型来得到定制化的功能。

我最近特别热衷于使用函数式思路构建丰富的 JavaScript 应用。随着 JavaScript 从简单的表单验证和 DOM 动画到全功能的应用程序，JavaScript 开始面临着各种特定问题，函数式也有了更有趣的舞台，如下所示。

- 通过可部分配置参数的函数构建大量 API。
- 使用递归函数来平滑需要一段时间的事件。
- 使用无突变（mutation-free）的随意替换的管道构建复杂的业务逻辑。

本书正是探索这片领域的理想书籍。在书中的 9 章和附录里，友好的导游和疯狂的科学家 Michael Fogus，将函数式编程层层剥开，让你一探究竟。一般很少有编程书籍能带给读者惊喜，但是这一本绝对可以。

好好享受吧！

Jeremy Ashkenas

序二

我还记得当我第一次读到 Douglas Crockford 的《JavaScript 精粹》时，我不仅从中学到了东西，而且 Crockford 只用了 172 页，就能带领读者避开 JavaScript 的各种问题，实在令人印象深刻。Crockford 的书既简洁，又能让读者充分消化并从中受益。

接下来，你会发现，Michael Fogus 给了我们一本类似 Crockford 的书。他吸收了 Crockford 以及其他前辈的中肯的意见，带我们深入函数式 JavaScript 编程的世界。我经常听说或看到（甚至我自己也会写到），JavaScript 是一种函数式编程语言，但这种说法（包括我自己）都似乎难以领会。甚至 Crockford 也只用了一章阐述函数，像许多作家一样，都集中于 JavaScript 的对象支持。Fogus 填补了这些重要的细节。

函数式编程从一开始就是计算机领域的重要的一部分，但它一直没有受到实践软件人员的广泛关注。但由于计算硬件速度和容量的不断提高，再加上我们的行业在创造并发、分布和大规模软件系统的需求不断扩大，函数式编程正迅速地普及。能获得这样的增长是因为对于开发者来说，函数式更容易推理、构建和维护。对函数式编程语言，如 Scala、Clojure 中，Erlang 和 Haskell 的关注也达到了历史新高，并仍在增加，至今仍不见削减。

当你读完 Michael 具有深刻见解的 JavaScript 的函数式编程，你会对他所提供的信息的深度和广度留下深刻的印象。他首先会让事情保持简单，解释如何避免使用 JavaScript 功能强大的对象原型系统，而使用函数和“数据抽象”来建模类的方式。在之后的章节中，解释了函数式数据转换的简单模型可以产生复杂而高效的更高层次的抽象。我猜你会随着 Fogus 的每个章节对这种方式的层次深入感到惊讶。

大部分软件开发工作需要实用主义，好在 Fogus 也强调了这一项。如果不实用，就算有优雅、精致和简洁的代码，最终都是毫无意义的。这也是函数式编程隐藏在阴影中这么多年的很大一部分原因。Fogus 通过帮助读者了解和评估与函数式编程相关的计算成本来解决这一问题。

当然，书就像软件，都少不了沟通。就像 Crockford 和 Fogus 的写作方式，既简短，又内容翔实，恰到好处。我没有夸大 Michael 的简洁和清晰的重要性，不然你会失

去他所提供的令人难以置信的想法和见解的。你会发现，不仅 Fogus 所提供的方法和代码优雅，他表达的方式也一样优雅。

Steve Vinoski

前言

什么是 Underscore

Underscore.js（以下简称 Underscore）是支持函数式编程的 JavaScript 库。Underscore 网站是这样描述的：

> Underscore 为 JavaScript 提供了大量的函数式编程的支持，类似 Prototype.js（或 Ruby）的 utility-belt，但没有扩展 JavaScript 内置对象。

“utility belt”指的是一套能帮助你解决很多常见问题的工具。

获取 Underscore

Underscore 网站上有最新的版本。你可以从网站上下载并放入应用目录。

使用 Underscore

你可以像使用所有其他库一样在你的项目中使用 Underscore。然而，有几点需要注意的是，首先，默认情况下，Underscore 定义了一个包含其所有函数的全局对象。要调用一个 Underscore 的函数，只需要调用“_”里的方法，如下面的代码：

```
_.times(4, function() { console.log("Major") });

// (console) Major
// (console) Major
// (console) Major
// (console) Major
```

很简单吧？

但如果你已经定义一个全局_变量，事情就没这么简单了。在这种情况下，Underscore 提供了一个`_.noConflict`函数，将重新绑定旧的_，并返回 Underscore 的引用。`_.noConflict`使用方式如下：

```
var underscore = _.noConflict();

underscore.times(4, function() { console.log("Major") });

// (console) Major
```

```
// (console) Major
// (console) Major
// (console) Major

_;
//=> Whatever you originally bound _ to
```

本书会介绍更多 Underscore 的细节，但记住，虽然我广泛使用（并认可）Underscore，但这并不是一本关于 Underscore 的书。

函数式 JavaScript 的源代码

许多年前，我想写基于函数式编程技术的 JavaScript 库。跟许多程序员一样，我曾通过实验、实践以及阅读 Douglas Crockford 的文章对 JavaScript 有所认识。虽然我继续完成了我的函数式库（Doris），但甚至连我自己都很少用它。

完成 Doris 后，我继续尝试广泛的函数式编程语言如 Scala 和 Clojure。此外，我花了很多时间编写 ClojureScript，特别是它的 JavaScript 编译器。基于这些经验，我非常了解函数式编程技术。因此，我决定尝试使用随后这几年学到的技术复活 Doris。我将其命名为 Lemonad，最后几乎是与本书同时完成的。

本书中大多数函数都是为了教学目的，但我扩展了一些并贡献到我的 Lemonad 库，以及 Underscore-contrib 库。

本书中的代码

本书的源代码可以在 GitHub 上获取。此外，你也可以进入本书的网址使用上面的 REPL 试试本书所有定义的函数。

符号约定

在编写本书（和一般编写 JavaScript）的过程中，我得出以下比较好的约定。

- 避免多次赋值变量。
- 不要使用 `eval`[①]。
- 不要修改内核对象如 `Array` 和 `Function`。
- 优先使用函数而不是方法。

① 跟所有强大的工具一样，eval 和 Function 常量都是双刃剑，我并不反对使用，但是建议尽可能少用。

- 如果项目一开始就定义函数，那么在接下来的阶段里也应该如此。

此外，我在本书中还用到了一些约定。

- 零参数的函数用于表示该参数并不重要。
- 在一些例子中，……用来表示其周围的代码段可忽略。
- `inst#method` 表示实例方法引用。
- `Object.method` 表示类型方法。
- 我倾向于用单行 `if/else` 语句，避免使用大括号。这样可以节省宝贵的垂直空间。
- 我喜欢用分号。

基本上除了函数式方面，这本书中的 JavaScript 代码就像现实中的大多数的 JavaScript 代码。

本书目标读者

我在几年前写一本 Scheme 编程语言的函数式编程的入门书籍的时候，就产生了写这本书的想法。尽管 Scheme 和 JavaScript 有一些共同的特点，但在许多重要方面截然不同。我想撇开语言来说说函数式编程。我写这本书介绍函数式编程是什么，什么是不可能在 JavaScript 中找到的。

我期望读者对 JavaScript 有基本的理解。可以通过很多书籍以及网上的资源和讨论来学习这门语言，这里就不占用本书篇幅介绍了。我还期望读者能对面向对象编程有所了解，如 Java 和 Ruby，Python 和 JavaScript。了解面向对象编程可以帮助你理解我偶尔使用的一些短语，但并不需要专家级的了解。

本书的合适读者是希望了解函数式编程的 JavaScript 程序员，或希望学习 JavaScript 的函数式程序员。对于后一种读者，还可以研究一些 JavaScript 的……古怪的部分，特别是可以参考 Douglas Crockford 的《JavaScript 精粹》（O'Reilly 出版）。最后，这本书还适合任何希望了解函数式编程，包括不打算使用 JavaScript 的读者。

本书组织结构

下面是 JavaScript 函数式编程的大纲。

第 1 章　JavaScript 函数式编程简介

这本书通过引入一些主题来开始，包括函数式编程和 Underscore.js。

第 2 章　一等函数与 Applicative 编程

第 2 章定义了一等函数，展示如何使用它们，并介绍了一些常见的应用。其中介绍了一个特别的技术，即利用一等函数实现 Applicative 编程。本章结尾还对软件开发中函数式编程的重要途径，即“数据思想”进行了探讨。

第 3 章　变量的作用域和闭包

第 3 章是一个过渡章，涵盖要了解函数式 JavaScript 编程需要注意的两个核心主题。通过覆盖变量作用域，包括在 JavaScript 中使用的方式：词法作用域，动态作用域和函数作用域。本章以闭包的介绍结尾，解释了工作原理，以及如何和为什么可能需要使用闭包。

第 4 章　高阶函数

本章建立在第 2、3 章基础上，介绍了一个重要的一等函数：高阶函数。虽然“高阶函数”听起来很复杂，本章会说明它其实是很直白的。

第 5 章　由函数构建函数

本章介绍了如何用其他函数“组合”新函数。组合函数是函数式编程的重要技术，本章将引导你了解这项技术。

第 6 章　递归

第 6 章是另一个过渡章节，将讨论递归，即一个直接或间接调用自身的函数。因为递归在 JavaScript 中是有局限的，因此不被经常使用；但是，本章会介绍几个绕过这些局限的方法。

第 7 章　纯度、不变性和更改政策

第 7 章介绍如何编写不会改变任何东西的函数。简单地说，函数式编程的便利性来源于不可变变量。本章将带你理解其中的含义。

第 8 章　基于流的编程

第 8 章涉及如何将任务甚至是整个系统，看作变换数据的“装配线”。

第 9 章　无类编程

最后一章的重点是介绍函数式编程是完全不同于基于类的面向对象编程的结构化应用程序的方式。

在这些章节之后补充了附录 A。

本书使用的约定

本书使用以下字体排版约定。

斜体

表示新的术语、网址、电子邮件地址、文件名和文件扩展名。

等宽字体

用于程序代码清单，出现在段落之内则表示程序中的元素，如变量、函数名、数据库、数据类型、环境变量、程序语句和关键字。

等宽加粗体

显示命令或其他应当由用户键入的文本。

等宽斜体

表示该文本应当更换为用户提供的值或者由上下文所决定的值。

示例代码的使用

本书的目的是为了帮助你完成工作任务。在一般情况下，这本书中包括的代码示例，你可以将其应用到你的程序和文档中。除非需要复制这些示例代码的相当大部分，否则无需联系我们以获得许可。比如说，当你编写的程序用到了本书中的若干示例代码，这并不需要特别许可。但是，销售或分发含有 O'Reilly 书籍附带的示例程序的光盘则需要获得许可。当你在回答他人问题时援引本书内容，或者引用书中的范例代码，也不用申请许可；而如果要把本书中的代码大量地引用到你的产品文档中，则需要许可。

对于引用时署名本书，我们表示感谢，但并不要求。一个署名通常包括标题、作者、出版商和 ISBN，例如“Functional JavaScript Michael Fogus（O'Reilly 出版）。版权所有 2013 Michael · Fogus，978-1-449-36072-6”。

如果你觉得你使用示例代码的情况超出了以上描述的不需要许可的范围，请随时联系我们：permissions@oreilly.com。

Safari®在线图书

记录

Safari 在线图书（www.safaribooksonline.com）是一个虚拟图书馆，让你可以轻松搜寻成千上万的顶尖技术书籍。

科技人才，软件开发者，网页设计师，以及商业和创意专业人士都将 Safari 在线图书作为研究、解决问题、学习和认证培训的主要资源。

Safari 的联机丛书提供了一系列的产品并为组织、政府机构和个人提供不同定价。用户可以访问和搜索出版社 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 等的数字内容。有关 Safari 在线图书的更多信息，请访问我们的在线网站。

如何联系我们

请将对本书的有关意见和问题告知出版商：

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

我们为这本书制作了网页，其中包含了勘误表、范例，以及其他补充资料。你可以通过这个地址访问：http://oreilly/functional_js。

请发送电子邮件至 bookquestions@oreilly.com 发表评论或询问关于本书的技术问题。

更多关于这本书、课程、会议和新闻的信息，查看下面的网站：http://www.oreilly.com。

我们的 Facebook 网站：http://facebook.com/oreilly。

在 Twitter 上 follow 我们：http://twitter.com/oreillymedia。

YouTube 上的频道：http://www.youtube.com/oreillymedia。

致谢

写一本书需要各方面的支持和努力，本书也不例外。首先，我要感谢我的好朋友 Rob Friesel 抽出时间提供反馈意见。此外，我要感谢 Jeremy Ashkenas 把我介绍给 O'Reilly，能让本书得以出版。而且是他写了非同小可的 Underscore.js 库。

我还要感谢这些人给我的反馈和启发：Chris Houser、David Nolen、Stuart Halloway、Tim Ewald、Russ Olsen、Alan Kay、Peter Seibel、Sam Aaron、Brenton Ashworth、Craig Andera、Lynn Grogan、Matthew Flatt、Brian McKenna、Bodil Stokke、Oleg Kiselyov、Dave Herman、Mashaaricda Barmajada ee Mahmud、Patrick Logan、Alan Dipert、Alex Redington、Justin Gehtland、Carin Meier、Phil Bagwell、Steve Vinoski、Reginald Braithwaite、Daniel Friedman、Jamie Kite、William Byrd、Larry Albright、Michael Nygard、Sacha Chua、Daniel Spiewak、Christophe Grand、Sam Aaron、Meikel Brandmeyer、Dean Wampler、Clinton Dreisbach、Matthew Podwysocki、Steve Yegge、David Liebke、Rich Hickey。

我编写本书时的配乐由 Pantha du Prince、Black Ace、Brian Eno、Béla Bartók、Dieter Moebius、Sun Ra、Broadcast、Scientist、John Coltrane 提供。

最后，所有这一切都要感谢我生命中的三位挚爱：Keita、Shota、Yuki。

目录