TURING 图灵程序设计丛书 Web开发系列

Apress®

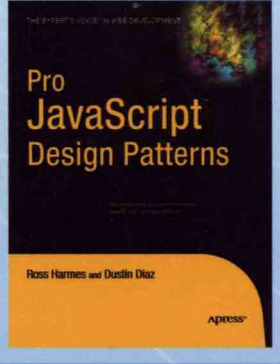
Pro JavaScript Design Patterns

JavaScript 设计模式

[美] Ross Harmes
Dustin Diaz 著

谢廷晟 译

- 从这里开始，真正掌握JavaScript的精髓
- Google和Yahoo专家联手揭秘世界顶尖公司的技术内幕
- Amazon全五星盛誉图书



人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 Web开发系列

Pro JavaScript Design Patterns

JavaScript 设计模式

[美] Ross Harmes
Dustin Diaz 著

人民邮电出版社
北 京

图书在版编目（CIP）数据

JavaScript 设计模式 /（美）哈梅斯（Harmes, R.），（美）迪亚斯（Diaz, D.）著；谢廷晟译 . —北京：人民邮电出版社，2009.1
（图灵程序设计丛书）
书名原文：Pro JavaScript Design Patterns
ISBN 978-7-115-19128-1

Ⅰ. J… Ⅱ. ①哈…②迪…③谢 Ⅲ. JAVA 语言－程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第171032号

内 容 提 要

本书共有两部分。第一部分给出了实现具体设计模式所需要的面向对象特性的基础知识，主要包括接口、封装和信息隐藏、继承、单体模式等内容。第二部分则专注于各种具体的设计模式及其在 JavaScript 语言中的应用，主要介绍了工厂模式、桥接模式、组合模式、门面模式等几种常见的模式。为了让每一章中的示例都尽可能地贴近实际应用，书中同时列举了一些 JavaScript 程序员最常见的任务，然后运用设计模式使其解决方案变得更模块化、更高效并且更易维护，其中较为理论化的例子则用于阐明某些要点。

本书适合各层次的 Web 前端开发人员阅读和参考，也适合有 C++/Java/C# 背景的服务器端程序员学习。

图灵程序设计丛书

JavaScript设计模式

◆ 著 [美]Ross Harmes Dustin Diaz
译 谢廷晟
责任编辑 傅志红
执行编辑 杨 爽

◆ 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 电子函件 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：16.75
字数：396千字 2009年 1 月第 1 版
印数：1－4000册 2009年 1 月河北第 1 次印刷

著作权合同登记号 图字：01-2008-5001号

ISBN 978-7-115-19128-1/TP

定价：45.00 元

读者服务热线：(010)88593802 印装质量热线：(010)67129223

反盗版热线：(010)67171154

版 权 声 明

对本书的赞誉

本书道前人所未道，引导你从编写代码转变为设计代码。书中绝大部分示例代码都来自YUI等实战项目，并进行了深入剖析。强烈推荐。

——Nicholas C. Zakas，著名JavaScript专家，Yahoo前端工程师，畅销书《JavaScript高级程序设计》作者

本书绝对值得细细品读，提供了大量有益而且有趣的JavaScript实现代码，其中对JavaScript面向对象特性的阐述是我读过的书中最出色的。

——Jeff Wilcox，微软Silverlight核心程序员

本书是JavaScript成熟的标志。两位作者详细剖析了面向对象实现的底层机制，并演示了在实战中如何灵活运用设计模式，获得优美的设计方案。

——Tiff Fehr，MSNBC.com用户体验工程师

想成为JavaScript专家？本书千万不能错过！

——Amazon.com读者评论

本书介绍的技术将使JavaScript如虎添翼。如果你要用JavaScript开发较大规模的程序，本书必不可少。

——JavaRanch.com

毫不夸张地说，这是我有生以来读到的最好的一本JavaScript图书。作者讲授了大量专家级的经验。

——Mostafa Farghaly，埃及程序员

本书是Web程序员深刻理解各种Ajax/JavaScript框架的宝典。

——theopensourcery.com

对于有一定经验的JavaScript程序员，本书绝对物超所值。想知道Google和Yahoo内部怎样开发企业级的程序吗？仔细研读吧。

——James Stewart，资深Web工程师

本书是市面上最好的一本关于JavaScript的书，适合想更多地了解JavaScript和设计模式的人。

——Amazon.com读者评论

还在怀疑JavaScript的强大？本书将使你大开眼界。

——Dzone.com

译 者 序

设计模式对于程序员来说并不是一个陌生话题。在Erich Gamma等人合著的经典著作《设计模式》出版之后，十几年间陆续出现了许多这方面的专著。不过它们大都结合Java和C++等传统的面向对象语言进行讲解，而讲述设计模式在动态语言中的实现的书则较为罕见。在早期的JavaScript编程实践中，这种语言只被用于做点为网页涂脂抹粉的小差事；程序的规模很小，也很简单。那个时候恐怕没有人会想到把设计模式用到这种“玩具语言”编写的程序中。随着Ajax技术的兴起，Web应用的许多逻辑都从服务器端转移到客户端执行，客户端JavaScript程序的作用越来越重要，其规模和复杂程度也越来越大，人们也越来越多地把面向对象方法应用到JavaScript程序设计中。在此背景下，有许多人开始研究设计模式在JavaScript程序设计中的应用，网上也陆续出现了一些关于这个话题的零星讨论。但是到目前为止，系统地探讨面向对象的程序设计模式在JavaScript语言中的实现的书，只此一本。（Michael Mahemoff所著的《Ajax设计模式》一书总结的是运用Ajax技术开发Web应用的各种设计模式，虽然也涉及大量JavaScript编程，但它与本书关注的焦点不同。本书讨论的是一些通用的面向对象设计模式在JavaScript中的实现，属于更基础性的东西，它们不仅仅适用于Web客户端编程。）

JavaScript这种语言与Java等传统的面向对象语言有很大的不同。它的动态性、词法作用域和基于原型的继承机制等特点可能会让很多初次接触它的程序员都有点不习惯，而且由于语言设计上的一些不完善，许多在传统面向对象语言中只是举手之劳的事在JavaScipt中却不得不依靠hack手法来实现。这也许就是那些已经熟知设计模式在Java等语言中实现方式的程序员也需要本书的原因。本书第一部分着重讲述了面向对象技术在JavaScript中的实现方法。这对于对JavaScript只有过初步了解的人非常有用（当然，本书不适合对JavaScript一窍不通的读者。他们应该先找一本JavaScript基础教材来看看，比如人民邮电出版社出版的《JavaScript基础教程》）。Java和C++编程老手们在学完这部分内容之后，想必应该能够在JavaScript程序设计中自行应用各种经典的设计模式了。不过不同的人可能会有一些不同的做法，因此继续看看本书第二部分，借鉴一下作者的方法也不无益处。对于那些从未学过设计模式的JavaScript程序员来说，本书的重要性更是毋庸置疑。不过，坦率地说，要想深入学习设计模式仅看本书是不够的。取代Gamma等人的《设计模式》并不是本书的目标。

就个人的感觉而言，我觉得本书最大的遗憾之处在于作者在讲述各种设计模式的实现时没有使用原型式继承，而是选择了类式继承。毕竟原型式继承才是JavaScript面向对象编程最自然的继承实现方式。不过正如作者所言，“有些人似乎天生就容易被原型式继承的简洁性吸引，而另一

些人却对更面熟的类式继承情有独钟”，这只是个人口味的问题。考虑到很多人对原型式继承都非常陌生，作者的这种选择可以理解。

在本书的翻译过程中，我先后两次厚着脸皮向本书的执行编辑杨爽女士请求推迟交稿时间，她总是和颜悦色地告诉我：多花点时间不要紧，翻译质量才是最重要的。在此我要感谢杨爽女士的宽容，并感谢她和本书的文字编辑罗凌云女士认真负责的编辑工作。此外，李松峰、贺师俊、郭晓刚、刘江、麦天志、肖鹏、周琦、崔驰坤、米全喜、任斌、霍泰稳、张一宁和徐毅等朋友在本书的翻译过程中曾提供了许多宝贵意见，在此我一并表示感谢。由于我的水平有限，翻译不当之处在所难免。谨请各位读者朋友不吝赐教。建议您在图灵公司的网站（www.turingbook.com）或互动出版网（www.china-pub.com）上本书的网页上提出自己的意见。

谢廷晟

2008年秋

前　言

目前，JavaScript到了一个转折关头。这门语言和它的用户都已经成熟起来。人们也开始认识到：它是一个复杂的课题，值得进一步研究。

设计模式运用在程序设计中已经有些年头了。它们最早被整理记录于Erich Gamma、Richard Helm、Ralph Johnson和John Vlissides〔绰号“四人帮”（the Gang of Four，后文中简写为Gof）〕合著的*Design Patterns*[①]一书中，现已被应用到各种各样的面向对象语言中。设计模式的魅力之一体现在它们被应用于各种语言和语法上时所表现出的一致性上。其基本结构是相同的，只是细节略有差别。例如，把一个用Java实现的模式转换为C++形式就很容易。

但是对JavaScript来说情况则有所不同。尽管所有那些能力JavaScript都有，但它们往往并非这种语言的正式部分，因而必须借助于一些晦涩的技巧和不那么直观的技术来模仿。这些年来，人们找到了许多方法用该语言来完成其设计者都未曾预计到的任务。那些常见的面向对象特性同样也要靠这样的手段实现。

本书收集整理了这些技巧和技术。第一部分提供了一个实现具体设计模式所需要的面向对象特性的基础。第二部分则专注于各种具体的设计模式及其在JavaScript语言中的应用。

为了让每一章中的示例都尽可能地贴近实际应用，我们花了不少心思。我们尽量列举一些JavaScript程序员最常见的任务，然后运用设计模式使其解决方案变得更模块化、更高效并且更易维护。而那些较为理论化的例子则用于阐明某些要点。我们知道，本书的价值最终将取决于它与你的日常工作和项目的紧密联系程度。

希望你能喜欢这本书。JavaScript是一种极其复杂而又灵活的语言，而且很适合于动手实践。你可以随时试试我们提供的示例代码。要是你找到实现某种设计模式的新方法，或者某种旧技术的新用途，请告诉我们一声。本书的附属网站http://jsdesignpatterns.com和Apress出版社的网站http://www.apress.com提供了更多信息以及可供下载的示例代码。

目标读者

本书主要面向两类读者。第一类读者是懂一点JavaScript并且想要加深对它的认识的Web开发人员或前端工程师，尤其是那些想要增进其对于JavaScript的面向对象能力的理解，学习如何提高其代码的模块化程度、可维护性和效率的人，他们可以从书中学到用JavaScript进行面向对象程序

① 中文版名为《设计模式——可复用面向对象软件的基础》（机械工业出版社，2004）。——译者注

设计的基本知识，还能学到各种具体的设计模式，懂得应该在什么场合使用这些设计模式，以及如何实现它们。这类读者已经比较熟悉JavaScript的基本语法，他们会更多地关注那些关于如何按特定设计模式重构现有代码的部分，以及对于每种模式的适用场合的说明。

第二类读者是一些主要使用Java和C++等服务器端编程语言的程序员，相对而言，他们对JavaScript比较陌生，但又希望能把自己在设计模式和面向对象程序设计方面的知识应用到客户端程序设计中。他们可以从本书中学到如何在JavaScript中实现接口、继承和封装等常见的面向对象特性。这类读者会发现书中的代码尤其有用，因为他们可能不太熟悉JavaScript和其他面向对象语言在语法方面的差别。也许他们对各种具体的设计模式已经比较熟悉，所以本书讲述面向对象技术在JavaScript中的实现方法的第一部分对他们可能更有意义。

那些对JavaScript和面向对象程序设计都不太熟悉的读者可能很难看懂某些示例中的代码。这并不是一本入门级的书，它假定读者已经具备一定的程序设计知识。尽管如此，我们还是尽可能地使自己的讲解做到深入浅出，让不同层次的读者都易于理解。

本书结构

本书分为两部分。第一部分讲述JavaScript面向对象程序设计的基础知识，其中的各章应该按顺序阅读。每一章都建立在前面一章的基础上，并且假定你已经读过之前的各章。读者最好先通读这一部分的所有章节，因为第二部分的各章都要用到第一部分讲述的技术，而且有时不会再加以说明。

第二部分讲述具体的设计模式及其在JavaScript中的实际应用，其中的各章可以按任意顺序阅读。有些章引用了其他章的内容，被引用的内容既有第一部分的，也有第二部分的，我们都给出了引文的章号。

第一部分

第1章（富有表现力的JavaScript）揭示了JavaScript语言富有表现力的特点。从中你可以体会到，这种语言允许你用各种各样的编程风格来完成同样的任务，还允许你在面向对象编程的过程中借用函数式编程的概念来丰富其实现方式。这一章解释了究竟为什么应该使用设计模式，以及它们在JavaScript程序设计中的运用是如何使代码更高效、更易于处理的。

第2章（接口）分析了其他面向对象语言实现接口的方式，并用JavaScript对它们在这方面的最佳特性进行了模仿。文中探讨了接口检查的各种可行方式，并给出了一个可用来检查对象是否具有必要方法的可重用的类。

第3章（封装和信息隐藏）探讨了在JavaScript中创建对象的各种不同方式，以及每种方式中用以创建公用（public）、私用（private）和受护（protected）方法[①]的可行技术。文中还对经过复杂封装的对象的适用场合进行了讨论。

① JavaScript并不像许多面向对象语言那样提供public、private和protected关键字及相关的特性。第3章中用一些技术模仿了其他语言对public成员和private成员的实现，但没有模仿protected成员的实现。——译者注

第4章（继承）讲述了在JavaScript中用以创建子类的各种技术。其中既讲了类式继承，也讲了原型式继承，还说明了它们各自的适用场合。文中还讲述了掺元类（mixin class）[①]，以及如何用它替代多亲继承（multiple inheritance）[②]。

第5章（单体模式）讨论了JavaScript中的单体模式、命名空间、代码组织，以及可以用来根据运行时环境动态定义方法的分支技术（branching）。其中还谈到了工厂模式等可以受益于单体的模式。

第6章（方法的链式调用）考察了JavaScript对方法进行链式调用的能力，以及这种做法为什么能得到更清晰、简练的代码。文中用这种技术创建了一个小小的JavaScript库，并把其中的方法和没有利用链式调用技术实现的对应方法进行了比较。

第二部分

第7章（工厂模式）讨论工厂模式。这种模式有助于消除那些彼此实例化对方的类之间的耦合，并改而用一个方法来确定要实例化哪个类。文中既讨论了另外用一个类（通常是一个单体）来生成实例的简单工厂模式，也讨论了用子类来确定一个成员对象应该是哪种具体类实例的较复杂的工厂模式。

第8章（桥接模式）讨论了一种既能把两个对象连接在一起，又能避免二者间的强耦合的方法。桥接元素把两个对象连接起来，同时又允许它们独立变化。文中演示了如何用桥接元素把函数松散地绑定到事件。这一章还创建了一个异步连接队列，用以示范桥接模式在保持代码的简洁性方面的作用。

第9章（组合模式）讨论了一种非常适合于创建Web上的动态用户界面的设计模式：组合模式。文中演示了如何使用这种模式来达到只用一条命令即可在许多对象上激发复杂或递归性的行为的目的，以及如何用它把一系列对象组织为复杂的层次体系（hierarchy）。文中还逐一说明了实现组合模式所必经的一系列步骤，并讨论了该模式的适用场合。

第10章（门面模式）讨论了一种用来为对象创建一个更完善的接口的方法。门面模式可以用来把现有接口转换为一个更便于使用的接口。文中解释了为什么大多数JavaScript库都是为这种语言在具体浏览器中的实现提供的一个门面。这一章也显示了如何用这种模式创建便利方法和事件工具库。

第11章（适配器模式）讨论了一种可以让现有接口楔合实际需要的模式。适配器也称包装器（wrapper），用来把不匹配的接口替换为一个可用于现有系统中的接口。文中探讨了如何用适配器弥合JavaScript库间的差异并简化从一种库过渡到另一种库的过程。其中考察了一个电子邮件API，并创建了一个有助于升级到新版本的适配器。

第12章（装饰者模式）讨论了一种可以为对象添加特性而又不必创建新的子类的方法。装饰者模式用于把对象透明地包装到另一种具有相同接口的对象中。这一章考察了装饰者的结构以及

① 也译为“混入类”。——译者注

② 也译“多重继承”，但感觉不够准确。

如何将其与工厂模式结合使用以自动创建内嵌对象。我们还创建了一个性能分析器，以示范如何用装饰者模式动态实现接口。

第13章（享元模式）讨论了另一种用于优化目的的模式：享元模式。文中示范了如何通过把大批独立对象转变为少量共享对象，从而大幅削减实现应用软件所需的对象数目。这一章还创建了一个Web日历和一个可重用的工具提示类，以示范如何按享元模式对类进行重构。

第14章（代理模式）讨论了可用于控制对对象的访问的代理模式。文中显示了如何为本体（real subject）[①]创建一个代理对象以作为其替身，并使其能被远程访问，还探讨了代理模式的种种用法，其中包括推迟对其创建需要耗用大量计算资源的类的实例化。这一章还创建了一个可用来推迟任何类的加载的通用类。

第15章（观察者模式）讨论了一种对对象的状态进行观察，并且当它发生变化时能得到通知的方法。观察者模式也称发布者-订阅者模式（publisher-subscriber pattern），用于让对象对事件进行监听以便对其作出响应。文中以报业为例说明了观察者模式的各种运作方式，还讨论了在使用一个动画库时可以订阅的各种事件。

第16章（命令模式）讨论了一种对方法调用进行封装的方式。借助命令模式，可以对方法调用进行参数化和传递，然后在需要的时候再加以执行。文中说明了这种模式的各种应用场合，其中包括创建用户界面——尤其是那种需要不受限制的取消操作的用户界面。这一章还讨论了命令模式的结构，并提供了几个示范其在JavaScript中的用法的实际例子。

第17章（职责链模式）讨论了用来消除请求的发送者和接收者之间的耦合的职责链模式。文中说明了在JavaScript中如何用这种模式处理事件的捕获和冒泡，如何创建弱耦合模块和优化事件监听器的绑定。

预备知识

为了使书中的代码示例更加清晰、目标更加明确，我们使用了一些便利函数来执行安装事件监听器、派生子类、处理Cookie、引用HTML元素等任务。我们没有使用YUI或jQuery等库所提供的类似功能，其目的在于避免让我们的代码依赖于其他库，以便读者可以将其与自己喜欢的任何库结合使用。各种主流JavaScript库都有与我们使用的函数相对应的函数。这些便利函数的完整代码可以从本书的网站http://jsdesignpatterns.com和Apress出版社的网站http://www.apress.com下载。下面是这些函数的简要说明。

- `$(id)`，根据ID值获取HTML元素的引用。其参数可以是字符串，也可以是字符串的数组。
- `addEvent(obj, type, func)`，把函数func作为元素obj的事件监听器。type表示该函数所要监听的事件。
- `addLoadEvent(func)`，将函数func关联到window对象的load事件。
- `getElementsByClass(searchClass, node, tag)`，获取所有class属性值为searchClass的元素的引用。node和tag这两个参数可有可无，它们可以用来缩小搜索范围。函数的返回值

① 也译为“实体”或“主体”。——译者注

是一个数组。

- insertAfter(parent, node, referenceNode)，插入元素node，其父元素为parent，其位置在referenceNode之后。
- getCookie(name)，获取与名为name的Cookie相关联的字符串。
- setCookie(name, value, expires, path, domain, secure)，把与名为name的Cookie相关联的字符串设置为value。除name和value外的其他参数均可有可无。
- deleteCookie(name)，将名为name的Cookie的过期时间设置到过去。[①]
- clone(object)，创建object的一个副本。用于原型式继承。见第4章。
- extend(subClass, superClass)，执行一些必要的工作，使subClass成为superClass的子类。见第4章。
- augment(receivingClass, givingClass)，将givingClass中的方法输入receivingClass中。见第4章。

下载代码

本书的附属网站http://jsdesignpatterns.com和Apress出版社的网站http://www.apress.com都有zip文件格式的示例代码供下载。[②]

联系作者

你可以通过dustin@jsdesignpatterns.com和ross@jsdesignpatterns.com与作者联系。

致谢

感谢认真负责的技术审稿人Simon Willison。要不是他，本书的准确性、实用性和趣味性都得大打折扣。他不厌其烦地为每一章提供了令人惊喜的反馈。

感谢那些在百忙之中抽空阅读本书草稿并提供意见和更正的同事和合作伙伴。Dave Marr和Ernest Delgado更是在排除打字错误、技术错误和病句方面发挥了重要作用。同时，也要感谢Lindsey Simon和Robert Otani，他们层出不穷的JavaScript幽默为我们提供了不少帮助。

感谢我们的朋友和家人。面对我们没完没了的有关写作和晦涩的技术细节的唠叨，他们总是很有耐心。他们的支持是我们前进的动力。

最后，真心感谢Apress出版社负责本书的工作人员Chris Mills、Tom Welsh、Dominic Shakeshaft、Richard Dal Porto，还有Jennifer Whipple，他们的耐心、毅力和善解人意尤其值得赞赏并且令人难忘。

① 即删除这个Cookie。——译者注

② 示例代码也可以在图灵网站（www.turingbook.com）本书网页免费注册下载。——编者注

第1章 富有表现力的JavaScript

JavaScript是现在最流行、应用最广泛的语言之一。由于所有现代浏览器都嵌入了JavaScript解释器，所以在大多数地方都能见到其身影。作为一种语言，它在我们的日常生活中起着非常重要的作用，支持着我们访问的网站，帮助Web呈现出多姿多彩的界面。

那为什么有些人还把它看作一种玩具式的语言，认为它不值得职业程序员关注呢？我们认为其原因在于，人们没有认清这种语言的全部能力及其在当今的编程世界中的独特性。JavaScript是一种极富表现力的语言，它具有一些C家族语言所罕见的特性。

本章将探讨一些令JavaScript如此富有表现力的特性。从中你可以体会到，这种语言允许你用各种方式完成同样的任务，还允许你在面向对象编程的过程中借用函数式编程中的概念来丰富其实现方式。本章解释了究竟为什么应该使用设计模式，以及它们在JavaScript程序设计的运用是如何使代码更高效、更易于处理的。

1.1 JavaScript的灵活性

JavaScript最强大的特性是其灵活性。作为JavaScript程序员，只要你愿意，可以把程序写得很简单，也可以写得很复杂。这种语言也支持多种不同的编程风格。你既可以采用函数式编程风格，也可以采用更复杂一点的面向对象编程风格。即使你根本不懂函数式编程或面向对象编程，也能写出较为复杂的程序。使用这种语言，哪怕只采用编写一个个简单的函数的方式，你也能高效地完成任务。这可能是某些人把JavaScript视同玩具的原因之一，但我们却认为这是一个优点。程序员只要使用这种语言的一个很小的、易于学习的子集就能完成一些有用的任务。这也意味着当你成长为一个更高级的程序员时，JavaScript在你手中的威力也在增长。

JavaScript允许你模仿其他语言的编程模式和惯用法。它也形成了自己的一些编程模式和惯用法。那些较为传统的服务器端编程语言具有的面向对象特性，JavaScript都有。

先来看一个用不同方法完成同样的任务的例子：启动和停止一个动画。如果你看不懂这些例子，别担心。我们在此使用的所有模式和技术都会在本书后面进行讲解。目前你可以把这一部分看作一个演示在JavaScript中用不同方法完成同一任务的实际例子。

如果你习惯于过程式的程序设计，那么可以这样做：

```
/* Start and stop animations using functions. */

function startAnimation() {
  ...
}

function stopAnimation() {
  ...
}
```

这种做法很简单，但你无法创建可以保存状态并且具有一些仅对其内部状态进行操作的方法的动画对象。下面的代码定义了一个类，你可以用它创建这种对象：

```
/* Anim class. */

var Anim = function() {
  ...
};
Anim.prototype.start = function() {
  ...
};
Anim.prototype.stop = function() {
  ...
};

/* Usage. */

var myAnim = new Anim();
myAnim.start();
...
myAnim.stop();
```

上述代码定义了一个名为Anim的类，并把两个方法赋给该类的prototype属性。第3章将详细讲述这种技术。如果你更喜欢把类的定义封装在一条声明中，则可改用下面的代码：

```
/* Anim class, with a slightly different syntax for declaring methods. */

var Anim = function() {
  ...
};
Anim.prototype = {
  start: function() {
    ...
  },

  stop: function() {
    ...
  }
};
```

这在传统的面向对象程序员看来可能更眼熟一点，他们习惯于看到类的方法声明内嵌在类的