PEARSON

新·锐·编·程·语·言·集·萃·

Programming in CoffeeScript

CoffeeScript 程序设计

【美】Mark Bates 著
Goddy Zhao 译

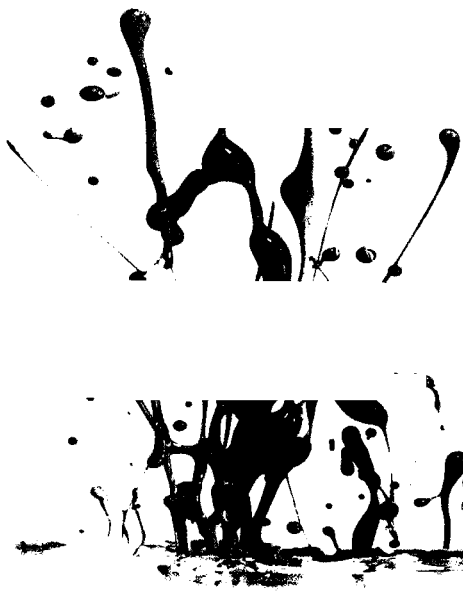
人民邮电出版社
POSTS & TELECOM PRESS

·新·锐·编·程·语·言·集·萃·

Programming in CoffeeScript

CoffeeScript 程序设计

【美】Mark Bates 著
Goddy Zhao 译



人民邮电出版社
北京

图书在版编目（CIP）数据

CoffeeScript程序设计 / (美) 贝茨 (Mark Bates) 著 ; Goddy Zhao译. -- 北京 : 人民邮电出版社, 2013.1
(新锐编程语言集萃)
书名原文: Programming in CoffeeScript
ISBN 978-7-115-30193-2

Ⅰ. ①C… Ⅱ. ①贝… ②G… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2012)第284974号

内 容 提 要

CoffeeScript 是一门新的编程语言，一门会被编译为 JavaScript 的语言。本书从运行和编译 CoffeeScript 的基础知识开始，逐步介绍其语法、控件结构、函数、集合和类等内容。本书的特色是，通过对相同页面的 CoffeeScript 代码和 JavaScript 代码的直接比较，让读者能够直观地了解 CoffeeScript 是如何改善了 JavaScript 的，进而能够用它构建强大、灵活、可维护、简洁、可靠以及安全的应用程序。此外，作者还在书中给出一些非常宝贵的提示，提醒读者如何才能更好地开发应用程序。

本书是一本理论和实践相结合的 CoffeeScript 入门教材，更是一本能够带领初学者充分理解并快速掌握 CoffeeScript 的好书，非常适合中高级 Web 开发者阅读。

新锐编程语言集萃

CoffeeScript 程序设计

◆ 著 [美] Mark Bates
译 Goddy Zhao
责任编辑 杨海玲

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 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：800×1000 1/16
印张：16
字数：359 千字 2013 年 1 月第 1 版
印数：1-3 000 册 2013 年 1 月河北第 1 次印刷

著作权合同登记号 图字：01-2012-6498 号

ISBN 978-7-115-30193-2

定价：45.00 元

读者服务热线：(010)67132692 印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第 0021 号

版权声明

译者序

随着 Web 2.0 的流行，尤其是如今 Node.js 的诞生，让服务器端 JavaScript 成为了可能，JavaScript 迎来了它生命的第二春。截止至现在，GitHub 中 21%[①]的开源项目都是用 JavaScript 编写的，占据榜首。RedMonk[②]在 2012 年 9 月 12 日发表的一篇关于统计最流行程序设计语言排行的分析报告中显示[③]，JavaScript 是目前最为流行的编程语言。微软公司的 Windows 8 操作系统中，加入了支持直接用 JavaScript 编写本地应用的特性。种种数据和现象都说明，JavaScript 又一次受到了全世界的关注。

然而，没有任何一门编程语言是完美的，JavaScript 也是如此，JavaScript 一直以来都因为其语言本身饱受争议：容易造成污染的全局变量、没有原生的命名空间、一大堆的假值、万恶的 `with` 和 `evil`，等等。为此，Douglas Crockford[④]专门写了一本名为 *JavaScript: The Good Parts*（《JavaScript 语言精粹》）的书，详细介绍了 JavaScript 中的“糟粕”与“精华”。如今，随着 JavaScript 的大规模使用，人们从心底里期待一个更好的 JavaScript。为此，官方的 ECMAScript[⑤]工作小组也制定了第 5 版的标准，Chrome、Firefox 等浏览器也开始陆续实现并支持第 5 版的新特性。除此之外，民间的声音也愈渐响亮，其中最有代表性的就当属 CoffeeScript 了。

和 Ruby 的到来让人眼前一亮一样，CoffeeScript 也给编写 JavaScript 带来了快感。简洁的语法让人把更多的精力放在逻辑本身，默认生成的匿名包装器函数让人无须再为全局变量的污染提心吊胆，操作符与别名让人无须再为一大堆的假值烦恼，内置的基于类的继承机制让人无须再为了搞清楚原型那些东西而绞尽脑汁。和 Ruby 一样，CoffeeScript 让人们更加快乐地编程。

对于 JavaScript 程序员来说，CoffeeScript 的学习成本非常低，因为它不像 Ruby 那样是一门全新的语言，它更多的是对 JavaScript 的扩展和增强；对于非 JavaScript 程序员而言，CoffeeScript 也很容易掌握，因为它的语法非常简洁，CoffeeScript 官方网站[⑥]用短短一页篇幅就展示了所有的语法。更何况，还有本书这样一本理论与实践并存、宽度与广度并具的 CoffeeScript 教材，可以帮助程序员更好地学习 CoffeeScript。

我始终坚信学习技术最好的方式就是实践，在实践的过程中遇到问题，再解决问题；在解决问题的过程中再去深入探寻其背后的理论依据，从而体系化地掌握学习的内容。本书就是这样一本理论和实践相结合的 CoffeeScript 入门教材。最为重要的是，它更加侧重于实践，提供

① https://github.com/languages
② http://redmonk.com/
③ http://redmonk.com/sogrady/2012/09/12/language-rankings-9-12/
④ http://www.crockford.com/
⑤ http://ecmascript.org
⑥ http://coffeescript.org/

了一个非常完整的包括前后端实现的待办事宜列表应用，带领读者学习和掌握 CoffeeScript。这也是我很喜欢本书的一个重要原因。除此之外，本书也不缺乏详细的理论，花了 4 章左右的篇幅详细介绍了 CoffeeScript 所有的语法和精髓。总的来说，这是一本能够带领初学者充分理解并快速掌握 CoffeeScript 的好书。

另外，我有责任提醒各位读者：CoffeeScript 也有它的不足，其中最为致命的就是没有很好的调试方式。原因就在于，编写用的是 CoffeeScript，而真正运行的是其编译后的 JavaScript 代码，因此，一旦出了错误，很难通过 JavaScript 错误来定位到底 CoffeeScript 源代码中哪里出错了。不过，好在 Chrome 的开发者工具以及 Firefox 的 Firebug 都在尝试添加 CoffeeScript 的调试功能，相信不久的将来，这个问题会得到解决。

最后，祝愿所有的读者都能够通过本书学习和掌握 CoffeeScript，与此同时，也能体会到编程的乐趣。

前 言

1999 年，我开始了专业的开发生涯，那年我第一次以开发者的身份领取薪水（那年是我开始对 Web 开发产生浓厚兴趣的时候，我并未将此前的几年开发时间计算在内。）。1999 年，Web 还是个“险恶之地”，HTML 中最常见到的还是 `font` 和 `table` 标签，CSS 初出茅庐，JavaScript [①] 也才刚刚崭露头角，并且主流浏览器之间还弥漫着对 JavaScript 各自不同实现的硝烟。这就意味着，编写的某段 JavaScript 代码可以在某个浏览器上正常工作，但却不一定能在其他浏览器上正确运行。正因如此，JavaScript 在 21 世纪初名声并不好。

在 21 世纪中期发生了两件重要的事情，改变了开发者对 JavaScript 的看法。第一件就是 AJAX[②]。它能让开发人员制作出交互性更好、速度更快的 Web 页面，使得用户无须手动刷新浏览器就能在后台对服务器进行远程调用。

第二件重要的事情就是 JavaScript 库的流行，如 Prototype[③]，它可以简化跨浏览器的 JavaScript 开发。AJAX 提高了 Web 应用的用户体验，像 Prototype 这样的 JavaScript 库则保证了 Web 应用可以跨主流浏览器工作。

到了 2010 年，当然也包括 2011 年，Web 应用逐步演变为“单页面”应用（single page application）。这类应用使用像 Backbone.js[④]这样的 JavaScript 框架。这类框架用 JavaScript 实现了前端的 MVC[⑤] 设计模式供开发者使用。整个应用使用 JavaScript 进行构建，在终端用户的浏览器端进行载入和运行。这些都服务于构建高响应、富客户端应用。

然而，对开发者而言，这并非就尽善尽美。尽管框架和工具简化了这类应用的开发，但是众所周知，JavaScript 语言本身有很多诟病，它是一门让人喜忧参半的语言。喜的是它非常强大，忧的是它充满了矛盾和设计陷阱，很快会让你的代码变得难以管理，而且会有隐蔽的潜在 bug。

开发者该如何是好？他们想要开发此类新应用，但是唯一被接受的浏览器语言就是 JavaScript。当然了，他们可以用 Flash[⑥]来写这类应用，以避免使用 JavaScript，但是这样一来就需要插件支持，并且在像 iOS[⑦]这样的不支持 Flash 的平台下就无法工作。

2010 年 10 月，我“邂逅”了 CoffeeScript[⑧]，当时它扬言要“驯服”JavaScript，将 JavaScript 这门怪异的语言最优秀的部分呈现出来。它拥有更为清晰的语法，比方说：CoffeeScript 摒弃了

① http://en.wikipedia.org/wiki/JavaScript
② http://en.wikipedia.org/wiki/Ajax_(programming)
③ http://www.prototypejs.org/
④ http://documentcloud.github.com/backbone/
⑤ http://en.wikipedia.org/wiki/Model–view–controller
⑥ http://www.adobe.com/
⑦ http://www.apple.com/ios/
⑧ http://www.coffeescript.org

大部分的标点符号[①]，采用了有意义的空格，从而保护 JavaScript 开发者免遭 JavaScript 语言设计缺陷产生的陷阱，例如糟糕的作用域问题以及比较操作符的误用。其中最值得称赞的是，以上这些工作，CoffeeScript 都会在将源代码编译为标准的 JavaScript 的过程中完成，编译出来的 JavaScript 可以在任意浏览器以及其他的 JavaScript 运行时环境中执行。

在我刚使用 CoffeeScript 时，尽管当时版本号已经是 0.9.4 了，但这门语言本身仍然“很粗糙”，还有不少问题。当时，我将其用在一个项目的客户端部分，主要是想看看它是否真的有我听说的那么好。但遗憾的是，那个时候有两个原因促使我将其搁置。第一个原因是，它还不够“成熟”，还有太多的 bug 并且功能尚未完善。

第二个原因是，我尝试使用 CoffeeScript 的应用并非是一个 JavaScript 很重的应用[②]。此应用中只用到了少量的校验和 AJAX，大部分的工作都让 Ruby on Rails[③]帮我处理掉了。

那么又是什么促使我“重拾”CoffeeScript 的呢？在我首次尝试 CoffeeScript 之后——6 个月之后的某一天，官方宣布[④]Rails 3.1 将以 CoffeeScript 作为其默认的 JavaScript 引擎。和绝大多数开发者一样，我被这个消息吸引了。要知道我之前就用过 CoffeeScript 并且觉得它不怎么好。他们是怎么想的呢？

和其他开发同事不同的是，我又花时间看了一下 CoffeeScript。6 个月对于任何一个项目来说都是很长的一段开发周期，CoffeeScript 也走过了相当长的一段路，因此我决定再次尝试 CoffeeScript，这次将它运用在 JavaScript 代码占相当大比重的应用中。经过几天的使用，我不仅改变了对 CoffeeScript 的看法，甚至成为了它的拥护者。

我不打算告诉你究竟是什么改变了我，也不想告诉你我为什么喜欢它。我想让你做出自己的选择。我希望通过阅读本书不仅能够改变你，而且也能让你成为这门漂亮、小巧的语言的拥护者，而这一切，都是出于你本人的意愿。不过，我会带你初窥一下后面的内容。下面是一段摘自真实应用的 CoffeeScript 脚本，其后是等效的 JavaScript 脚本。尽情享受吧！

例 （源代码：sneak_peak.coffee）

```
@updateAvatars = ->
  names = $('.avatar[data-name]').map -> $(this).data('name')
  Utils.findAvatar(name) for name in $.unique(names)
```

例 （源代码：sneak_peak.js）

```
(function() {

  this.updateAvatars = function() {
  var name, names, _i, _len, _ref, _results;
  names = $('.avatar[data-name]').map(function() {
    return $(this).data('name');
```

① 比方说，最常见的分号、大括号等。——译者注
② 即富客户端应用。——译者注
③ http://www.rubyonrails.org
④ http://www.rubyinside.com/rails-3-1-adopts-coffeescript-jquery-sass-and- controversy-4669.html

```
    });
    _ref = $.unique(names);
    _results = [];
    for (_i = 0, _len = _ref.length; _i < _len; _i++) {
      name = _ref[_i];
      _results.push(Utils.findAvatar(name));
    }
    return _results;
  };

}).call(this);
```

什么是 CoffeeScript

CoffeeScript 是一门会被编译为 JavaScript 的语言。我知道这个解释的内容并不丰富，但是事实确实如此。CoffeeScript 和 Ruby[①]以及 Python[②]很像，它设计的初衷就是帮助开发者更加高效地编写 JavaScript 代码。通过移除没有必要的诸如括号、分号这样的标点符号，用有意义的空格取而代之，可以让开发者更多地将注意力集中在代码本身，而不必去关心大括号是否关闭这样的问题。

比方说，你可能会写下面这样的 JavaScript 代码。

例 （源代码：punctuation.js）

```
(function() {

  if (something === something_else) {
    console.log('do something');
  } else {
    console.log('do something else');
  }

}).call(this);
```

那么为什么不试着把它写成下面的形式。

例 （源代码：punctuation.coffee）

```
if something is something_else
  console.log 'do something'
else
  console.log 'do something else'
```

CoffeeScript 还提供了几种快捷方式来简化复杂代码块的编写。例如，下面这段代码就可以实现数组值循环，不需要关心其下标。

① http://en.wikipedia.org/wiki/Ruby_(programming_language)
② http://en.wikipedia.org/wiki/Python_(programming_language)

例 （源代码：array.coffee）

```
for name in array
  console.log name
```

例 （源代码：array.js）

```
(function() {
  var name, _i, _len;

  for (_i = 0, _len = array.length; _i < _len; _i++) {
    name = array[_i];
    console.log(name);
  }

}).call(this);
```

除了这些改进的语法糖之外，CoffeeScript 还可以帮助写出更好的 JavaScript 代码。例如，帮助准确地确定变量和类的作用域，帮助确保正确使用了比较操作符，等等。这些在阅读本书时都能看到。

CoffeeScript、Ruby 以及 Python 经常会因它们相似的优点共同受到关注。CoffeeScript 直接借鉴了 Ruby、Python 这样的语言提供的简洁语法。正因如此，CoffeeScript 比 JavaScript 这门更像 Java[①]或者 C++[②]的语言更具“现代感”。与 JavaScript 相同的是，CoffeeScript 可以在任何编程环境中使用。不论是用 Ruby、Python、PHP[③]、Java 还是用.NET[④]开发的应用，都没有关系。编译后的 JavaScript 代码都可以很好地和它们一起工作。

因为 CoffeeScript 会编译为 JavaScript，所以你仍然可以使用现在使用的所有 JavaScript 库，可以用 jQuery[⑤]、Zepto[⑥]、Backbone[⑦]、Jasmine[⑧]等，并且它们都可以很好地工作。而这种情况对于一门新语言来讲并不多见。

听起来很不错吧，不过我仿佛听到了你在问：与 JavaScript 相比，CoffeeScript 的缺点有哪些呢？这是个很好的问题。答案是：缺点肯定有，但还不算多。首先，尽管 CoffeeScript 是一种很好的编写 JavaScript 代码的方式，但是，凡是 JavaScript 无法实现的，使用 CoffeeScript 也依然无能为力。比方说，我不可能用 CoffeeScript 创建一个类似 Ruby 中最著名的 method_missing[⑨]的 JavaScript 版本。另外，CoffeeScript 的最大缺点就是，对于你和你的团队成员，它完全是另外一门要学习的语言。不过好在，它足够简单，你会看到，CoffeeScript 非常容易学。

最后，如果出于某种原因，CoffeeScript 不适合你或者你的项目，你也可以使用生成的 JavaScript。总之，说真的，你没有理由不在下一个项目甚至是当前项目中尝试 CoffeeScript

① http://en.wikipedia.org/wiki/Java_(programming_language)
② http://en.wikipedia.org/wiki/C%2B%2B
③ http://en.wikipedia.org/wiki/Php
④ http://en.wikipedia.org/wiki/.NET_Framework
⑤ http://www.jquery.com
⑥ https://github.com/madrobby/zepto
⑦ http://documentcloud.github.com/backbone
⑧ http://pivotal.github.com/jasmine/
⑨ http://ruby-doc.org/docs/ProgrammingRuby/html/ref_c_object.html#Object.method_ missing

（CoffeeScript 和 JavaScript 互相配合得很好）。

本书适合什么样的读者

本书适合中高级 JavaScript 开发者。之所以我觉得本书不大适合对 JavaScript 不熟悉或者仅对其略知一二的人是有原因的。

首先，本书不会教 JavaScript，这是一本介绍 CoffeeScript 的书。虽然在读的过程中你肯定也能学到一些零碎的 JavaScript 知识（CoffeeScript 会让你学到更多关于 JavaScript 的知识），但是本书不会从头开始教你 JavaScript。

例 这段代码做了什么？（源代码：example.js）

```
(function() {
  var array, index, _i, _len;

  array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

  for (_i = 0, _len = array.length; _i < _len; _i++) {
    index = array[_i];
    console.log(index);
  }

}).call(this);
```

如果你不知道上面这段示例代码是什么意思，我建议你就此打住。不用担心，我是真心想要你再回来继续阅读下去。我只是觉得，如果你能对 JavaScript 有深入的理解，就能最大限度地利用这本书。在本书的讲述过程中，我通常会介绍一些 JavaScript 的基础知识，帮助你更好地理解 CoffeeScript。不过，尽管如此，在继续阅读之前，对 JavaScript 有一定的基础还是非常重要的。因此，请去找本介绍 JavaScript 的优秀书籍（有很多这样的书）来读，然后再和我一起踏上成为 CoffeeScript 高手的旅途。

对于已经是 JavaScript 高手的你，让我们一起踏上旅途吧！本书将教你如何使用 CoffeeScript 编写更简洁、更好的 JavaScript 代码。

如何阅读本书

这里我提供一些阅读本书的方法，希望能帮助你循序渐进地学习 CoffeeScript。第一部分中的各章应该按顺序阅读，因为每一章都是建立在前一章的基础之上的，所以请不要跳着阅读。

在阅读每一章的过程中，你会注意到以下一些事情。

首先，每当我介绍一些外部库、想法和概念的时候，都会加上脚注，写上相应的网站地址，通过它你可以学到更多相关的内容。尽管我很想和你喋喋不休地讲些其他东西，像 Ruby，但是

本书篇幅有限。因此，如果对我提到的某些东西感兴趣，想要在继续阅读之前了解更多相关的信息，那就请你访问对应的网站，满足你对知识的渴望之后，再回来继续阅读本书。

其次，在每一章中，我有时会先介绍问题错误的解决方案。在看明白了错误的方式后，我们才能审视它、理解它，进而想出正确的解决方案。运用这种方式的一个典型例子在第 1 章中，我们讨论了从 CoffeeScript 编译为 JavaScript 的多种不同方式。

本书中还会出现如下内容：

> **提示**：这里是一些有用的提示。这些是我认为或许会对你有帮助的一些提示。

最后，本书中通常都会一次出现两三段代码块。首先是 CoffeeScript 脚本，然后是对应的编译后的 JavaScript 脚本，如果示例有输出的话，最后，可能还会有示例的输出结果（如果我认为有必要展示的话），如下所示。

例 （源代码：example.coffee）

```
array = [1..10]

for index in array
  console.log index
```

例 （源代码：example.js）

```
(function() {
  var array, index, _i, _len;

  array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

  for (_i = 0, _len = array.length; _i < _len; _i++) {
    index = array[_i];
    console.log(index);
  }

}).call(this);
```

输出 （源代码：example.coffee）

```
1
2
3
4
5
6
7
8
9
10
```

有时，还会特意展示一些错误，如下所示。

例 （源代码：oops.coffee）

```
array = [1..10]

oops! index in array
  console.log index
```

输出 （源代码：oops.coffee）

```
Error: In content/preface/oops.coffee, Parse error on line 3: Unexpected 'UNARY'
    at Object.parseError (/usr/local/lib/node_modules/coffee-script/lib/coffee-script/
➥parser.js:470:11)
    at Object.parse (/usr/local/lib/node_modules/coffee-script/lib/coffee-script/
➥parser.js:546:22)
    at /usr/local/lib/node_modules/coffee-script/lib/coffee-script/coffee-script.
➥js:40:22
    at Object.run (/usr/local/lib/node_modules/coffee-script/lib/coffee-script/
➥coffee-script.js:68:34)
    at /usr/local/lib/node_modules/coffee-script/lib/coffee-script/command.js:135:29
    at /usr/local/lib/node_modules/coffee-script/lib/coffee-script/command.js:110:18
    at [object Object].<anonymous> (fs.js:114:5)
    at [object Object].emit (events.js:64:17)
    at afterRead (fs.js:1081:12)
    at Object.wrapper [as oncomplete] (fs.js:252:17)
```

本书的组织结构

为了让读者从本书中尽可能地获益，我将本书分为两个部分。

第一部分：核心 CoffeeScript

本书第一部分自上向下地介绍了整个 CoffeeScript 语言。阅读完这部分内容之后，就完全可以应对各种 CoffeeScript 项目了，包括第二部分内容里提到的那些。

第 1 章“从这里开始”，介绍各种编译和运行 CoffeeScript 的方法。除此之外，这一章还会介绍 CoffeeScript 自带的功能强大的命令行工具和 REPL。

第 2 章“基础知识”，介绍 CoffeeScript 与 JavaScript 的不同点。内容涵盖语法、变量、作用域，以及更多后续章节所需的基础知识。

第 3 章“控制结构”，介绍语言中的一个重要部分——控制结构，比如 `if`、`else`。这一章还会介绍 CoffeeScript 中的操作符和 JavaScript 中的操作符的区别。

第 4 章“函数与参数”，详细介绍 CoffeeScript 中的函数。内容涵盖函数定义、函数调用以及一些额外的东西，比如默认参数和 splat 操作符。

第 5 章“集合与迭代”，从数组到对象，依次介绍在 CoffeeScript 如何使用、操作，以及迭代集合对象。

第 6 章“类”，第一部分的最后一章，介绍 CoffeeScript 中的类。内容涵盖类的定义、类的扩展、函数重载等。

第二部分：CoffeeScript 实践

本书第二部分介绍 CoffeeScript 实践。通过学习 CoffeeScript 周边的生态系统以及构建完整的应用，让你精通 CoffeeScript。

第 7 章“Cake 与 Cakefile”，介绍 CoffeeScript 自带的 Cake 工具。它用来创建构建脚本、测试脚本等。与之相关的概念都会介绍。

第 8 章“使用 Jasmine 测试”，测试是软件开发过程中非常重要的一环，这一章会带你快速一览最流行的 CoffeeScript/JavaScript 测试库之一——Jasmine。这一章还会通过一个计算器类的测试脚本来带着你体验一把流行的测试驱动开发模式。

第 9 章“Node.js 介绍”，简单介绍事件驱动的服务器端框架——Node.js。这一章会使用 CoffeeScript 来构建一个简单的 HTTP 服务器，该服务器可以根据网页浏览器的请求自动将 CoffeeScript 文件编译为 JavaScript 文件。

第 10 章“示例：待办事宜列表第 1 部分（服务器端）”，将介绍构建一个待办事宜列表应用的服务器端部分。在第 9 章的基础上，我们用 Express.js web 框架以及 MongoDB 的 ORM 框架 Mongoose 来构建一个 API。

第 11 章“示例：待办事宜列表第 2 部分（客户端，使用 jQuery）”，介绍使用流行的 jQuery 库来为第 10 章中建立的待办事宜列表 API 构建客户端部分。

第 12 章“示例：待办事宜列表第 3 部分（客户端，使用 Backbone.js）”，抛弃 jQuery，采用客户端框架 Backbone.js 对待办事宜列表应用进行重构。

安装 CoffeeScript

我不是很喜欢在书中介绍安装指南，主要是因为到了书上架开售的时候，这些安装指令也许已经过时了。不过，有的时候出版社编辑会觉得应当在书中介绍安装指南。本书便是如此。

安装 CoffeeScript 非常简单。最简单的方式就是访问 CoffeeScript 的官方网站，网址是 http://www.coffeescript.org/，然后根据上面的安装指南安装即可。

我相信像 CoffeeScript 和 Node[①]这样的项目的维护者一定会保证官方网站上的安装指南的时效性，所以，直接根据官网的安装指南安装是最好的方式。

① http://nodejs.org

本书撰写时，CoffeeScript 的版本号是 1.2.0。书中所有的例子在该版本的 CoffeeScript 下都可以工作。

如何运行书中示例

在 https://github.com/markbates/Programming-In-CoffeeScript 可以下载本书所有示例的源代码。该站点上的内容一目了然，所有的示例都可以在对应的示例文件中找到。示例文件可以根据出现的章号在对应的章文件夹中找到。

除非有特殊说明，否则，所有的示例代码应该都能在终端以如下形式运行：

```
> coffee example.coffee
```

现在你已经知道如何运行本书示例代码了，再安装好 CoffeeScript，我们就可以马上进入第 1 章的内容了。

致 谢

有句话在我第一本书中已经说过，在这里我还要重复一遍：写书真的是一项艰苦的工作！我相信不会有人对此有不同的感受。如果真的有，要么他在撒谎，要么他就是斯蒂芬·金[①]。幸运的是，我介于两者之间。

写书既是一项独立的工作，也需要团队的努力。我哄孩子睡觉后，就一头扎进了书房，一连打开几瓶吉尼斯黑啤，调大音乐声，开始独自写书，一连好几个小时，一直到凌晨。每当完成一章，就发给我的编辑，他会把我的稿子发给其他一些人从不同方面对其进行改进，这些人我可能都不认识。他们简单的会纠正语法或者拼写错误，复杂的会帮助改善书的结构或者指出书中示例代码不清楚的地方。因此，说真的，写作可能是一个人独自在小黑屋中完成的，但是，最终的产品一定是一群人共同努力的结果。

在这里，我有机会对所有为你现在手中（或者下载的）这本高质量的书做出过努力的人表示感谢。下面，请允许我效仿奥斯卡颁奖礼致感谢词，对在感谢名单上遗漏的人，我深感抱歉。

首先，我最要感谢的是我美丽的太太 Rachel。她是我见过的最支持我也最坚强的人之一。每一个夜晚在她身边入睡，每一个早晨从她身旁醒来。凝视着她的双眸，我能看到无私的爱，无比幸福。而且，不论我写书、创业还是做任何我觉得开心的事情，她都在背后支持我。她给了我两个帅气的儿子，反过来，我给她的则是我蹩脚的幽默和我用过的手机。我俩的婚姻中，我无疑是受益者，为此，我会永远心存感激。

接下来，我要感谢我的两个儿子 Dylan 和 Leo。尽管他们没有对本书作出直接贡献，但是他们给了我对生活的动力和激情，这一切除了孩子无人能给。儿子，爸爸非常爱你们。

我还要感谢我的父母（特别是你，母亲！）以及其他的家庭成员，谢谢你们一直以来对我的支持，同时，还时刻让我保持自省。我爱你们。

下面，我还要感谢 Debra Williams Cauley。Debra 是本书的编辑、负责人，同时也是我第一本书——《Ruby 分布式编程》（*Distributed Programming with Ruby*）的“心理医生”。我只希望其他的作者也能同样幸运，能有机会与像 Debra 这样好的编辑合作。她真的非常有耐心。

我希望下次再写书的话还能与 Debra 合作。我无法想象写书要是没有她会怎样。真的谢谢你，Debra。

在写技术书的过程中，有些人至关重要，那就是技术审校者。技术审校者的工作就是阅读每个章节，并从技术角度对内容进行评论，就好像在回答这样的问题：“这里介绍这些合适吗？”

---

① 斯蒂芬·金是一位作品多产，屡获奖项的美国畅销书作家，编写过剧本、专栏评论，曾担任电影导演、制片人以及演员。——译者注

他们扮演书的读者，同时又都懂技术。因此，他们的反馈非常重要。本书有几位审校者，其中我特别想提的两位是 Stuart Garner 和 Dan Pickett。他们对审校工作非常负责，甚至做了许多超越其职责的事情，而且对我做过的蠢事或者说过的蠢话直言不讳。他们总要被我不分昼夜的邮件和电话打扰，但却总能给我很好的反馈。要不是我想“独吞”版税，我真想把版税分给他们。（别担心，他们也会得到自己的报酬。他们每个人都得到了工作时间一小时的休息。）谢谢 Dan、Stuart 以及其他审校者们，谢谢你们的辛苦付出。

我要感谢许多以不同方式对本书做出贡献的朋友。有人为本书设计了封面，有人建了索引，有人实现了程序设计语言（CoffeeScript）以及其他与此相关的数之不尽的工作。下面是这些人（我知道的）的名单，排名不分先后： Jeremey Ashkenas、Trevor Burnham、Dan Fishman、Chris Zahn、Gregg Pollack、Gary Adair、Sandra Schroeder、Obie Fernandez、Kristy Hart、Andy Beaster、Barbara Hacha、Tim Wright、Debbie Williams、Brian France、Vanessa Evans、Dan Scherf、Gary Adair、Nonie Ratcliff 以及 Kim Boedigheimer。

我还要感谢所有在我开始写书时听我唠叨的人。我知道，这对绝大多数人而言有点儿枯燥，但是，我就是喜欢听自己的声音。感谢所有容忍我啰唆的朋友。

最后，我要感谢所有读者。感谢你们购买本书，并支持像我这样的人，纯粹只想将自己的知识分享给别人来帮助开发者们。为了你们，我投入了大量的工作时间以及精力来书写本书。我希望在你们阅读完本书时，能够对 CoffeeScript 有更深的理解，并希望它能够对你们的开发工作产生影响。祝你好运！