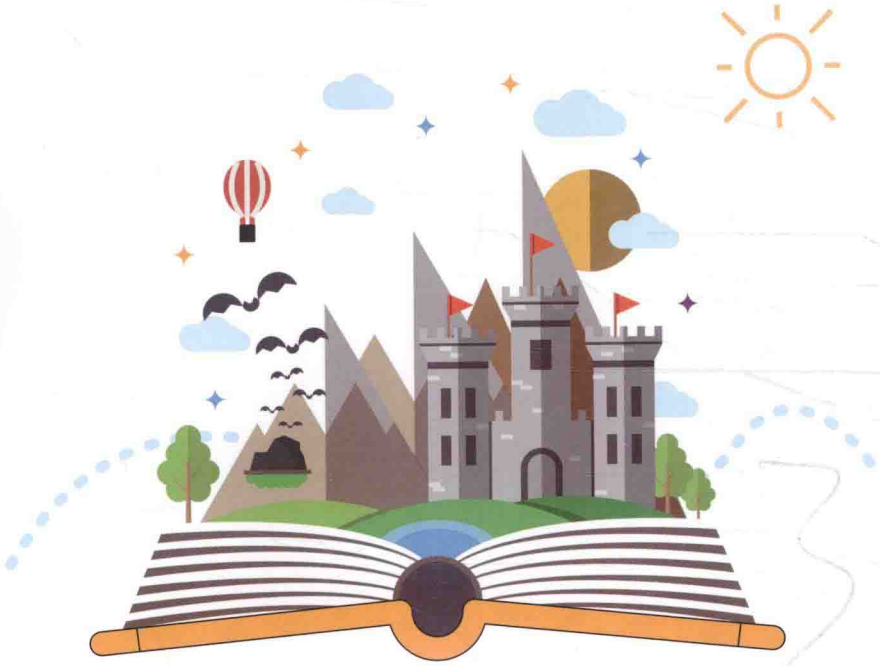
Broadview®
www.broadview.com.cn



前端

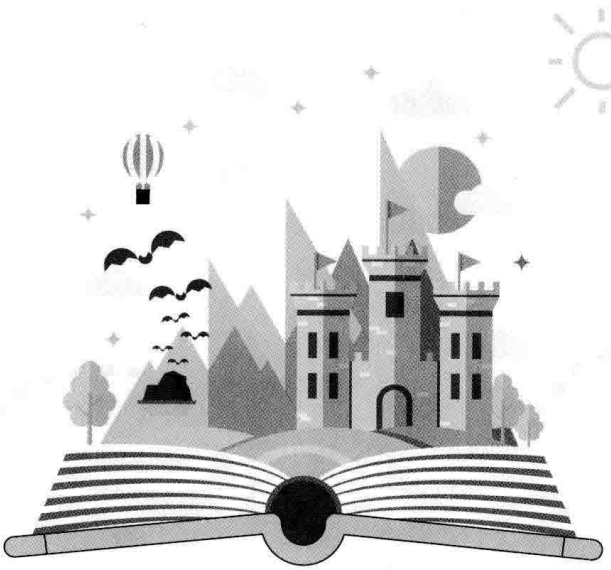
函数式攻城指南

欧阳继超 / 著

作为Ramda.js的作者之一，我非常喜欢Braithwaite的 *JavaScript Allonge*，喜欢Fogus的 *Functional JavaScript*，我非常激动又有一本关于JavaScript的函数式书籍，希望尽快能见到这本书的英文版。

CrossEye

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn



前端

函数式攻城指南

欧阳继超 / 著

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

内 容 简 介

在后端，函数式语言层出不穷。在前端，函数式最后的边界也已经被渐渐打破。Scala 的 Scala.js、Clojure 的 ClojureScript 都试图同构移向前端。然而，原生 JavaScript 其实也可以通过丰富的库让前端的函数式编程一样的舒适和优雅。

本书涵盖了大部分函数式编程思想，包括 JavaScript 的函数式支持，Clojure 风格的集合、递归、函数组合、宏、模式匹配、实用的 Monads，以及前端的并发编程。

本书适合想要了解函数式编程的 JavaScript 程序员或者想学习 JavaScript 的函数式程序员阅读。

图书在版编目（CIP）数据

前端函数式攻城指南 / 欧阳继超著. —北京：电子工业出版社，2016.10
ISBN 978-7-121-29724-3

Ⅰ. ①前… Ⅱ. ①欧… Ⅲ. ①函数－程序设计 Ⅳ. ①TP311.1

中国版本图书馆 CIP 数据核字（2016）第 200135 号

责任编辑：董　英
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
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：12　　字数：187 千字
版　　次：2016 年 10 月第 1 版
印　　次：2016 年 10 月第 1 次印刷
印　　数：3000 册　　定价：59.00 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）88254888，88258888

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。

本书咨询联系方式：010-51260888-819　faq@phei.com.cn。

序

函数式编程可以说是非常古老的编程方式，但是近年来函数式编程越来越受到人们的关注。不管是 Google 力推的 Go、学术派的 Scala 与 Haskell，还是 Lisp 的新方言 Clojure，这些新的函数式编程语言都越来越受到人们的关注。

当然不仅是后端函数式编程语言层出不穷，前端也不甘示弱。虽然前端浏览器只支持一门语言——JavaScript，但是能支持函数式编程的 JavaScript 库越来越多，比如 Functional JavaScript[1]、Underscore、lodash 等。不仅如此，还有一些能编译成 JavaScript 的语言，能让前端的函数式编程发挥到极致，例如 Haskell 的 PureScript、Scala 的 Scalajs、Clojure 的 ClojureScript。

我两次都以 Clojure 结尾，是因为我喜欢把重点留到最后。Clojure 独特于其他语言，它既是一门新的语言、一门函数式编程范式的语言，又流淌着古老的血液——Lisp[2]。这是我选择用 Clojure 来诠释函数式编程的原因之一。

那么为什么我要选 JavaScript 作为函数式编程的目标呢？Michael Fogus 用 200 多页向大家展示了不一样的 *Functional JavaScript* 编程方式，可惜 Fogus 作为 ClojureScript 编译器的贡献者，竟然选择了 Underscore 作为函数式库，直接导致并不能完全展示 JavaScript 所能达到的函数式编程能力。有趣的是，ClojureScript 的作

---

1 没错，名字就叫这个。

2 还记得《计算机程序构造与解释》里面用的怪怪的括号语言 Scheme 吗？

者把 ClojureScript 的不可变（Immutable）数据结构移植到了 JavaScript，这彻底将 JavaScript 的函数式编程提升到了用其他库都完成不了的新高度[3]。不仅如此，Mozilla 的 Sweet.js（https://github.com/mozilla/sweet.js）更是完成了另一个突破——JavaScript 的 macro，它虽然不能算是函数式的概念，但也算是 Lisp 语言的一项独门绝技了[4]。

这一切的一切，都让我忍不住要帮 Fogus 出一本续集，用 JavaScript 实现其他函数式编程语言如 Clojure 甚至是 Haskell[5]的奇技淫巧，让大家进一步感受用 JavaScript 这门不完美的语言同样可以编写出优雅的函数式代码，以不一样的方式思考和解决问题。于是不管你是想转行 JavaScript 的 Clojure 开发者，还是想了解 Clojure 或函数式编程的 JavaScript 开发者，都可以在此找到一些启发。但这并不是一本 JavaScript 入门的好书[6]。

3 虽然 Facebook 也尝试实现自己的一个不可变数据结构 Immutable.js（https://facebook.github.io/immutable-js）。

4 虽然很多新的语言都在尝试实现 macro，比如 Scala、Rust 和 Julia，但是语法过于复杂。

5 就当是 bonus 吧。

6 Douglas 的《JavaScript 语言精粹》是个不错的选择。另外如果读者的英文好的话，还有一本可以在线免费阅读的 *JavaScript Allonge*（https://leanpub.com/javascriptallongesix/read）。

前言

1. 看本书之前你要知道

1）最好能看懂 JavaScript 代码

这既不是一本介绍 Clojure 的书，也不是一本介绍 JavaScript 的书，而是一本介绍如何用 JavaScript 函数式编程的书。其中一些函数式的思想和表现形式都借用了 Clojure，因此叫作 Clojure 风格的函数式 JavaScript，但是并不要求在读本书前会 Clojure[1]，而只需要能阅读 JavaScript 代码。如果你会 Clojure，则可以完全忽略我解释 Clojure 代码的段落，当然 JavaScript 的部分才是重点。

2）你可能买错书了，如果你

- 想学 JavaScript

这不是一本 JavaScript 的教科书，这里只会介绍如何用 JavaScript 进行函数式编程，所以如果想要系统地学习 JavaScript，则学习《JavaScript 语言精粹》可能已经足够了。另外，如果读者的英文水平好的话，则还有一本可以在线免费阅读的 *JavaScript Allonge*。

- 想学 Clojure

同样，这也不是一本 Clojure 的教科书，只含有一些用于阐述函数式编程思想

---

1 就像计算机程序构造与解释中说的，Lisp 语言基本没有语法，就像学习象棋的规则只用花很少的时间，而如何下好棋，才是学习的关键，也是乐趣所在。

的 Clojure 代码。你确实可以学到一些 Clojure 编程知识，但很可能是非常零碎且不完整的知识。如果想要系统地了解和学习 Clojure，则非常推荐你阅读 *The Joy of Clojure*[2]，另外，如果读者英文比较好，还有一本可以免费在线阅读的 *CLOJURE for the BRAVE and TRUE* 也非常不错。

- 是函数式编程的专家

如果你已经在日常工作或学习中使用 Scala、Clojure 或者 Haskell 等函数式语言编程的话，那么本书对你在函数式编程上的帮助不会太大。**不过**：这本书对缓解你从函数式语言迁移到 JavaScript 编程的不适应该是非常有效的。这也正是本书的目的之一。

2．准备环境

在开始阅读本书之前，如果你希望能运行书中的代码，可能需要一些环境的配置。而且书中的所有源码和运行方式都可以在本书的 Github 仓库[3]中找到。当然如果你使用 Emacs（同时还配置了 org babel 的话）阅读本书的源码，对于大部分代码只需要光标放在代码处按 c-c c-c 即可。

- JavaScript

原生的 JavaScript 没有什么好准备的，可以通过 Node 或者 Firefox（推荐）的 Console 运行代码。当然第 5 章会有一些使用 Sweet.js 写的 Macro，这则需要安装 Sweet.js。

– 安装 Node/io.js

（1）下载 Node.js。

（2）如果使用 Mac，可以直接用 Brew 安装。

---

2 中文叫《Clojure 编程乐趣》，但是只有第一版的，原书已经第二版了。我刚好有幸翻译了作者 Michael Fogus 的另一本书《JavaScript 函数式编程》。

3 https://github.com/jcouyang/clojure-flavored-javascript/tree/source。

```
brew install node
# 或者
brew install iojs
```

– 安装 Sweet.js

在安装完 Node.js 之后在命令行输入：

```
npm install -g sweet.js
```

- Clojure

书中的 Clojure 代码大都用来描述函数式编程的概念，当然如果想要运行书中的 Clojure 代码，首先需要安装 JVM 或者 JDK[4]，至少需要 1.6 版本，推荐安装 1.8 版本。

– 安装 leiningen

leiningen 是 Clojure 的包管理工具，类似于 Node 的 Npm、Ruby 的 bundle、Python 的 pip。另外 leiningen 还提供脚手架的功能。可以通过官网的脚本安装[5]。Mac 用户可以简单地使用 brew install leiningen 安装。

安装完成之后，就可以运行 lein repl，打开 repl，试试输入下列 Clojure 代码，你将会立马看见结果。

```
(+ 1 1)
;=> 2
```

– 编辑器

如果更喜欢使用编辑器来编辑更长的一段代码，我推荐非 Emacs 用户使用 Light Table[6]，Intellij 用户则使用 cursive[7]。当然如果读者已经在使用 Emacs，那就更

4 http://www.oracle.com/technetwork/java/javase/downloads/index.html。

5 http://leiningen.org。

6 http://lighttable.com/。

7 https://cursive-ide.com/。

完美了，Emacs cider mode[8]是 Clojure 编程不错的选择。

3. 本书中的代码

书中的所有源码和运行方式都可以在本书的 Github 仓库[9]中找到，书中几乎所有的例子都以测试的形式完成。

4. 反馈

如果你是开源贡献者，那么应该很习惯通过 Github Issue[9]提交任何反馈，如果是 Pull Request，那就更好了。当然如果没有使用过 Github Issue 也没有关系，这里[10]有非常详细的教程。

5. 代码风格约定

本书的 JavaScript 代码都遵循 Airbnb JavaScript Style Guide[11] 中的 ES5 和 React 的风格约定。

6. 本书的组织结构

第 1 章

将介绍 JavaScript 的基本函数式背景，简要地介绍为什么要关心函数式编程，为什么说 Underscore 不够函数式，JavaScript 要作为完整的函数式语言还缺些什么？

第 2 章

主要介绍 Clojure 的集合数据结构。这是个无聊但是又很重要的章节，可以说函数式编程最基本、最重要的就是集合操作。本章会涉及如何操作集合、惰性求值与惰性序列。

---

8 https://github.com/clojure-emacs/cider。

9 https://github.com/jcouyang/clojure-flavored-javascript/issues。

10 https://guides.github.com/features/issues/。

11 https://github.com/airbnb/javascript。

第 3 章

在了解了持久性数据结构后，我们可能会产生疑惑，如果数据结构都是不可变的，那么循环该怎么写呢？本章就是要解开各种使用不可变数据结构的疑惑，用这些不可变数据结构可以切换一种编程的思维方式。

第 4 章

Underscore 并不利于函数组合，但是函数组合其实是函数式编程最重要的思想。在这一章里面，我会详细介绍为什么说 Underscore 错了，而为什么要喜欢上柯里化，以及 Clojure 1.7 新推出的 Transducer 又是如何帮助我们更容易组合出更高效的函数的。

第 5 章

我特别不情愿把 Macro 翻译成宏。宏特别容易让人以为是 C 语言里面那个 #define 宏，虽然都是宏，但其实那里跟这里说的 Macro 不是一个级别的。Macro 是 Lisp 语言之所以特别的原因之一。本章我们就来看看到底什么是、为什么，以及如何在 JavaScript 中使用 Macro。

第 6 章

这里说的模式匹配包括两种：一种是按位置或者 key 匹配集合，取出相应数据。另一种是 Haskell 风格的对函数参数的模式匹配。本章篇幅比较小，因为模式匹配并不是 Clojure（也不是 JavaScript）的主要模式，尽管在一些有强大类型系统的函数式语言（Scala、Haskell）中比较重要。

第 7 章

Monad 这个范畴论里出来的神秘玩意，但你可能没有在意，其实这在前端世界早都被玩腻了。本章将会介绍 Monad 和它的朋友们，并且将带你体验 JavaScript 的 Promise，以及 Reactive 编程。

第 8 章

并发编程一直是令人头疼的编程方式，直到 Clojure 和 Go 的出现，彻底改变了我们并发编程的方式。而对于单线程的 JavaScript，基于事件循环的并发模型也一直困扰着我们，到底能从 Clojure 学些什么，可以使我们的前端并发编程之路更顺畅一些呢？本章将带你熟悉并发、JavaScript 的并发模型，以及 CSP 并发模型。

7．本书使用的约定

本书使用以下字体排版约定。

1）楷体

表示新的术语。

2）等宽字体

代码清单，出现在段落之内则表示变量、函数名、关键字等。

3）粗体

重点概念。

4）下画线

需要填入的词，我可能已经帮大家填上了。

5）横线

可以忽略的词。

反侵权盗版声明

目录