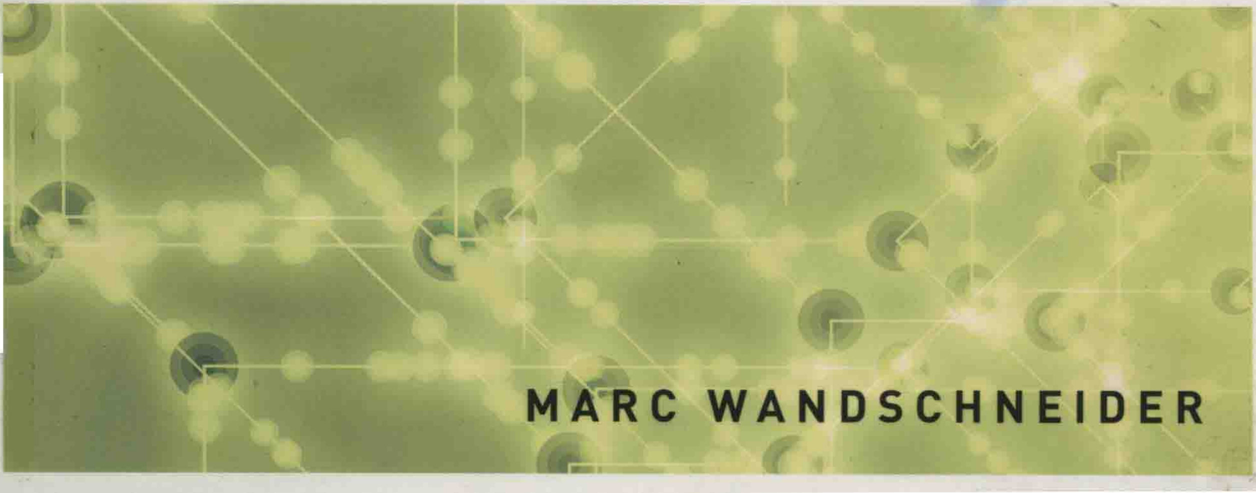
LEARNING Node.js

A Hands-On Guide to Building Web Applications in JavaScript®



MARC WANDSCHNEIDER

Learning Node.js

A Hands-On Guide to Building Web Applications in JavaScript

Marc Wandschneider

常州大学图书馆
藏书章

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales
international@pearsoned.com

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2013936406

Copyright © 2013 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-321-91057-8
ISBN-10: 0-321-91057-5

Text printed in the United States on recycled paper at RR Donnelley & Sons, Crawfordsville, Indiana.

First printing: June 2013

Editor-in-Chief
Mark Taub

Acquisitions Editor
Mark Taub

Marketing Manager
Stephane Nakib

Managing Editor
Kristy Hart

Senior Project Editor
Lori Lyons

Copy Editor
Chuck Hutchinson

Indexer
Tim Wright

Proofreader
Anne Goebel

Technical Reviewer
Bill Glover

Editorial Assistant
Olivia Basegio

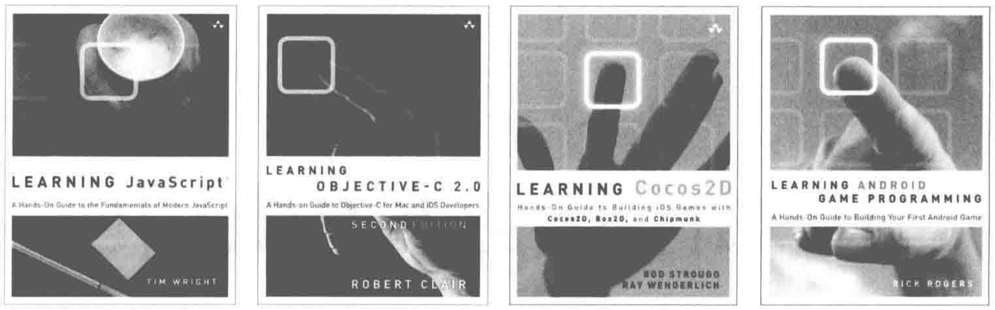
Cover Designer
Chuti Prasertsith

Compositor
Nonie Ratcliff

Manufacturing Buyer
Dan Uhrig

Learning Node.js

Addison-Wesley Learning Series


LEARNING JavaScript
A Hands-On Guide to the Fundamentals of Modern JavaScript
TIM WRIGHT
LEARNING OBJECTIVE-C 2.0
A Hands-on Guide to Objective-C for Mac and iOS Developers
SECOND EDITION
ROBERT CLAIR
LEARNING Cocos2D
Hands-On Guide to Building iOS Games with Cocos2D, Box2D, and Chipmunk
ROD STROUGO
RAY WENDERLICH
LEARNING ANDROID GAME PROGRAMMING
A Hands-On Guide to Building Your First Android Game
RICK ROGERS

Addison-Wesley

Visit **informit.com/learningseries** for a complete list of available publications.

The Addison-Wesley Learning Series is a collection of hands-on programming guides that help you quickly learn a new technology or language so you can apply what you've learned right away.

Each title comes with sample code for the application or applications built in the text. This code is fully annotated and can be reused in your own projects with no strings attached. Many chapters end with a series of exercises to encourage you to reexamine what you have just learned, and to tweak or adjust the code as a way of learning.

Titles in this series take a simple approach: they get you going right away and leave you with the ability to walk off and build your own application and apply the language or technology to whatever you are working on.

Addison-Wesley informIT | Safari Books Online

ALWAYS LEARNING PEARSON

试读结束：需要全本请在线购买： www.ertongbook.com

❖

Much love to Tina, for simply being there.

❖

Acknowledgments

I'd like to thank all the Marks at PHPTR (it's a common name, it seems) who have helped me make this book and other projects a reality. The copy editors have been brilliant and helpful.

A huge debt of gratitude is due to Bill Glover and Idriss Juhoor for their excellent technical and style reviews.

And finally, much love to Tina, for simply being there.

About the Author

Marc Wandschneider co-founded Adylitica, a leading specialist in massively scalable web and mobile app design and development. He travels the globe, consulting as a lead manager for software projects and teams. A graduate of McGill University's School of Computer Science, he spent five years at Microsoft, developing and managing developers on the Visual Basic, Visual J++, and .NET Windows Forms teams. As Contract Software Developer/Architect at SourceLabs, he built the SWiK open source Wiki platform. He authored *PHP and MySQL LiveLessons* and *Core Web Application Development with PHP and MySQL*.

Table of Contents

Introduction 1

Why Node.js? 1

The Web 1

New Technologies 2

What Exactly Is Node.js? 2

Who Is This Book For? 4

How to Use This Book 4

Download the Source Code 5

Part I: Learning to Walk 7

1 Getting Started 9

Installing Node.js 9

Installation on Windows 9

Installation on the Mac 12

Installation on Linux 14

Running Node.js and "Hello World!" 16

The Node Shell 16

Editing and Running JavaScript Files 17

Your First Web Server 17

Debugging Your Node.js Programs 19

Staying Up-to-Date and Finding Help 22

Summary 23

2 A Closer Look at JavaScript 25

Types 25

Type Basics 26

Constants 26

Numbers 27

Booleans 28

Strings 29

Objects 32

Arrays 34

Type Comparisons and Conversions 38
Functions 39
Basics 39
Function Scope 42
Language Constructs 43
Classes, Prototypes, and Inheritance 44
Prototypes and Inheritance 45
Errors and Exceptions 47
Some Important Node.js Globals 47
global 47
console 48
process 48
Summary 48

3 Asynchronous Programming 49
The Old Way of Doing Things 49
The Node.js Way of Doing Things 50
Error Handling and Asynchronous Functions 53
The callback Function and Error Handling 54
Who Am I? Maintaining a Sense of Identity 56
Being Polite—Learning to Give Up Control 58
Synchronous Function Calls 61
Summary 61

Part II: Learning to Run 63

4 Writing Simple Applications 65
Your First JSON Server 65
Returning Some Data 67
Node Pattern: Asynchronous Loops 69
Learning to Juggle: Handling More Requests 72
More on the Request and Response Objects 78
Increased Flexibility: GET Params 79
Modifying Things: POST Data 83
Receiving JSON POST Data 84
Receiving Form POST Data 87
Summary 88

试读结束：需要全本请在线购买： www.ertongbook.com

5 Modules 89
Writing Simple Modules 89
Modules and Objects 91
npm: The Node Package Manager 92
Consuming Modules 93
Searching for Modules 93
Module Caching 94
Cycles 95
Writing Modules 95
Creating Your Module 96
Developing with Your Module 101
Publishing Your Modules 102
A Module So Common It Should Be Built In 103
The Problem 104
The Solution 105
Summary 112

6 Expanding Your Web Server 113
Serving Static Content with Streams 113
Reading a File 114
Serving Static Files in a Web Server with Buffers 116
Serving Up More Than Just HTML 118
Assembling Content on the Client: Templates 120
The HTML Skeleton Page 123
Serving Static Content 124
Modifying Your URL Scheme 124
The JavaScript Loader/Bootstrapper 126
Templating with Mustache 127
Your Home Page Mustache Template 129
Putting It All Together 130
Summary 133

Part III Writing Web Applications 135

7 Building Web Applications with Express 137
Installing Express 137
Hello World in Express 138

Routing and Layers in Express 139
Routing Basics 140
Updating Your Photo Album App for Routing 141
REST API Design and Modules 144
API Design 144
Modules 146
Additional Middleware Functionality 148
Usage 148
Configurations 149
Ordering of Middleware 150
Static File Handling 151
POST Data, Cookies, and Sessions 153
Better Browser Support for PUT and DELETE 155
Compressing Output 156
HTTP Basic Authentication 157
Error Handling 158
Summary 159

8 Databases I: NoSQL (MongoDB) 161
Setting Up MongoDB 161
Installing MongoDB 161
Using Mongo DB in Node.js 162
Structuring Your Data for MongoDB 163
It's All JavaScript 163
Data Types 164
Understanding the Basic Operations 164
Connecting and Creating a Database 165
Creating Collections 165
Inserting Documents into Collections 166
Updating Document Values 167
Deleting Documents from Collections 168
Querying Collections 168
Updating Your Photo Albums App 171
Writing the Low-Level Operations 171
Modifying the API for the JSON Server 177
Updating Your Handlers 177
Adding Some New Pages to the Application 183

Recapping the App Structure 187
Summary 188

9 **Databases II: SQL (MySQL) 189**
Getting Ready 189
Installing MySQL 189
Adding the mysql Module from npm 190
Creating a Schema for the Database 190
Basic Database Operations 191
Connecting 191
Adding Queries 192
Adding Authentication to the Application 192
Updating the API to Support Users 193
Examining the Core User Data Operations 193
Updating the Express Application 197
Creating the User Handler 198
Creating the Login and Register Pages 201
Resource Pooling 204
Getting Started 204
Working with Connections 205
Authenticating the API 205
Summary 209

Part IV Getting the Most Out of Node.js 211

10 **Deployment and Development 213**
Deployment 213
Level: Basic 214
Level: Ninja 216
Multiprocessor Deployment: Using a Proxy 218
Multiple Servers and Sessions 221
Virtual Hosting 224
Built-in Support 225
Proxy Server Support 227
Securing Your Projects with HTTPS/SSL 227
Generating Test Certificates 228
Built-in Support 228
Proxy Server Support 229

Multiplatform Development 230
Locations and Configuration Files 230
Handling Path Differences 231
Summary 232

11 Command-Line Programming 233
Running Command-Line Scripts 233
UNIX and Mac 233
Windows 235
Scripts and Parameters 236
Working with Files Synchronously 237
Basic File APIs 237
Files and Stats 239
Listing Contents of Directories 240
Interacting with the User: stdin/stdout 240
Basic Buffered Input-Output 240
Unbuffered Input 241
The Readline Module 243
Working with Processes 247
Simple Process Creation 247
Advanced Process Creation with Spawn 248
Summary 251

12 Testing 253
Choosing a Framework 253
Installing *Nodeunit* *254*
Writing Tests 254
Simple Functional Tests 255
Testing Asynchronous Functionality 258
RESTful API Testing 258
Testing Secured Resources 261
Summary 262

Index 263

Introduction

Welcome to *Learning Node.js*. Node.js is an exciting new platform for writing network and web applications that has created a lot of buzz over the past couple of years and rapidly gathered a sizeable following in the developer community. In this book, I teach you more about it, why it is special, and get you up and writing Node.js programs in short order. You'll soon find that people are rather flexible with the name of Node.js and will refer to it frequently as just Node or even "node." I certainly do a lot of that in this book as well.

Why Node.js?

Node.js has arisen for a couple of primary reasons, which I explain next.

The Web

In the past, writing web applications was a pretty standard process. You have one or more servers on your machine that listens on a *port* (for example, *80* for HTTP), and when a request is received, it forks a new process or a thread to begin processing and responding to the query. This work frequently involves communicating with external services, such as a database, memory cache, external computing server, or even just the file system. When all this work is finally finished, the thread or process is returned to the pool of "available" servers, and more requests can be handled.

It is a reasonably linear process, easy to understand, and straightforward to code. There are, however, a couple of disadvantages that continue to plague the model:

1. Each of these threads or processes carries some overhead with it. On some machines, PHP + Apache can take up as much as 10–15MB per process. Even in environments where a large server runs constantly and forks threads to process the requests, each of these carries some overhead to create a new stack and execution environment, and you frequently run into the limits of the server's available memory.
2. In most common usage scenarios where a web server communicates with a database, caching server, external server, or file system, it spends most of its time sitting around doing nothing and waits for these services to finish and return their responses. While it is sitting there doing nothing, this thread is effectively "blocked" from doing anything else. The resources it consumes and the process or thread in which it runs are entirely frozen waiting for those responses to come back.

Only after the external component has finally sent back its response will that process or thread be free to finish processing, send a response to the client, and then reset to prepare for another incoming request.

So, although it's pretty easy to understand and work with, you do have a model that can be quite inefficient if your scripts spend most of their time waiting for database servers to finish running a query—an extremely common scenario for a lot of modern web applications.

Many solutions to this problem have been developed and are in common use. You can buy ever bigger and more powerful web servers with more memory. You can replace more powerful and feature-rich HTTP servers such as Apache with smaller, lightweight ones such as *lighttpd* or *nginx*. You can build stripped-down or reduced versions of your favorite web programing language such as PHP or Python. (Indeed, Facebook has taken this one step further and built a system that converts PHP to native C++ code for maximal speed and optimal size.) Or you can throw more servers at the problem to increase the number of simultaneous connections you can accommodate.

New Technologies

Although the web developers of the world have continued their eternal struggle against server resources and the limits on the number of requests they can process, a few other interesting things have happened in the meantime.

JavaScript, that old (meaning 1995 or so) language that came to be most well known (and frequently reviled) for writing client-side scripts in the web browser, has been growing in popularity again. Modern versions of web browsers are cleaning up their implementations of it and adding in new features to make it more powerful and less quirky. With the advent of client libraries for these browsers, such as jQuery, script.aculo.us, or Prototype, programming in JavaScript has become fun and productive. Unwieldy APIs have been cleaned up, and fun, dynamic effects have been added.

At the same time, a new generation of browser competition has erupted, with Google's Chrome, Mozilla's Firefox, Apple's Safari, and Microsoft's Internet Explorer all vying for the crown of browser king. As part of this, all these companies are investing heavily in the JavaScript portion of these systems as modern web applications continue to grow ever-more dynamic and script-based. In particular, Google Chrome's V8 JavaScript runtime is particularly fast and also open-sourced for use by anybody.

With all these things in place, the opportunity arose for somebody to come along with a new approach to network (web) application development. Thus, the birth of Node.js.

What Exactly Is Node.js?

In 2009, a fellow named Ryan Dahl was working for a company called Joyent, a cloud and virtualization services company in California. He was looking to develop push capabilities for

试读结束：需要全本请在线购买： www.ertongbook.com