

SAMS
**Teach
Yourself**

- 全球销量逾百万册的系列图书
- 连续十余年打造的经典品牌
- 直观、循序渐进的学习教程
- 掌握关键知识的最佳起点
- “Read Less, Do More”（精读多练）的教学理念
- 以示例引导读者完成最常见的任务

每章内容针对初学者精心设计，**1**小时轻松阅读学习，
24小时彻底掌握关键知识

每章**案例与练习题**助你轻松完成常见任务，
通过**实践**提高应用技能，巩固所学知识

Node.js

入门经典

[英] George Ornbo 著
傅强 陈宗斌 译

 人民邮电出版社
POSTS & TELECOM PRESS

Node.js

入门经典

[英] George Ornbo 著
傅强 陈宗斌 译



人民邮电出版社
北京

图书在版编目(CIP)数据

Node.js入门经典 / (英) 奥尔波 (Ornbo,G.) 著 ;
傅强, 陈宗斌译. — 北京 : 人民邮电出版社, 2013. 4
ISBN 978-7-115-31107-8

I. ①N… II. ①奥… ②傅… ③陈… III. ①
JAVA语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2013)第034588号

版 权 声 明

George Ornbo: Sams Teach Yourself Node.js in 24 Hours

ISBN: 0672335956

Copyright © 2013 by Pearson Education, Inc.

Authorized translation from the English languages edition published by Pearson Education, Inc.

All rights reserved.

本书中文简体字版由美国 Pearson 公司授权人民邮电出版社出版。未经出版者书面许可, 对本书任
何部分不得以任何方式复制或抄袭。

版权所有, 侵权必究。



Node.js 入门经典

- ◆ 著 [英] George Ornbo
译 傅 强 陈宗斌
责任编辑 傅道坤
- ◆ 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>
北京鑫正大印刷有限公司印刷
- ◆ 开本: 787×1092 1/16
印张: 22
字数: 544千字 2013年4月第1版
印数: 1-3500册 2013年4月北京第1次印刷

著作权合同登记号 图字: 01-2012-7073号

ISBN 978-7-115-31107-8

定价: 59.00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

内容提要

Node.js 是一套用来编写高性能网络服务器的 JavaScript 工具包，从 2009 年诞生之日起，就获得了业内专家和技术社区的强烈关注。而本书采用直观、循序渐进的方法对如何使用 Node.js 来开发及具速度和可扩展性优势的服务器端应用程序进行了讲解。

本书分为 6 部分，第 1 部分介绍了 Node.js 的基本概念和特性；第 2 部分讲解如何借助 HTTP 模块和 Express Web 框架，使用 Node.js 创建基本的网站；第 3 部分介绍了调试和测试 Node.js 应用程序的工具，以及部署 Node.js 应用的方法；第 4 部分讲解了 Node.js 实现实时编程的能力以及 Socket.IO；第 5 部分介绍了 Node.js API 以及构建 Node.js 应用程序所使用的组件；第 6 部分则介绍了 CoffeeScript 这款 Java 预编译器的知识，以及如何在 Node.js 中使用中间件、Backbone.js 来创建单页面应用的知识。

本书内容循序渐进、深入浅出、步骤详尽，而且附有大量适合动手实践的示例，可帮助读者在最短的时间内掌握 Node.js。本书适合对 Node.js 感兴趣的零基础人员阅读，也适合对 Web 前端开发、后端开发感兴趣的技术人员阅读。



关于作者

George Ornbo 是英国的一位 JavaScript 和 Ruby 开发人员。他开发 Web 应用程序已有 8 年时间，一开始是以自由职业者的身份工作，最近则为伦敦的 `pebble` 工作。他的博客地址是 <http://shapedshed.com>，在网络中大多数常见的地方他都以@shapedshed 出现。

献辞

本书献给我的妻子 Kirsten。没有你的支持，本书就不可能问世。

致谢

感谢 Trina MacDonald 和 Pearson 团队给了我编写本书的机会。你们的鼓励和引导是无价之宝。

感谢本书的技术编辑 Remy Sharp。你找出了大量错误并监督了评审过程。我欠你一顿酒！本书中若还有错误，那肯定是我自己的过失。

感谢我在 pebble {code} 的同事。在本书写作伊始，你们就是我的后盾。你们允许我在大型项目中灵活安排时间，让我得以完成本书，我深表感激。

前 言

Node.js 可以让开发人员在服务器上使用 JavaScript, 这让熟悉 JavaScript 的开发人员又多了一种服务器端的开发技能, 但 Node.js 并非仅限于此。它重新思考了在现代 Web 环境下的网络编程, 在这个环境下, 应用程序可能需从许多不同的地方读写数据, 也可能有上百万个并发用户。

在具有传统的计算机科学学位的开发人员眼中, JavaScript 就是一种玩具语言。但是, JavaScript 已经经历了无数次的挑战, 而且如今在 Web 的浏览器和服务器端(借助于 Node.js)中已经不可或缺。现在是编写 JavaScript (尤其是在服务器上)的最好时节!

Node.js 表示一个开发平台, 在创建适用于现在 Web 的应用程序时, Node.js 大有裨益, 这些应用程序包括:

- 实时应用程序;
- 多人游戏;
- 单页面应用程序;
- 基于 JSON 的 API。

Node.js 专注于速度和可扩展性, 而且在无需昂贵硬件的情况下, 能处理上千个并发用户的需求。Node.js 项目最近成为 GitHub 上最受关注的项目, 如今, eBay、LinkedIn 和 Microsoft 这样的公司已经开始使用它。

Node.js 绝不仅仅只是服务器上的 JavaScript。它是一个功能齐全的网络编程平台, 能够针对现代 Web 编程的需求做出响应。

本书读者对象

本书并不要求读者一定具备编程经验, 但是如果有基本的 JavaScript 使用经验会更好。由于 Node.js 主要从终端运行, 因此知道什么是终端, 以及如何运行基本的命令也会更有帮助。最后, 由于 Node.js 主要是一个网络编程工具, 因此, 如果读者对 Internet 的运行机制略知一二, 会更好。

学习 Node.js 的理由

如果读者对创建有许多用户、处理联网数据或者有实时要求的应用程序感兴趣，那么 Node.js 是完成这些任务的极佳工具。此外，如果为浏览器创建应用程序，Node.js 可以让服务器是 JavaScript 的，这可以简化服务器和客户端之间的数据共享。Node.js 是现代 Web 的现代工具箱。

本书组织结构

本书首先讲解了 Node.js 的基础知识，包括运行你的第一个 Node.js 程序以及使用 npm (Node 包管理器)，然后介绍了网络编程，以及 Node.js 使用 JavaScript 回调来支持异步编程风格的方法。

在本书第 2 部分，我们将学习如何通过使用 HTTP 模块和 Express (一个 Node.js 的 Web 框架)，并借助 Node.js 创建基本的网站。我们还将学习如何使用 MongoDB 来让数据持久化。

第 3 部分介绍用于调试和测试 Node.js 应用程序的工具，其中介绍了许多用来支持开发的调试工具和测试框架。我们还将学习如何将 Node.js 应用程序部署到许多第三方服务上，包括 Heroku 和 Nodester。

第 4 部分讲解 Node.js 的实时能力并介绍 Socket.IO。我们将学习如何在浏览器和服务器之间发送消息，并构建一个完整的聊天服务器示例和一个实时的 Twitter 客户端。最后我们将学习如何使用 Node.js 创建 JSON API。

第 5 部分以 Node.js API 为主，并讲解用于创建 Node.js 应用程序的构件 (building block)。我们将学习进程、子进程、事件、缓冲区和流。

第 6 部分介绍的是读者可能想了解的一些高级主题。我们将学习 CoffeeScript 这个 JavaScript 预编译器，Node.js 如何使用中间件，以及如何使用 Backbone.js 与 Node.js 一起创建单页面应用程序。第 22 章将介绍如何使用 npm 编写并发布你自己的 Node.js 模块。

代码示例

本书每章都带有几个代码示例。这些示例旨在帮助读者更好地理解 Node.js。读者可从 <http://bit.ly/nodejsbook-examples> 下载这些代码，也可从 <https://github.com/shapeshed/nodejsbook.io.examples> 的 GitHub 库下载。

目 录

第 1 部分 入门

第 1 章 Node.js 介绍	2	2.6.1 本地安装	13
1.1 什么是 Node.js	2	2.6.2 全局安装	13
1.2 使用 Node.js 能做什么	3	2.7 如何找模块文档	14
1.3 安装并创建第一个 Node.js 程序	3	2.8 使用 package.json 指定依赖关系 (dependency)	14
1.3.1 验证 Node.js 正确安装	4	2.9 小结	16
1.3.2 创建“Hello World” Node.js 程序	4	2.10 问与答	16
1.4 小结	5	2.11 测验	16
1.5 问与答	6	2.11.1 问题	16
1.6 测验	6	2.11.2 答案	17
1.6.1 问题	6	2.12 练习	17
1.6.2 答案	7		
1.7 练习	7	第 3 章 Node.js 的作用	18
第 2 章 npm (Node 包管理器)	8	3.1 设计 Node.js 的目的	18
2.1 npm 是什么	8	3.2 理解 I/O	19
2.2 安装 npm	9	3.3 处理输入	19
2.3 安装模块	9	3.4 联网的 I/O 是不可预测的	22
2.4 使用模块	10	3.5 人类是不可预测的	24
2.5 如何找模块	11	3.6 处理不可预测性	25
2.5.1 官方来源	11	3.7 小结	26
2.5.2 非官方来源	12	3.8 问与答	26
2.6 本地和全局的安装	13	3.9 测验	27
		3.9.1 问题	27
		3.9.2 答案	27
		3.10 练习	27

第 4 章 回调 (Callback)	29	6.5.5 routes	58
4.1 什么是回调	29	6.5.6 views	58
4.2 剖析回调	33	6.6 介绍 Jade	59
4.3 Node.js 如何使用回调	34	6.6.1 使用 Jade 定义页面结构	60
4.4 同步和异步代码	36	6.6.2 使用 Jade 输出数据	62
4.5 事件循环	39	6.7 小结	68
4.6 小结	39	6.8 问与答	68
4.7 问与答	39	6.9 测验	68
4.8 测验	40	6.9.1 问题	69
4.8.1 问题	40	6.9.2 答案	69
4.8.2 答案	40	6.10 练习	69
4.9 练习	40	第 7 章 深入 Express	70
第 2 部分 使用 Node.js 的基本网站		7.1 Web 应用程序中的路由	70
第 5 章 HTTP	44	7.2 在 Express 中路由如何工作	70
5.1 什么是 HTTP	44	7.3 添加 GET 路由	71
5.2 使用 Node.js 的 HTTP 服务器	44	7.4 添加 POST 路由	72
5.2.1 一个基础的服务器	44	7.5 在路由中使用参数	73
5.2.2 加入头 (Header)	45	7.6 让路由保持可维护性	74
5.2.3 检查响应头	46	7.7 视图渲染	75
5.2.4 Node.js 中的重定向	49	7.8 使用本地变量	76
5.2.5 响应不同的请求	50	7.9 小结	78
5.3 使用 Node.js 的 HTTP 客户端	52	7.10 问与答	78
5.4 小结	53	7.11 测验	78
5.5 问与答	53	7.11.1 问题	79
5.6 测验	54	7.11.2 答案	79
5.6.1 问题	54	7.12 练习	79
5.6.2 答案	54	第 8 章 数据的持久化	80
5.7 练习	54	8.1 什么是持久的数据	80
第 6 章 Express 介绍	55	8.2 将数据写入文件	81
6.1 什么是 Express	55	8.3 从文件读取数据	82
6.2 为什么使用 Express	55	8.4 读取环境变量	83
6.3 安装 Express	56	8.5 使用数据库	84
6.4 创建一个基础的 Express 站点	56	8.5.1 关系数据库	84
6.5 探索 Express	58	8.5.2 NoSQL 数据库	85
6.5.1 app.js	58	8.6 在 Node.js 中使用 MongoDB	85
6.5.2 node_modules	58	8.6.1 安装 MongoDB	86
6.5.3 package.json	58	8.6.2 连接 MongoDB	87
6.5.4 public	58	8.6.3 定义文档	89
		8.6.4 将 Twitter Bootstrap 包含进来	90

8.6.5 索引 (Index) 视图	91	第 11 章 部署 Node.js 应用程序	133
8.6.6 创建 (Create) 视图	93	11.1 准备好部署	133
8.6.7 编辑视图	95	11.2 在云上托管	133
8.6.8 删除任务	98	11.3 Heroku	135
8.6.9 添加闪出消息	99	11.3.1 注册 Heroku	135
8.6.10 验证输入的数据	101	11.3.2 为 Heroku 准备应用程序	136
8.7 小结	102	11.3.3 将应用程序部署到 Heroku	137
8.8 问与答	103	11.4 Cloud Foundry	138
8.9 测验	103	11.4.1 注册 Cloud Foundry	138
8.9.1 问题	103	11.4.2 为 Cloud Foundry 准备应用程序	139
8.9.2 答案	103	11.4.3 将应用程序部署到 Cloud Foundry	140
8.10 练习	104	11.5 Nodester	141
第 3 部分 调试、测试与部署		11.5.1 注册 Nodester	141
第 9 章 调试 Node.js 应用程序		11.5.2 为 Nodester 准备应用程序	142
9.1 调试	106	11.5.3 将应用程序部署到 Nodester	143
9.2 STUDIO 模块	107	11.6 其他 PaaS 提供商	144
9.3 Node.js 调试器	111	11.7 小结	144
9.4 Node Inspector	113	11.8 问与答	144
9.5 关于测试的注释	116	11.9 测验	145
9.6 小结	116	11.9.1 测验	145
9.7 问与答	116	11.9.2 答案	145
9.8 测验	117	11.10 练习	145
9.8.1 问题	117	第 4 部分 使用 Node.js 的中间站点	
9.8.2 答案	117	第 12 章 介绍 Socket.IO	
9.9 练习	117	12.1 现在要开始学习一些完全不同的技术了	148
第 10 章 测试 Node.js 应用程序		12.2 动态 Web 简史	148
10.1 为什么测试	119	12.3 Socket.IO	149
10.2 Assert (断言) 模块	120	12.4 基础的 Socket.IO 示例	150
10.3 第三方测试工具	122	12.5 从服务器发送数据到客户端	152
10.4 行为驱动的开发 (Behavior Driven Development)	125	12.6 将数据广播给客户端	156
10.4.1 Vows	125	12.7 双向数据	160
10.4.2 Mocha	128	12.8 小结	163
10.5 小结	131		
10.6 问与答	131		
10.7 测验	132		
10.7.1 问题	132		
10.7.2 答案	132		
10.8 练习	132		

12.9 问与答	163	15.4 从 JavaScript 对象创建 JSON	212
12.10 测验	164	15.5 使用 Node.js 消费 JSON 数据	213
12.10.1 问题	164	15.6 使用 Node.js 创建 JSON API	216
12.10.2 答案	164	15.6.1 在 Express 中以 JSON 发送数据	216
12.11 练习	165	15.6.2 构建应用程序	219
第 13 章 一个 Socket.IO 聊天服务器	166	15.7 小结	224
13.1 Express 和 Socket.IO	166	15.8 问与答	225
13.2 添加昵称	168	15.9 测验	225
13.2.1 将昵称发送给服务器	169	15.9.1 问题	225
13.2.2 管理昵称列表	171	15.9.2 答案	225
13.2.3 使用回调来验证	174	15.10 练习	226
13.2.4 广播昵称列表	178		
13.2.5 添加消息收发功能	179		
13.3 小结	183		
13.4 问与答	184		
13.5 测验	184		
13.5.1 问题	184		
13.5.2 答案	184		
13.6 练习	185		
第 14 章 一个流 Twitter 客户端	186		
14.1 流 API	186		
14.2 注册 Twitter	187		
14.3 和 Node.js 一起使用 Twitter 的 API	189		
14.4 从数据中挖掘含义	191		
14.5 将数据推送到浏览器	194		
14.6 创建一个实时的爱恨表	197		
14.7 小结	206		
14.8 问与答	206		
14.9 测验	206		
14.9.1 问题	206		
14.9.2 答案	206		
14.10 练习	207		
第 15 章 JSON API	208		
15.1 API	208		
15.2 JSON	209		
15.3 使用 Node.js 发送 JSON 数据	211		
		第 5 部分 探索 Node.js API	
		第 16 章 进程模块	228
		16.1 进程是什么	228
		16.2 退出进程以及进程中的 错误	230
		16.3 进程与信号	230
		16.4 向进程发送信号	231
		16.5 使用 Node.js 创建脚本	233
		16.6 给脚本传递参数	234
		16.7 小结	236
		16.8 问与答	236
		16.9 测验	237
		16.9.1 问题	237
		16.9.2 答案	237
		16.10 练习	238
		第 17 章 子进程模块	239
		17.1 什么是子进程	239
		17.2 杀死子进程	241
		17.3 与子进程通信	242
		17.4 集群 (Cluster) 模块	244
		17.5 小结	246
		17.6 问与答	246
		17.7 测验	246
		17.7.1 问题	246
		17.7.2 答案	246

17.8	练习	247	20.9	练习	279
第 18 章	事件模块	248	第 6 部分 进一步的 Node.js 开发		
18.1	理解事件	248	第 21 章	CoffeeScript	282
18.2	通过 HTTP 演示事件	251	21.1	什么是 CoffeeScript	282
18.3	用事件玩乒乓	254	21.2	安装与运行 CoffeeScript	284
18.4	动态编写事件侦听器程序	255	21.3	为什么要使用预编译器	285
18.5	小结	258	21.4	CoffeeScript 的功能	286
18.6	问与答	258	21.4.1	最小语法	286
18.7	测验	259	21.4.2	条件和比较	287
18.7.1	问题	259	21.4.3	循环	288
18.7.2	答案	259	21.4.4	字符串	289
18.8	练习	259	21.4.5	对象	290
第 19 章	缓冲区模块	260	21.4.6	类、继承和 super	291
19.1	二进制数据初步	260	21.5	调试 CoffeeScript	294
19.2	从二进制到文本	261	21.6	对 CoffeeScript 的反应	294
19.3	二进制和 Node.js	262	21.7	小结	295
19.4	Node.js 中的缓冲区是什么?	264	21.8	问与答	295
19.5	写入缓冲区	265	21.9	测验	296
19.6	向缓冲区追加数据	266	21.9.1	问题	296
19.7	复制缓冲区	267	21.9.2	答案	296
19.8	修改缓冲区中的字符串	267	21.10	练习	296
19.9	小结	268	第 22 章	创建 Node.js 模块	298
19.10	问与答	268	22.1	为什么创建模块	298
19.11	测验	268	22.2	流行的 Node.js 模块	298
19.11.1	问题	268	22.3	package.json 文件	299
19.11.2	答案	269	22.4	文件夹结构	301
19.12	练习	269	22.5	开发和测试模块	302
第 20 章	流模块	270	22.6	添加可执行文件	304
20.1	流简介	270	22.7	使用面向对象或者基于原型的编程	305
20.2	可读流	272	22.8	通过 GitHub 共享代码	306
20.3	可写流	275	22.9	使用 Travis CI	307
20.4	通过管道连接流	276	22.10	发布到 npm	309
20.5	流的 MP3	277	22.11	公开模块	310
20.6	小结	278	22.12	小结	310
20.7	问与答	278	22.13	问与答	310
20.8	测验	279	22.14	测验	311
20.8.1	问题	279	22.14.1	问题	311
20.8.2	答案	279			

22.14.2 答案	311	第 24 章 结合使用 Backbone.js 与	
22.15 练习	311	Node.js	326
第 23 章 使用 Connect 创建中间件	312	24.1 什么是 Backbone.js	326
23.1 什么是中间件	312	24.2 Backbone.js 如何工作	327
23.2 Connect 中的中间件	313	24.3 一个简单的 Backbone.js	
23.3 使用中间件的访问控制	317	视图	332
23.4 按 IP 地址限制访问	319	24.4 使用 Backbone.js 创建记录	336
23.5 将用户强制到单个域上	322	24.5 小结	337
23.6 小结	324	24.6 问与答	337
23.7 问与答	324	24.7 测验	338
23.8 测验	324	24.7.1 问题	338
23.8.1 问题	324	24.7.2 答案	338
23.8.2 答案	325	24.8 练习	338
23.9 练习	325		

第1部分 入门

第1章 Node.js 介绍

第2章 npm (Node 包管理器)

第3章 Node.js 的作用

第4章 回调 (Callback)

第 1 章

Node.js 介绍

在本章中你将学到：

- Node.js 是什么，为什么要创建它；
- 使用 Node.js 能创建的应用程序示例；
- 创建并运行第一个 Node.js 程序。

1.1 什么是 Node.js

2009 年 11 月 8 日，Ryan Dahl 在 jsconf.eu 这个关于 JavaScript 的会议上进行了一次演讲，将 Node.js 介绍给了 JavaScript 社区。他对于许多程序设计语言难以实现并发（同时做多件事情）并且经常导致糟糕的性能的问题颇为苦恼。他希望能够更容易地编写出快速的、支持许多用户并且高效地使用内存的联网软件，于是他创建了 Node.js。

在 Ryan Dahl 探索解决该问题的方法的同时，诸如 Apple 和 Google 这样的公司也开始在浏览器技术上大举投入。

鉴于 Google 依赖于浏览器来交付诸如 Gmail 这样的产品，Google 的工程师们创建了 V8，这是个给 Google Chrome 浏览器编写的 JavaScript 引擎，它也是专门为 Web 而设计的经过高度优化的软件。Google 希望 V8 能够发扬光大，于是将其以 BSD (Berkeley Software Distribution) 协议开源。

Ryan Dahl 决定使用 V8 引擎来创建 JavaScript 服务器端环境，这样做具有如下理由。

- V8 引擎极快。
- V8 专注于 Web，所以在处理超文本传输协议 (HTTP)、域名系统 (DNS) 和传输控制协议 (TCP) 等事务上驾轻就熟。

➤ JavaScript 在 Web 上人人皆知，所以大多数开发人员都能使用它。

从核心上说，Node.js 是个事件驱动的服务器端 JavaScript 环境。也就是说，我们可以像使用 PHP、Ruby 和 Python 语言那样，使用 JavaScript 创建服务器端的应用程序。对于网络以及创建与网络交互的软件，它尤为专注。

1.2 使用 Node.js 能做什么

Node.js 是个程序设计平台，只要有想法和足够的编程技艺，它就无所不能。它既可以创建对文件系统进行操作的小段脚本，也可以创建大规模的 Web 应用程序来运行整个业务。由于 Node.js 的独特设计，它非常适合于多人游戏、实时系统、联网软件和具有上千个并发用户的应用程序。

以下是一些使用 Node.js 的公司。

- LinkedIn
- eBay
- Yahoo!
- Microsoft

能使用 Node.js 创建的应用程序有：

- 实时多人游戏；
- 基于 Web 的聊天客户端；
- 将网络中的数据源进行合并的混搭软件 (Mashup)；
- 单页面浏览器应用程序；
- 基于 JSON 的 API。

注意：什么是服务器端的 JavaScript？

大多数 Web 开发人员都熟悉于使用 JavaScript 操纵浏览器中的 Web 页面并与之交互。这就是通常所称的客户端 JavaScript，因为它发生在浏览器或者客户端。服务器端 JavaScript 发生在把页面发送给浏览器之前的服务器上。当然，使用的是同样的语言！

**By the
Way**

1.3 安装并创建第一个 Node.js 程序

说得够多的了！现在来看看运行中的 Node.js 并编写你的第一个 Node.js 程序。首先得安装 Node.js。用于 Windows 和 OSX 的安装程序可以在 Node.js 的主页下载：<http://nodejs.org/>。要想在这些平台上安装 Node.js，只需下载相关文件并双击安装程序即可。如果使用 Linux 或者想手动编译 Node.js，请在 <https://github.com/joyent/node/wiki/installation> 上找操作指南。