

*Learning JavaScript Design Patterns*



# JavaScript 设计模式

O'REILLY®

[美] *Addy Osmani* 著  
徐涛 译

 人民邮电出版社  
POSTS & TELECOM PRESS

013043596

TP312JA  
1494

# JavaScript 设计模式

[美] Addy Osmani 著

徐 涛 译



北航

C1651785

人民邮电出版社

北京

TP312JA  
1494

106830810

## 图书在版编目 (C I P) 数据

JavaScript设计模式 / (美) 奥斯马尼 (Osmani, A.)  
著 ; 徐涛译. -- 北京 : 人民邮电出版社, 2013. 6  
ISBN 978-7-115-31454-3

I. ①J… II. ①奥… ②徐… III. ①JAVA语言—程序  
设计 IV. ①TP312

中国版本图书馆CIP数据核字(2013)第063408号

## 版权声明

Copyright© 2012 by O'Reilly Media, Inc.

Simplified Chinese Edition, jointly published by O'Reilly Media, Inc. and Posts & Telecom Press, 2013. Authorized translation of the English edition, 2012 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

本书中文简体版由 **O'Reilly Media, Inc.** 授权人民邮电出版社出版。未经出版者书面许可, 对本书的任何部分不得以任何方式复制或抄袭。

版权所有, 侵权必究

- 
- ◆ 著 [美] Addy Osmani
  - 译 徐 涛
  - 责任编辑 陈冀康
  - 责任印制 程彦红 焦志炜
  
  - ◆ 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: 787×1000 1/16
  - 印张: 16
  - 字数: 301千字 2013年6月第1版
  - 印数: 1-3000册 2013年6月河北第1次印刷
- 
- 著作权合同登记号 图字: 01-2013-1022 号

定价: 49.00 元

读者服务热线: (010)67132692 印装质量热线: (010)67129223  
反盗版热线: (010)67171154

---

# 内容提要

设计模式是解决软件设计中常见问题的可复用方案。学习任何编程语言，设计模式都是一个令人兴奋和极具吸引力的话题。

本书是 JavaScript 设计模式的学习指南。全书分为 14 章。首先介绍了什么是模式、模式的结构、类别、模式的分类、如何编写模式等等；然后，集中介绍了很多流行的设计模式在 JavaScript 中的应用，包括 Module（模块）模式、Observer（观察者）模式、Facade（外观）模式和 Mediator（中介者）模式；最后，还探讨了模块化的 JavaScript 模式、jQuery 及其插件中的设计模式。

本书适合专业的 Web 开发人员和前端工程师阅读。通过阅读本书，他们将能够提高对设计模式的认识，并学会如何将设计模式应用到 JavaScript 编程语言中。

---

# 前言

设计模式是解决软件设计中常见问题的可复用方案。探索任何编程语言时，设计模式都是一个令人兴奋和极具吸引力的话题。

原因之一是：设计模式是许多先前开发人员总结出的经验，我们可以借鉴这些经验进行编程，以确保能够以优化的方式组织代码，为我们解决棘手的问题提供参考。

设计模式还是我们用来描述解决方案的常用词汇。当我们想要向其他人表述一种以代码形式构建解决方案的方式时，描述设计模式比描述语法和语义要简单得多。

在本书中，我们将探讨 JavaScript 编程语言中经典的与现代的设计模式的应用。

## 目标读者

本书的目标读者是专业开发人员，希望提高对设计模式的认识，并学会如何将设计模式应用到 JavaScript 编程语言中。

本书对有些概念（闭包、原型继承）只做了一些基本介绍，以便于理解。如果想要进一步了解这些概念，下面列出了一些推荐书目，方便大家阅读。

如果想要学习如何编写出美观、结构化和组织良好的代码，相信这本书就是为你而准备的。

## 致谢

本书中提到的很多设计模式是根据我的个人经验总结出来的，还有很多设计模式是 JavaScript 社区前辈总结出来的经验。本作品是众多开发人员的经验结合产物。与斯托扬·斯蒂凡诺夫为防止致谢名单中出现错漏的明智做法（在《JavaScript 模式》中）类似，我列出了致谢名单以及本书参考的文献。

如果参考文献中遗漏了任何著作或链接，我深表歉意。如果您与我联系，我一定会及时将您的著作或链接添加到参考文献中。

## 其他读物

虽然本书也面向初学者和中级开发人员，但也需要读者对 JavaScript 有基本的了解。如果想要深入了解该语言，我很乐意向您推荐以下书目。

- 《JavaScript 权威指南》，作者：David Flanagan
- 《JavaScript 编程精解》，作者：Marius Haverbeke
- 《JavaScript 模式》，作者：Stoyan Stefanov
- 《编写可维护的 JavaScript》<sup>1</sup>，作者：Nicholas Zakas
- 《JavaScript 语言精粹》，作者：Douglas Crockford

## 本书约定

本书使用下列排版约定。

斜体 (*Italic*)

表示专业词汇、链接 (URL)、文件名和文件扩展名。

等宽字体 (Constant width)

表示广义上的计算机编码，它们包括变量或函数名、数据库、数据类型、环境变量、语句和关键字。

等宽粗体 (**Constant width bold**)

表示应该由用户按照字面引入的命令或其他文本。

---

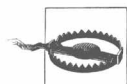
<sup>1</sup> 编者注：《编写可维护的 JavaScript》已由人民邮电出版社出版 (ISBN9787115310088，定价 55 元，2013 年 3 月)。

等宽斜体 (*Constant width italic*)

表示应该由用户替换或取决于上下文的值。



这个图标表示提示、建议或一般说明。



这个图标表示警告或提醒。

## 代码示例

这本书是为了帮助你做好工作。一般来说，你可以在程序和文档中使用本书的代码。你无须联系我们获取许可。例如，使用来自本书的几段代码写一个程序是不需要许可的。出售和散布 O’Reilly 书中用例的光盘 (CD-ROM) 是需要许可的。通过引用本书用例和代码来回答问题是不需要许可的。把本书中大量的用例代码并入到你的产品文档中是需要许可的。

我们赞赏但不强求注明信息来源。一条信息来源通常包括标题、作者、出版者和国际标准书号 (ISBN)。例如：“Learning JavaScript Design Patterns by Addy Osmani (O’Reilly). Copyright 2012 Addy Osmani, 978-1-449-33181-8”。

如果你感到对示例代码的使用超出了正当引用或这里给出的许可范围，请随时通过 [permissions@oreilly.com](mailto:permissions@oreilly.com) 联系我们。

## Safari®在线图书

Safari 在线图书 (Safari Books Online) 是一家按需服务的数字图书馆，提供来自领先出版商的技术类和商业类专业参考书目和视频。

专业技术人员、软件开发人员、Web 设计师、商业和创意专家将 Safari Books Online 作为他们研究、解决问题、学习和认证培训的主要资源。

Safari Books Online 为组织、政府机构和个人提供一系列的产品组合和定价计划。

用户可以在一个来自各个出版社的可完全搜索的数据库中访问成千上万的书籍、培训视频和正式出版前的手稿，这些出版社包括：O'Reilly Media、Prentice Hall Professional、Addison-Wesley Professional、微软出版社、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 等等。欲获得有关 Safari Books Online 的更多信息，请在线访问我们。

## 联系我们

关于本书的建议和疑问，可以与下面的出版社联系：

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

我们将关于本书的勘误表，例子以及其他信息列在本书的网页上，网页地址是：

<http://www.oreilly.com/catalog/9781449302146>

如果要评论本书或者咨询关于本书的技术问题，请发邮件到：

[bookquestions@oreilly.com](mailto:bookquestions@oreilly.com)

想了解关于 O'Reilly 图书、课程、会议和新闻的更多信息，请访问以下网站：

<http://www.oreilly.com.cn>

<http://www.oreilly.com>

## 致谢

我要感谢热情的技术审稿人帮助我检查和改进本书，这些审稿人还包括整个社区里的同行。他们的知识和热情对这一工作做出了不可思议的贡献。这些技术审稿人的官方 tweet 和博客也经常为我提供一些想法和灵感，我竭诚向大家推荐他们的 tweet 和博客。



- Nicholas Zakas (<http://nczonline.net>, [@slicknet](http://twitter.com/slicknet)(<http://twitter.com/slicknet>))
- Andréa Hansson (<http://andreehansson.se>, [@peolanha](http://twitter.com/peolanha)(<http://twitter.com/peolanha>))
- Luke Smith (<http://lucassmith.name>, [@lis\\_n](http://twitter.com/lis_n)([http://twitter.com/lis\\_n](http://twitter.com/lis_n)))
- Eric Ferraiuolo (<http://ericf.me/>, [@ericf](http://ericf.me/) (<http://ericf.me/>))
- Peter Michaux (<http://michaux.ca>, [@petermichaux](http://twitter.com/petermichaux))
- Alex Sexton (<http://alexsexton.com>, [@slexaxton](http://twitter.com/slexaxton)(<http://twitter.com/slexaxton>))

我还要感谢丽贝卡·墨菲 (<http://rebeccamurphey.com>, [@rmurphey](http://twitter.com/rmurphey) (<http://twitter.com/rmurphey>)) 给我写作的灵感, 更重要的是, 本书在 GitHub 和 O'Reilly 出版公司都可以获取。

最后, 我要感谢我的妻子埃莉, 她非常支持我的出书工作。

# 目录

第 1 章	介绍	1
第 2 章	什么是模式	3
	我们每天都在使用模式	4
第 3 章	模式状态测试、Proto 模式及三法则	6
第 4 章	设计模式的结构	8
第 5 章	编写设计模式	11
第 6 章	反模式	13
第 7 章	设计模式类别	15
第 8 章	设计模式分类	17
	有关类 (Class) 的要点	17
第 9 章	JavaScript 设计模式	20
9.1	Constructor (构造器) 模式	21
9.1.1	对象创建	21
9.1.2	基本 Constructor (构造器)	23
9.1.3	带原型的 Constructor (构造器)	24
9.2	Module (模块) 模式	25
9.2.1	对象字面量	25
9.2.2	Module (模块) 模式	27
9.2.3	Module 模式变化	31
9.3	Revealing Module (揭示模块) 模式	36
9.3.1	优点	38
9.3.2	缺点	38
9.4	Singleton (单例) 模式	38
9.5	Observer (观察者) 模式	42
9.5.1	Observer (观察者) 模式和 Publish/Subscribe (发布/订阅) 模式的区别	47
9.5.2	优点	49
9.5.3	缺点	49
9.5.4	Publish/Subscribe 实现	49
9.6	Mediator (中介者) 模式	59

9.6.1	基本实现 .....	60
9.6.2	高级实现 .....	61
9.6.3	示例 .....	67
9.6.4	优点和缺点 .....	68
9.6.5	中介者 (Mediator) 与观察者 (Observer) .....	69
9.6.6	中介者 (Mediator) 与外观 (Facade) .....	69
9.7	Prototype (原型) 模式 .....	70
9.8	Command (命令) 模式 .....	73
9.9	Facade (外观) 模式 .....	75
	有关抽象的要点 .....	78
9.10	Factory (工厂) 模式 .....	78
	9.10.1 何时使用 Factory 模式 .....	81
	9.10.2 何时不应使用 Factory 模式 .....	81
	9.10.3 Abstract Factory (抽象工厂) .....	81
9.11	Mixin 模式 .....	82
	9.11.1 子类化 .....	83
	9.11.2 Mixin (混入) .....	84
9.12	Decorator (装饰者) 模式 .....	88
	9.12.1 伪经典 Decorator (装饰者) .....	91
	9.12.2 使用 jQuery 的装饰者 .....	96
	9.12.3 优点和缺点 .....	97
9.13	Flyweight (享元) 模式 .....	98
	9.13.1 使用 Flyweight 模式 .....	98
	9.13.2 Flyweight 和共享数据 .....	99
	9.13.3 实现经典 Flyweight (享元) .....	99
	9.13.4 转换代码以使用 Flyweight (享元) 模式 .....	103
	9.13.5 基本工厂 .....	105
	9.13.6 管理外部状态 .....	106
	9.13.7 Flyweight (享元) 模式和 DOM .....	107
<b>第 10 章</b>	<b>JavaScript MV* 模式 .....</b>	<b>112</b>
10.1	MVC .....	112
	Smalltalk-80 MVC .....	113
10.2	为 JavaScript 开发人员提供的 MVC .....	114
	10.2.1 Model (模型) .....	114
	10.2.2 View (视图) .....	116

10.2.3	Controller (控制器)	119
10.2.4	Spine.js 与 Backbone.js	120
10.3	MVC 为我们提供了什么	122
10.4	JavaScript 中的 Smalltalk-80 MVC	122
10.4.1	深入挖掘	123
10.4.2	总结	123
10.5	MVP	124
10.5.1	Model、View 和 Presenter	124
10.5.2	MVP 或 MVC?	125
10.5.3	MVC、MVP 和 Backbone.js	126
10.6	MVVM	128
10.6.1	历史	129
10.6.2	Model	129
10.6.3	View	130
10.6.4	ViewModel	133
10.6.5	小结: View 和 ViewModel	135
10.6.6	小结: ViewModel 和 Model	135
10.7	利与弊	135
10.7.1	优点	135
10.7.2	缺点	136
10.8	使用更松散数据绑定的 MVVM	136
10.9	MVC、MVP 与 MVVM	141
10.10	Backbone.js 与 KnockoutJS	142
<b>第 11 章</b>	<b>模块化的 JavaScript 设计模式</b>	<b>144</b>
11.1	脚本加载器要点	145
11.2	AMD	145
11.2.1	模块入门	146
11.2.2	使用 Dojo 的 AMD 模块	150
11.2.3	AMD 模块设计模式 (Dojo)	151
11.2.4	使用 jQuery 的 AMD 模块	152
11.2.5	AMD 总结	155
11.3	CommonJS	155
11.3.1	入门指南	156
11.3.2	使用多个依赖	157
11.3.3	支持 CommonJS 的加载器和框架	158

11.3.4	CommonJS 适用于浏览器吗?	158
11.3.5	延伸阅读	159
11.4	AMD 和 CommonJS: 互相竞争, 标准同效	159
	UMD: 用于插件的 AMD 和 CommonJS 兼容模块	160
11.5	ES Harmony	165
11.5.1	具有 Imports 和 Exports 的模块	166
11.5.2	从远程数据源加载的模块	167
11.5.3	模块加载器 API	167
11.5.4	用于服务器的类 CommonJS 模块	168
11.5.5	具有构造函数、getter 和 setter 的类	168
11.5.6	ES Harmony 总结	169
11.5.7	延伸阅读	170
11.6	总结	170
<b>第 12 章</b>	<b>jQuery 中的设计模式</b>	<b>171</b>
12.1	Composite (组合) 模式	171
12.2	Adapter (适配器) 模式	173
12.3	Facade (外观) 模式	174
12.4	Observer (观察者) 模式	177
12.5	Iterator (迭代器) 模式	180
12.6	延迟初始化	181
12.7	Proxy (代理) 模式	183
12.8	Builder (生成器) 模式	184
<b>第 13 章</b>	<b>jQuery 插件设计模式</b>	<b>187</b>
13.1	模式	188
13.2	Lightweight Start 模式	189
	延伸阅读	191
13.3	完整的 Widget Factory 模式	191
	延伸阅读	194
13.4	嵌套命名空间插件模式	194
	延伸阅读	196
13.5	自定义事件插件模式 (使用 Widget Factory)	196
	延伸阅读	198
13.6	使用 DOM-to-Object Bridge 模式的原型继承	198
	延伸阅读	200
13.7	jQuery UI Widget Factory Bridge 模式	200

延伸阅读 .....	203
13.8 使用 Widget Factory 的 jQuery Mobile Widget .....	203
13.9 RequireJS 和 jQuery UI Widget Factory .....	206
13.9.1 用法 .....	208
13.9.2 延伸阅读 .....	209
13.10 全局选项和单次调用可重写选项（最佳选项模式） .....	209
延伸阅读 .....	211
13.11 高可配和高可变的插件模式 .....	211
延伸阅读 .....	213
13.12 是什么使插件超越模式 .....	213
13.12.1 质量 .....	214
13.12.2 代码风格 .....	214
13.12.3 兼容性 .....	214
13.12.4 可靠性 .....	214
13.12.5 性能 .....	214
13.12.6 文档 .....	215
13.12.7 维护的可能性 .....	215
13.13 总结 .....	215
13.14 命名空间模式 .....	215
13.15 命名空间基础 .....	216
13.15.1 单一全局变量 .....	216
13.15.2 命名空间前缀 .....	217
13.15.3 对象字面量表示法 .....	217
13.15.4 嵌套命名空间 .....	221
13.15.5 立即调用的函数表达式（IIFE） .....	222
13.15.6 命名空间注入 .....	224
13.16 高级命名空间模式 .....	226
13.16.1 自动嵌套的命名空间 .....	227
13.16.2 依赖声明模式 .....	229
13.16.3 深度对象扩展 .....	229
13.16.4 推荐 .....	232
<b>第 14 章 总结 .....</b>	<b>233</b>
<b>附录 参考文献 .....</b>	<b>235</b>

# 第 1 章

## 介绍

编写易于维护的代码，其中一个最重要方面是能够找到代码中重复出现的主题并优化它们。这也是设计模式有价值的地方。

在本书第 1 章，我们将探讨可应用于任何编程语言的设计模式发展史和重要性。如果大家已经看过或熟悉这段发展史，可以直接跳到第 2 章继续阅读。

设计模式来源于土木工程师克里斯托弗·亚历山大（[http://en.wikipedia.org/wiki/Christopher\\_Alexander](http://en.wikipedia.org/wiki/Christopher_Alexander)）的早期作品。他经常发表一些作品，内容是总结他在解决设计问题方面的经验，以及这些知识与城市和建筑模式之间有何关联。有一天，亚历山大突然发现，重复使用这些模式可以让某些设计构造取得我们期望的最佳效果。

亚历山大与萨拉-石川佳纯和穆雷·西尔弗斯坦合作创造了一种建筑模式语言，帮助设计师提高自己的设计能力，以解决任何规模的设计和建筑挑战。这就是亚历山大在 1977 年发表的一篇题为《建筑模式语言》的文章，其后又制作成一本完整的精装书出版（[http://www.amazon.co.uk/Pattern-Language-Buildings-Construction-Environmental/dp/0195019199/ref=sr\\_1\\_1?s=books&ie=UTF8&qid=1329440685&sr=1-1](http://www.amazon.co.uk/Pattern-Language-Buildings-Construction-Environmental/dp/0195019199/ref=sr_1_1?s=books&ie=UTF8&qid=1329440685&sr=1-1)）。

大约在 30 年前，软件工程师们开始把亚历山大编写的建筑设计原则纳入首个有关设计模式的文档中，成为初级开发人员改进编程技巧的指南。需要指出的是，设计模式背后的概念实际上自编程行业诞生以来就已经存在了，虽然是以一种不太正式的形式存在。

关于软件工程设计模式的最早和最具代表性的作品是 1995 年出版的《设计模式：可复用面向对象软件的基础》一书。该书的作者是 Erich Gamma([http://en.wikipedia.org/wiki/Erich\\_Gamma](http://en.wikipedia.org/wiki/Erich_Gamma))、Richard Helm ([http://www.amazon.co.uk/Pattern-Language-Buildings-Construction-Environmental/dp/0195019199/ref=sr\\_1\\_1?s=books&ie=UTF8&qid=1329440685&sr=1-1](http://www.amazon.co.uk/Pattern-Language-Buildings-Construction-Environmental/dp/0195019199/ref=sr_1_1?s=books&ie=UTF8&qid=1329440685&sr=1-1))、Ralph Johnson([http://en.wikipedia.org/wiki/Ralph\\_Johnson](http://en.wikipedia.org/wiki/Ralph_Johnson)) 和 John Vlissides ([http://en.wikipedia.org/wiki/John\\_Vlissides](http://en.wikipedia.org/wiki/John_Vlissides))，他们以“四人组”著称（或简称为 GoF）。

GoF 发表的著作大大推动了设计模式概念在编程领域中的进一步发展，因为它描述了很多开发技术和误区，并列举了 23 个面向对象设计中最常用的经典设计模式。我们将在第 7 章进一步详细介绍这些设计模式。

我们将在本书中了解一些流行的 JavaScript 设计模式，并探讨为何某些设计模式比其他模式更适用于我们编写的程序。要记住，设计模式不仅适用于原生 JavaScript（即标准 JavaScript 代码），也适用于 jQuery 和 Dojo 等抽象库。在开始之前，我们先来了解一下软件设计中“模式”的确切定义。



# 什么是模式

模式是一种可复用的解决方案，可用于解决软件设计中遇到的常见问题，如在我们编写的 JavaScript 应用程序的实例中。另一种模式的方式是将解决问题的方法制作成模板，并且这些模板可应用于多种不同的情况。

那么，为什么了解和熟悉模式是很重要的？设计模式有三大好处。

*模式是已经验证的解决方案。*

它们为解决软件开发中遇到的问题提供可靠的方法，也就是使用已经验证的解决方案，这些解决方案体现了开发人员的经验及见解，他们为定义和改进这些方法提供了帮助，从而形成现在的模式。

*模式很容易被复用。*

模式通常是指一种立即可用的解决方案，可以对其进行修改以满足个人需求。该特性使得这些模式的功能非常强大。

*模式富有表达力。*

看到模式时，通常就表示有一个设置好的结构和表达解决方案的词汇，以帮助我们非常轻松地表达出所实现的大型解决方案。