

Node.js in Action

**MIKE CANTELON
MARC HARTER
T.J. HOLOWAYCHUK
NATHAN RAJLICH**



**MANNING
SHELTER ISLAND**

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact


Special Sales Department
Manning Publications Co.
20 Baldwin Road
PO Box 261
Shelter Island, NY 11964
Email: orders@manning.com

©2014 by Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps.

© Recognizing the importance of preserving what has been written, it is Manning's policy to have the books we publish printed on acid-free paper, and we exert our best efforts to that end. Recognizing also our responsibility to conserve the resources of our planet, Manning books are printed on paper that is at least 15 percent recycled and processed without the use of elemental chlorine.

 Manning Publications Co.
20 Baldwin Road
PO Box 261
Shelter Island, NY 11964

Development editor: Renae Gregoire
Copyeditor: Andy Carroll
Proofreader: Katie Tennant
Typesetter: Dottie Marsico
Cover designer: Marija Tudor

ISBN 9781617290572

Printed in the United States of America

1 2 3 4 5 6 7 8 9 10 – MAL – 18 17 16 15 14 13

brief contents

PART 1	NODE FUNDAMENTALS.....	1
1	■ Welcome to Node.js	3
2	■ Building a multiroom chat application	14
3	■ Node programming fundamentals	37
PART 2	WEB APPLICATION DEVELOPMENT WITH NODE	69
4	■ Building Node web applications	71
5	■ Storing Node application data	97
6	■ Connect	123
7	■ Connect's built-in middleware	145
8	■ Express	176
9	■ Advanced Express	202
10	■ Testing Node applications	242
11	■ Web application templating	264
PART 3	GOING FURTHER WITH NODE	293
12	■ Deploying Node applications and maintaining uptime	295
13	■ Beyond web servers	309
14	■ The Node ecosystem	343

contents

foreword xiii
preface xv
acknowledgments xvi
about this book xviii
about the cover illustration xx

PART 1 **NODE FUNDAMENTALS**.....1

1 **Welcome to Node.js** 3

- 1.1 Built on JavaScript 4
- 1.2 Asynchronous and evented: the browser 5
- 1.3 Asynchronous and evented: the server 7
- 1.4 DIRTY applications 8
- 1.5 DIRTY by default 10
 - Simple async example* 11 ■ *Hello World HTTP server* 12
 - Streaming data* 12
- 1.6 Summary 13

2 **Building a multiroom chat application** 14

- 2.1 Application overview 15
- 2.2 Application requirements and initial setup 17
 - Serving HTTP and WebSocket* 17 ■ *Creating the application file structure* 18 ■ *Specifying dependencies* 19 ■ *Installing dependencies* 19

- 2.3 Serving the application’s HTML, CSS, and client-side JavaScript 20
 - Creating a basic static file server* 20
 - *Adding the HTML and CSS files* 23
- 2.4 Handling chat-related messaging using Socket.IO 25
 - Setting up the Socket.IO server* 26
 - *Handling application scenarios and events* 27
- 2.5 Using client-side JavaScript for the application’s user interface 31
 - Relaying messages and name/room changes to the server* 32
 - Showing messages and available rooms in the user interface* 33
- 2.6 Summary 36

3 *Node programming fundamentals* 37

- 3.1 Organizing and reusing Node functionality 38
 - Creating modules* 40
 - *Fine-tuning module creation using module.exports* 42
 - *Reusing modules using the node_modules folder* 43
 - *Caveats* 44
- 3.2 Asynchronous programming techniques 46
 - Handling one-off events with callbacks* 46
 - *Handling repeating events with event emitters* 50
 - *Challenges with asynchronous development* 57
- 3.3 Sequencing asynchronous logic 58
 - When to use serial flow control* 59
 - *Implementing serial flow control* 61
 - *Implementing parallel flow control* 63
 - Leveraging community tools* 65
- 3.4 Summary 67

PART 2 WEB APPLICATION DEVELOPMENT WITH NODE69

4 *Building Node web applications* 71

- 4.1 HTTP server fundamentals 72
 - How Node presents incoming HTTP requests to developers* 73
 - A basic HTTP server that responds with “Hello World”* 74
 - Reading request headers and setting response headers* 75
 - Setting the status code of an HTTP response* 75

- 4.2 Building a RESTful web service 76
 - Creating resources with POST requests* 77
 - *Fetching resources with GET requests* 79
 - *Removing resources with DELETE requests* 80
- 4.3 Serving static files 81
 - Creating a static file server* 82
 - *Handling server errors* 85
 - Preemptive error handling with fs.stat* 86
- 4.4 Accepting user input from forms 87
 - Handling submitted form fields* 87
 - *Handling uploaded files using formidable* 90
 - *Calculating upload progress* 94
- 4.5 Securing your application with HTTPS 94
- 4.6 Summary 96

5 Storing Node application data 97

- 5.1 Serverless data storage 98
 - In-memory storage* 98
 - *File-based storage* 99
- 5.2 Relational database management systems 102
 - MySQL* 102
 - *PostgreSQL* 110
- 5.3 NoSQL databases 112
 - Redis* 112
 - *MongoDB* 117
 - *Mongoose* 119
- 5.4 Summary 121

6 Connect 123

- 6.1 Setting up a Connect application 124
- 6.2 How Connect middleware works 125
 - Middleware that does logging* 126
 - *Middleware that responds with “hello world”* 126
- 6.3 Why middleware ordering matters 127
 - When middleware doesn’t call next()* 128
 - *Using middleware order to perform authentication* 128
- 6.4 Mounting middleware and servers 129
 - Middleware that does authentication* 130
 - *A middleware component that presents an administration panel* 131

- 6.5 Creating configurable middleware 133
 - Creating a configurable logger middleware component* 133
 - Building a routing middleware component* 135
 - Building a middleware component to rewrite URLs* 137
- 6.6 Using error-handling middleware 138
 - Connect's default error handler* 139 ■ *Handling application errors yourself* 139 ■ *Using multiple error-handling middleware components* 141
- 6.7 Summary 144

7 *Connect's built-in middleware* 145

- 7.1 Middleware for parsing cookies, request bodies, and query strings 146
 - cookieParser(): parsing HTTP cookies* 147 ■ *bodyParser(): parsing request bodies* 150 ■ *limit(): request body limiting* 151
 - query(): query-string parser* 153
- 7.2 Middleware that implements core web application functions 154
 - logger(): logging requests* 155 ■ *favicon(): serving a favicon* 157 ■ *methodOverride(): faking HTTP methods* 158
 - vhost(): virtual hosting* 160 ■ *session(): session management* 161
- 7.3 Middleware that handles web application security 165
 - basicAuth(): HTTP Basic authentication* 165 ■ *csrf(): cross-site request forgery protection* 167 ■ *errorHandler(): development error handling* 168
- 7.4 Middleware for serving static files 170
 - static(): static file serving* 170 ■ *compress(): compressing static files* 172 ■ *directory(): directory listings* 174
- 7.5 Summary 175

8 *Express* 176

- 8.1 Generating the application skeleton 178
 - Installing the Express executable* 180 ■ *Generating the application* 180 ■ *Exploring the application* 180
- 8.2 Configuring Express and your application 183
 - Environment-based configuration* 183

- 8.3 Rendering views 185
 - View system configuration* 185
 - *View lookup* 188
 - *Exposing data to views* 189
- 8.4 Handling forms and file uploads 194
 - Implementing the photo model* 194
 - *Creating a photo upload form* 194
 - *Showing a list of uploaded photos* 197
- 8.5 Handling resource downloads 198
 - Creating the photo download route* 198
 - *Implementing the photo download route* 199
- 8.6 Summary 201

9 *Advanced Express* 202

- 9.1 Authenticating users 203
 - Saving and loading users* 204
 - *Registering new users* 209
 - Logging in registered users* 214
 - *User-loading middleware* 217
- 9.2 Advanced routing techniques 219
 - Validating user content submission* 219
 - *Route-specific middleware* 223
 - *Implementing pagination* 225
- 9.3 Creating a public REST API 229
 - Designing the API* 229
 - *Adding Basic authentication* 229
 - Implementing routing* 230
 - *Enabling content negotiation* 234
- 9.4 Error handling 236
 - Handling 404 errors* 236
 - *Handling errors* 238
- 9.5 Summary 241

10 *Testing Node applications* 242

- 10.1 Unit testing 243
 - The assert module* 244
 - *Nodeunit* 247
 - *Mocha* 249
 - Vows* 254
 - *Should.js* 256
- 10.2 Acceptance testing 258
 - Tobi* 259
 - *Soda* 260
- 10.3 Summary 262

11 *Web application templating* 264

- 11.1 Using templating to keep code clean 265
 - Templating in action* 266

- 11.2 Templating with Embedded JavaScript 269
 - Creating a template* 269
 - *Manipulating template data using EJS filters* 271
 - *Integrating EJS into your application* 274
 - Using EJS for client-side applications* 275
- 11.3 Using the Mustache templating language with Hogan 276
 - Creating a template* 276
 - *Mustache tags* 277
 - *Fine-tuning Hogan* 279
- 11.4 Templating with Jade 280
 - Jade basics* 281
 - *Logic in Jade templates* 284
 - *Organizing Jade templates* 287
- 11.5 Summary 290

PART 3 GOING FURTHER WITH NODE293

12 *Deploying Node applications and maintaining uptime* 295

- 12.1 Hosting Node applications 295
 - Dedicated and virtual private servers* 297
 - *Cloud hosting* 297
- 12.2 Deployment basics 299
 - Deploying from a Git repository* 300
 - *Keeping Node running* 300
- 12.3 Maximizing uptime and performance 301
 - Maintaining uptime with Upstart* 302
 - *The cluster API: taking advantage of multiple cores* 304
 - *Hosting static files and proxying* 306
- 12.4 Summary 307

13 *Beyond web servers* 309

- 13.1 Socket.IO 310
 - Creating a minimal Socket.IO application* 310
 - *Using Socket.IO to trigger page and CSS reloads* 312
 - *Other uses of Socket.IO* 315
- 13.2 TCP/IP networking in depth 316
 - Working with buffers and binary data* 316
 - *Creating a TCP server* 318
 - *Creating a TCP client* 321
- 13.3 Tools for interacting with the operating system 323
 - The process global singleton* 324
 - *Using the filesystem module* 327
 - *Spawning external processes* 331

- 13.4 Developing command-line tools 336
 - Parsing command-line arguments* 336
 - *Working with stdin and stdout* 337
 - *Adding colored output* 339
- 13.5 Summary 342

14 *The Node ecosystem* 343

- 14.1 Online resources for Node developers 344
 - Node and module references* 344
 - *Google Groups* 345
 - IRC* 346
 - *GitHub issues* 346
 - 14.2 GitHub 347
 - Getting started on GitHub* 348
 - *Adding a project to GitHub* 349
 - *Collaborating using GitHub* 352
 - 14.3 Contributing to the npm repository 354
 - Preparing a package* 355
 - *Writing a package specification* 355
 - Testing and publishing a package* 356
 - 14.4 Summary 358
- appendix A* *Installing Node and community add-ons* 359
- appendix B* *Debugging Node* 367
- appendix C* *Extending and configuring Express* 374
- index* 379