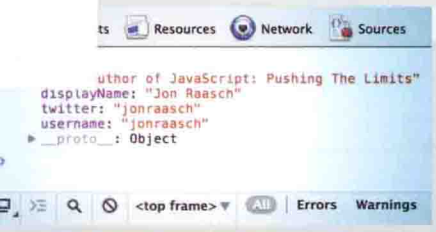
TURING 图灵程序设计丛书

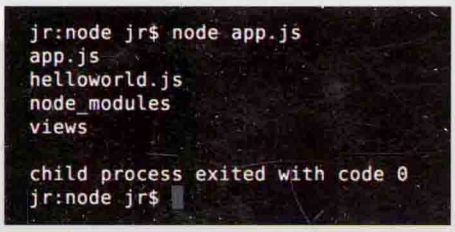
遵循最佳实践 全面深入挖掘JavaScript精华
助你构建21世纪杀手级应用

# JavaScript 编程实战

JavaScript Programming
Pushing the Limits

[美] Jon Raasch 著
吴海星 译





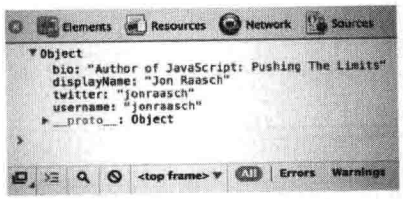
人民邮电出版社
POSTS & TELECOM PRESS

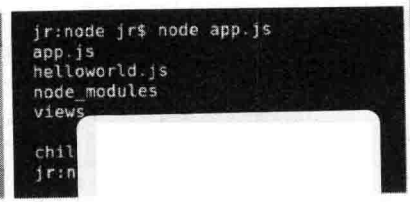
TURING 图灵程序设计丛书

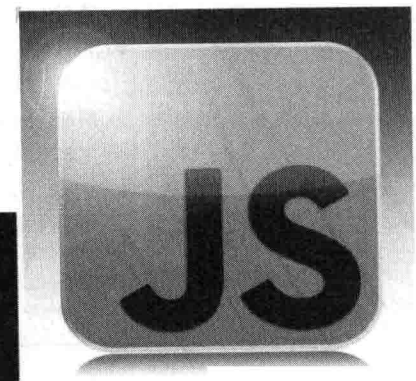
# JavaScript 编程实战

JavaScript Programming
Pushing the Limits

[美] Jon Raasch 著
吴海星 译




人民邮电出版社
北 京

**图书在版编目（CIP）数据**

JavaScript编程实战 / （美）拉希（Raasch, J.）著 ；吴海星译. -- 北京 ：人民邮电出版社，2014.3
（图灵程序设计丛书）
书名原文：JavaScript programming: pushing the limits
ISBN 978-7-115-34548-6

Ⅰ. ①J… Ⅱ. ①拉… ②吴… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2014)第017956号

## 内 容 提 要

本书深入探讨了如何基于JavaScript技术从头开始创建真实的应用，共分为四个部分。第一部分介绍了最佳实践以及库、框架与插件，为构建应用奠定坚实的基础。第二部分讨论了前端的构建，包括Backbone.js、JavaScript模板，以及表单处理和校验的相关内容。第三部分涉及如何用Node.js编写服务器端JavaScript。最后一部分挑战程序的功能极限，介绍了如何构建实时应用程序、调整移动领域的Web程序、JavaScript图形处理技术等内容。

本书适合所有熟悉JavaScript并希望提升相关技术水平的开发人员和设计人员学习参考。

◆ 著　[美] Jon Raasch
译　吴海星
责任编辑　丁晓昀
执行编辑　陈婷婷
责任印制　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：800×1000　1/16
印张：20
字数：499千字　2014年3月第1版
印数：1－4 000册　2014年3月河北第1次印刷
著作权合同登记号　图字：01-2013-8801号

定价：59.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第0021号

# 版 权 声 明

# 关于作者

Jon Rassch 是一位专门从事桌面端及移动端现代 Web 程序开发的自由职业者。他极度沉迷于用户体验，所构建的 JavaScript 程序专注于服务用户，旨在为用户改善性能、提高可用性或增强功能。他坚信只要满足了用户的需求，商业目标就能随之达成。除本书外，他还著有 *Smashing WebKit*，并与人合著了 *Smashing Mobile Web Development* 一书。Jon 在在线杂志 *Smashing Magazine* 和自己的博客上发表了无数文章，是最佳实践的完美主义者，经常穿着睡衣工作。他现居住在美国俄勒冈州的波特兰市，Twitter 账号 Jon @jonraasch，博客地址 jonraasch.com。

# 关于贡献者

Kevin Bradwick 有十多年的 Web 开发经验，热衷于构建并分享产出高质量代码的知识。他现在供职于 BBC，负责其国内和国际网站应用的开发和维护。

# 致　　谢

我要对回馈开发者社区的所有人致以深挚的谢意。感谢所有为开源项目做过贡献、写过教程、回答过问题，甚至提交过 bug 的朋友。我们只是站在了巨人的肩膀上，无数贡献者通过过去几年的努力将 Web 开发带到一个美好的境界。对于 Web 开发人员而言，这是一个迷人的时代，而这一切都要感谢我们优秀的社区。

——Jon Rassch

# 引　言

现在写 JavaScript 比以往任何时候都有意思。最近这些年，HTML5 引入了大量的 JavaScript API，不断在浏览器中开疆拓土，JavaScript 开发人员能做的事情越来越多。用这些 API 可以实现一些又炫又酷的特性，比如 3D 图形、地理位置定位数据以及高性能动画等。不过 HTML5 的高级之处并不仅仅在于这些让用户颇为震撼的华丽组件，它还具备大量能精简开发流程以及让你创建出下一代 Web 应用的 API。

如果 JavaScript 在浏览器中的发展不足以燃起你的热情，还有服务器端的 Node.js。这可不是学校里用 JavaScript 做服务器的作业，Node.js 是为现代 Web 应用模型精心准备的工业级服务器方案，它能支持用户间的实时消息传递。

这本书要教你学会如何创建下一代 JavaScript 程序。我不仅要让你知其然，还要让你知其所以然，所以除了讲述如何构建程序，我还会讲解其背后的原理。读完本书，你就能掌握构建富客户端交互式程序的技术。但更重要的是，本书能加深你对工程最佳实践的理解，不管你将来用不用 JavaScript，都会成为更有责任感的开发人员。

本书分为四部分。

- **第一部分：坚实的基础**。开篇介绍了一些常用的最佳实践，并论述了本书的核心思想：松耦合和关注点分离。然后探讨了一些开发工具，并教你如何用 Grunt.js 确立测试驱动开发（TDD）方式。最后比较了几个可以用来快速启动开发的类库和框架，并教你如何为每个项目挑选最合适的框架。
- **第二部分：构建前端**。这一部分会深入探讨如何以 Backbone.js 为基础搭建程序，也就是建立一个遵循 MVC 模式的前端，把数据从界面中分离出来。你还能学到如何用 `pushState` 搭建一个基于 Ajax 的界面导航系统。接下来我们会拓展你的 Backbone 知识，教你如何用 JavaScript 模板引擎渲染界面。通过模板可以进一步把程序中的数据从显示层分离出来。

  本部分最后介绍了一些表单处理和校验的最佳实践，给你的程序一个完整的基础。我们会使用渐进式增强的方法，从 HTML5 的基本状态开始，然后加上 JavaScript 部件，以及为了兼容老浏览器所需的 polyfill。最后在 Backbone 中创建一个可以自动跟程序后台同步的表单。
- **第三部分：编写服务器端 JavaScript**。本部分将教你如何用 Node.js 编写服务器端 JavaScript。你会了解一些关于 Node 的基础知识，包括它的工作原理、何时使用它，以及该如何使用 Node 平台的常用 Node 开发模式。

  接下来，我会向你介绍 Express 框架，它能精简 Node 开发，助你旗开得胜。我会教你如何设置路由，用模板渲染页面，处理表单提交的数据。最后是 MongoDB，一个能用在 Node.js 环境中的 NoSQL 数据库，可以代替 MySQL 这种传统的关系型数据库。

- **第四部分：挑战极限**。在本部分中我们要挑战程序的功能极限。首先介绍如何构建一个实时的应用程序，把你的服务器端 JavaScript 知识跟第二部分介绍的客户端技术结合到一起。你会看到 WebSocket，以及在 Node 中如何用 Socket.IO 实现它们。然后我会给出一个用 Backbone.js、Express 和 MongoDB 构建实时应用程序的例子。

  接着会介绍程序移动组件的构建。你将学到如何生成响应式内容，并用一个移动框架完成移动开发工作。我还会教你如何用 Hammer.js 编出触摸手势的处理程序，以及如何利用地理位置这样的移动专用 API。然后你会发现 PhoneGap 如何在浏览器级 API 的差距中架起桥梁，并基于 JavaScript 创建一个专用的移动程序。

  进入移动领域后，你会看到 HTML5 最令人激动的能力之一：在浏览器中画图。你将学到如何处理画布和 SVG，以及如何使用 Raphaël SVG 库。最后学习如何用 WebGL 和 Three.js 库渲染 3D 图形。

  第 12 章给出了一个用在程序推出之前的最终问题检查列表，你将学会如何测定性能，以及解决所有性能问题的技术。还会了解如何部署应用程序，以及在哪里部署的最佳实践。

## 本书网站

本书网站为 http://www.wiley.com/go/ptl/javascriptprogramming。在这里可以下载本书示例中的样例代码和演示，以及延伸资源的链接。如果某些内容你理解起来有困难，或者只是想试试样例代码，都可以到这个网站上去看看。

## 开始挑战

本书是一幅构建现代 Web 应用程序的路线图，介绍了你将面对的常见问题的解决方案，还有可以处理任何事情的通用最佳实践。本书面向高级前端开发人员，以及对 Node 感兴趣并有扎实的 JavaScript 基础的后端开发人员。JavaScript 达到中等水平的开发人员应该也能看懂书中的概念和示例（最终他们的技能将会有长足的进步）。尽管你应该充分理解客户端 JavaScript，但因为第三部分是从最基础的内容开始的，所以即便你没有服务器端 JavaScript 的经验，也能理解 Node 相关的内容。

那么请你准备好，跟我一起构建下一代 Web 应用程序吧！

# 目　录

## 第一部分　坚实的基础



## 第二部分　构建前端

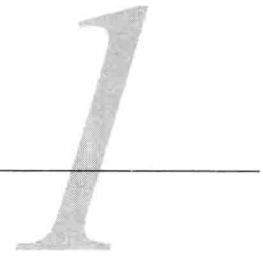
Part 1

# 第一部分

# 坚实的基础

本部分内容

# 第1章 最佳实践



坚实的基础对任何应用程序来说都至关重要。在写代码之前，必须先对应用程序的架构加以规范。程序有什么功能，将会如何实现？更重要的是这些功能彼此之间如何协作，换句话说，程序的体系是什么样的？

要回答这些问题，需要搞研究、做原型，并有坚实的最佳实践基础。尽管我不能帮你研究或实现程序中某些组件的原型，但我可以把从最佳实践中取得的经验传授给你。

本章介绍了松耦合的基本工程概念，详细解释了一个实现松耦合的办法：JavaScript MVC 和模板引擎。之后介绍一些开发工具，比如 Weinre、版本控制和 CSS 预处理。最后介绍如何在项目中设置 Grunt、让文件合并和最小化之类的工作实现自动化处理。用 Grunt 可以确立测试驱动开发的模式，无论什么时候修改了文件，都会让程序通过一组测试。

## 1.1 松耦合

如果本书只能让你在一件事上受益，我希望你能从中学会如何避免在程序中出现紧耦合。紧耦合是一个古老的工程术语，指不同组件之间的交叉依赖关系太强了。比如说你买了台电视，这台电视还内置了一台蓝光播放机。但如果电视坏了会怎么样？蓝光播放机可能还完好无损，可以正常工作，但电视一旦坏了，就不能显示画面了。从工程角度来看，最好是避免这种紧耦合的情况出现，而采用独立的、外置的蓝光播放机。

这种模式对软件开发也适用。设计应用程序时基本上应该用功能单一、相互独立的模块。通过解耦这些任务，可以将不同模块之间的依赖程度降到最低。这样可以尽可能地让每个模块保持“单纯”，能够专注于单一任务，不用考虑跟程序中其他代码之间的关系。

但决定把哪些任务分组到一个模块中可能是一个挑战。可惜这个问题没有一个放之四海而皆准的解决方案：模块太少会导致耦合度太高，模块太多又会造成不必要的抽象。最好的办法是折中：在程序中使用数量合理、内聚程度高的模块。内聚程度高的模块就是把处理单一、明确任务的相关功能组合到一起。

### 1.1.1 紧耦合的问题

紧耦合的例子在我们身边随处可见。如果你跟我一样，那你应该也已经用手机取代了音乐播放器、电子游戏机，甚至手电筒。把这么多功能集成在一个简单的设备上肯定很方便。在这种情况下，紧耦合是有意义的。但这也意味着可能会出现连锁反应，如果一个东西出了问题，其他东西也会出问题，

比如你用手机听了很长时间音乐后，可能会突然发现手电筒电量不足了。

软件开发中的紧耦合也不一定都是坏事，缺乏设计的耦合才更糟糕。应用程序中几乎总会有一些依赖关系，关键是要避免在互不相干的任务间出现不必要的耦合。如果你不努力隔离出不同的模块，就会得到一个脆弱不堪的程序，很可能会因为一个小 bug 彻底崩溃。当然，你肯定会想尽一切办法避免出现 bug，但如果每个 bug 都能搞垮整个程序，你也无能为力。

另外，调试一个紧耦合的程序极其困难。当程序整个都瘫痪时，几乎不可能准确定位到最先出现 bug 的地方。这个问题就源自常说的*空心粉式代码*。跟一堆空心粉一样，代码的脉络相互交织在一起，很难把它们分开。

### 1.1.2 松耦合的优势

即便你很少遇到 bug，松耦合仍然有些值得称道的优势。构建松耦合程序的一个主要原因实际上可以追溯到另一个经典的工程基础理念：可更换的部件。生产过程中经常要重新构建程序中的某些部分。如果谷歌开始对它的翻译 API 收费了，最好换成其他的。如果有个组件扩展性很差并在负载增加后越来越慢了，最好重新构建它。

如果程序耦合太紧密，一个模块中的变动就可能引发连锁反应，所有依赖模块都必须进行调整以适应这个变化。松耦合的程序可以避免这种额外的开发投入，把修改代码的影响限定在各自的模块内。

另外，松耦合也使得跟其他开发人员的协作更容易。如果所有组件都互不相干，就更容易实现并行工作。所有开发人员都可以放心大胆地完成自己的开发任务，而不用担心会破坏别人做的东西。

最后一点，松耦合的程序测试起来也更容易。如果程序的每一部分都处理一个单独、具体的任务，很容易设置单元测试来确保这些任务在任何情况下都能正确执行。本章后续内容中对单元测试有更多的介绍。

> 理想情况下，代码永远都不需要重构。但即便你能考虑到所有可能的情景，现实情况中也总有无法预见的问题出现，所以为什么要大费周章呢？试图提前解决问题会导致“过早优化”，肯定会拖慢开发速度。按照敏捷开发的理念，开发人员应该只关心眼前的问题，将来的问题将来再解决。松耦合可以精简敏捷开发过程，让代码库可以随着条件的改变自然发展。

## 1.2 JavaScript MVC 和模板

继松耦合之后，JavaScript 模型视图控制器（MVC）和模板是本书要强调的另一个设计模式。它们提供了一个可以把程序各方面解耦的结构。

### 1.2.1 MVC

MVC 是一种鼓励松耦合的设计模式。它把驱动程序的数据从显示数据的视觉界面上分离出来。采用 MVC 框架后，可以在不修改底层数据的情况下改变前端界面的风格。因为 MVC 将关注点分解到了三个相互关联的组件中：模型、视图和控制器，如图 1-1 所示。

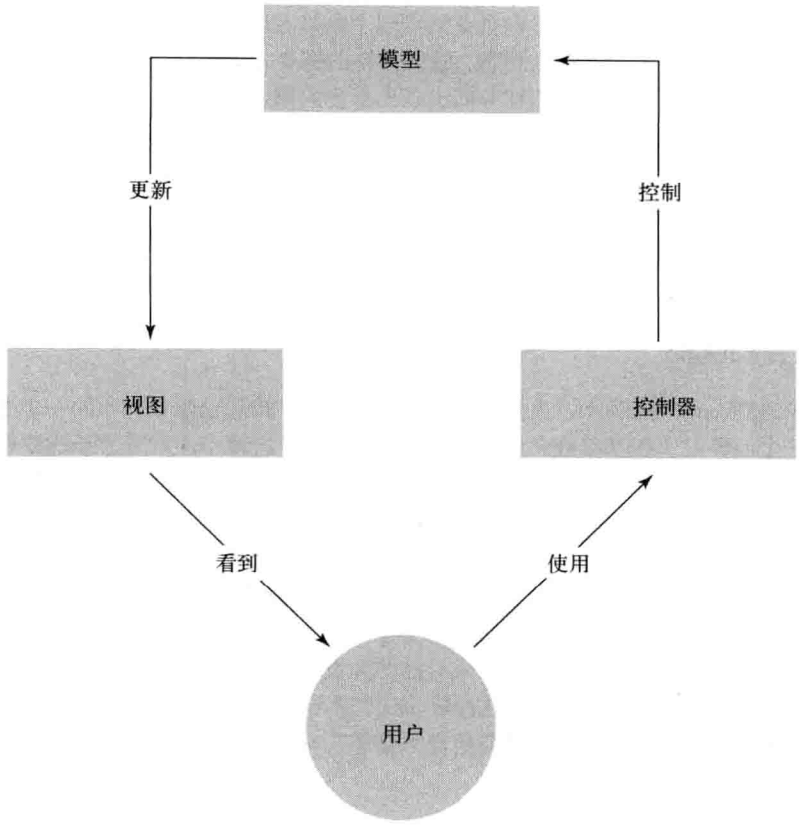



图 1-1　MVC 中 3 个组件之间的关系

### 1. 模型

MVC 中的组件模型就是驱动程序的数据。你可以把模型层看作程序的域逻辑层，它是程序要处理的全部数据。在一个简单的介绍性网站中，模型层可能只包含几个表示网站内容的对象：文本、图片路径，等等。而比较复杂的程序通常会用大量的模型来表示程序所需的各种数据。其中一个例子就是表示用户的模型（用户名、密码，等等）。

> 模型中的数据通常会保存在数据库中，或者本地存储这样的数据存储库中。那样数据能跨越多个会话存留。

### 2. 视图

视图的职责就是展示用户界面。在 Web 程序中，视图产生的最终结果就是 HTML 标记，因为它要把内容显示在屏幕上。但视图要比简单的标记复杂，它必须重新处理来自模型中的数据，将其变成可以作为 HTML 标记呈现的格式。