## Node.js Design Patterns

Third Edition

Design and implement production-grade Node.js applications using proven patterns and techniques

Mario Casciaro Luciano Mammino



## Node.js Design Patterns

Third Edition

Copyright © 2020 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Producer: Tushar Gupta Acquisition Editor – Peer Reviews: Suresh Jain Project Editor: Tom Jacob Content Development Editors: Joanne Lovell, Bhavesh Amin Copy Editor: Safis Editing Technical Editor: Saby D'silva Proofreader: Safis Editing Indexer: Manju Arasan Presentation Designer: Sandip Tadge

First published: December 2014

Second Edition: July 2016

Third Edition: July 2020

Production reference: 1240720

Published by Packt Publishing Ltd. Livery Place 35 Livery Street Birmingham B3 2PB, UK.

ISBN 978-1-83921-411-0

www.packt.com

## Table of Contents

Preface	xi
Chapter 1: The Node.js Platform	1
The Node.js philosophy	2
Small core	2
Small modules	2
Small surface area	3
Simplicity and pragmatism	4
How Node.js works	5
I/O is slow	5
Blocking I/O	5
Non-blocking I/O	6
Event demultiplexing	7
The reactor pattern	9
Libuv, the I/O engine of Node.js	11
The recipe for Node.js	12
JavaScript in Node.js	13
Run the latest JavaScript with confidence	13
The module system	14
Full access to operating system services	14
Running native code	15
Summary	16
Chapter 2: The Module System	17
The need for modules	18
Module systems in JavaScript and Node.js	19
The module system and its patterns	20
The revealing module pattern	20

CommonJS modules	22
A homemade module loader	22
Defining a module	24
module.exports versus exports	25
The require function is synchronous	26
The resolving algorithm	26
The module cache	28
Circular dependencies	29
Module definition patterns	33
Named exports	33
Exporting a function	34
Exporting a class	35
Exporting an instance	36
Modifying other modules or the global scope	37
ESM: ECMAScript modules	38
Using ESM in Node.is	39
Named exports and imports	39
Default exports and imports	42
Mixed exports	43
Module identifiers	45
Async imports	45
Module loading in depth	48
Loading phases	48
Read-only live bindings	49
Circular dependency resolution	50
Modifying other modules	56
ESM and CommonJS differences and interoperability	60
ESM runs in strict mode	60
Missing references in ESM	60
Interoperability	61
Summary	62
Chapter 3: Callbacks and Events	63
The Callback pattern	64
The continuation-passing style	64
Synchronous CPS	65
Asynchronous CPS Non-CPS callbacks	67 67
Synchronous or asynchronous?	67
An unpredictable function	68
Unleashing Zalgo	68
Using synchronous APIs	70
Guaranteeing asynchronicity with deferred execution	72

\_

Node.js callback conventions	73
The callback comes last	73
Any error always comes first	74 74
Uncaught exceptions	74
The Observer pattern	77
The EventEmitter	78
Creating and using the EventEmitter	79
Propagating errors	80
Making any object observable	80
EventEmitter and memory leaks	82
Synchronous and asynchronous events	83
EventEmitter versus callbacks	85
Combining callbacks and events	86
Summary	88
Exercises	88
Chapter 4: Asynchronous Control Flow Patterns with Callbacks	89
The difficulties of asynchronous programming	90
Creating a simple web spider	90
Callback hell	93
Callback best practices and control flow patterns	94
Callback discipline	95
Applying the callback discipline	95
Sequential execution	98
Executing a known set of tasks in sequence	99
Sequential iteration	100
Parallel execution	104
The nattern	106
Fixing race conditions with concurrent tasks	108
Limited parallel execution	110
Limiting concurrency	112
Globally limiting concurrency	113
The async library	119
Summary	120
Exercises	121
Chapter 5: Asynchronous Control Flow Patterns	
with Promises and Async/Await	123
Promises	124
What is a promise?	125
Promises/A+ and thenables	127
The promise API	128

-

Creating a promise	130
Promisification	131
Sequential execution and iteration	133
Parallel execution	136
Limited parallel execution	137
Implementing the TaskQueue class with promises	138
Updating the web spider	139
Async/await	141
Async functions and the await expression	141
Error handling with async/await	143
A unified trycatch experience	143
The "return" versus "return await" trap	144
Sequential execution and iteration	145
Parallal execution	147
	147
The problem with infinite requiring promise resolution obside	149
The problem with minute recursive promise resolution chains	152
Summary	156
Exercises	15/
Chapter 6: Coding with Streams	159
Discovering the importance of streams	160
Buffering versus streaming	160
Spatial efficiency	161
Gzipping using a buffered API	162
Gzipping using streams	163
	163
Composability	167
Adding client-side encryption	167
Getting started with streams	170
Anatomy of streams	170
Readable streams	170
Reading from a stream	171
Implementing Readable streams	174
Writable streams	179
Writing to a stream	179
Backpressure	181
Implementing Writable streams	182
Duplex streams	185
Transform streams	185
Implementing Transform streams	186
Pritering and aggregating data with mansionin streams	109
Cass HILOUGH Suleans Observability	193
Cool vability	100
[iv]	

Late piping	194
Lazy streams	197
Connecting streams using pipes	198
Pipes and error handling Better error handling with nineline()	200
Asynchronous control flow patterns with streams	201
Sequential execution	203
Unordered parallel execution	200
Implementing an unordered parallel stream	200
Implementing a URL status monitoring application	208
Unordered limited parallel execution	210
Ordered parallel execution	212
Piping patterns	214
Combining streams	214
Implementing a combined stream	217
Forking streams	219
Implementing a multiple checksum generator	220
Merging Streams	ZZ I 221
Multiplexing and demultiplexing	221
Building a remote logger	224
Multiplexing and demultiplexing object streams	229
	000
Summary	230
Summary Exercises	230 230
Summary Exercises Chapter 7: Creational Design Patterns	230 230 233
Summary Exercises <u>Chapter 7: Creational Design Patterns</u> Factory	230 230 233 234
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation	230 230 233 234 235
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation	230 230 233 234 235 236
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler	230 230 233 234 235 236 238
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild	230 230 233 234 235 236 238 241
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder	230 230 233 234 235 236 238 241 241 241
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder Implementing a URL object builder	230 230 233 234 235 236 238 241 241 244
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder Implementing a URL object builder In the wild	230 230 233 234 235 236 238 241 241 244 244
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder Implementing a URL object builder In the wild Revealing Constructor	230 230 233 234 235 236 238 241 241 241 244 248 249
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder Implementing a URL object builder In the wild Revealing Constructor Building an immutable buffer	230 230 233 234 235 236 238 241 241 241 244 248 249 250
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder Implementing a URL object builder In the wild Revealing Constructor Building an immutable buffer In the wild	230 230 233 234 235 236 238 241 244 244 244 248 249 250 253
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder Implementing a URL object builder In the wild Revealing Constructor Building an immutable buffer In the wild Singleton	230 230 233 234 235 236 238 241 241 244 244 244 244 248 249 250 253 253
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder Implementing a URL object builder In the wild Revealing Constructor Building an immutable buffer In the wild Singleton Wiring modules	230 230 233 234 235 236 238 241 241 244 244 248 249 250 253 253 253
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder In the wild Revealing Constructor Building an immutable buffer In the wild Singleton Wiring modules Singleton dependencies	230 233 233 234 235 236 238 241 241 244 248 249 250 253 253 253 253
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder In the wild Revealing Constructor Building an immutable buffer In the wild Singleton Wiring modules Singleton dependencies Dependency Injection	230 233 233 234 235 236 238 241 244 244 248 249 250 253 253 253 253 253 253
Summary Exercises Chapter 7: Creational Design Patterns Factory Decoupling object creation and implementation A mechanism to enforce encapsulation Building a simple code profiler In the wild Builder Implementing a URL object builder In the wild Revealing Constructor Building an immutable buffer In the wild Singleton Wiring modules Singleton dependencies Dependency Injection Summary	230 233 234 235 236 238 241 241 244 244 248 249 250 253 253 253 253 253 253 253

Chapter 8: Structural Design Patterns	269
Proxy	269
Techniques for implementing proxies	271
Object composition	272
Object augmentation	275
I ne built-in Proxy object	277
Creating a logging Writable stream	200
Change observer with Proxy	201
In the wild	202
	200
Techniques for implementing descriptors	203
Composition	200
Object augmentation	288
Decorating with the Proxy object	289
Decorating a LevelUP database	290
Introducing LevelUP and LevelDB	290
Implementing a LevelUP plugin	291
In the wild	293
The line between proxy and decorator	294
Adapter	294
Using LevelUP through the filesystem API	295
In the wild	298
Summary	299
Exercises	300
Chapter 9: Behavioral Design Patterns	301
Strategy	302
Multi-format configuration objects	304
In the wild	308
State	308
Implementing a basic failsafe socket	310
Template	315
A configuration manager template	316
In the wild	318
Iterator	319
The iterator protocol	319
The iterable protocol	322
Iterators and iterables as a native JavaScript interface	324
Generators	326
Generators in theory	327
A simple generator function	327
How to use generators in place of iterators	320
5 1	
[ vi ]	

Async generators 334   Async iterators and Node.js streams 335   In the wild 336   Middleware 337   Middleware in Express 337   Middleware a pattern 338   Creating a middleware framework for ZeroMQ 340   The Middleware a pattern 338   Creating a middleware to process messages 344   In the wild 347   Command 349   A more complex command 349   A more complex command 349   Summary 353   Exercises 354   Chapter 10: Universal JavaScript for Web Applications 357   Sharing code with the browser 358   JavaScript modules in a cross-platform context 359   Module bundlers 363   Using webpack 369   Fundamentals of cross-platform development 377   Design patterns for cross-platform development 378   Build-time code branching 377   Design patterns for cross-platform development 378   Abrief introduction to React 379   Hello React <	Async iterators	331
Async iterators and Node js streams 335 In the wild 336 Middleware 337 Middleware in Express 337 Middleware in Express 337 Middleware in Express 337 Middleware as a pattern 338 Creating a middleware framework for ZeroMQ 340 The Middleware Manager 340 Implementing the middleware to process messages 342 Using the ZeroMQ middleware to process messages 342 Using the ZeroMQ middleware framework 344 In the wild 347 Command 347 Command 349 Summary 353 Exercises 354 Chapter 10: Universal JavaScript for Web Applications 357 Sharing code with the browser 358 JavaScript modules in a cross-platform context 359 Module bundlers 360 How a module bundler works 363 Using webpack 369 Fundamentals of cross-platform development 371 Runtime code branching 372 Challenges of runtime code branching 377 Design patterns for cross-platform development 378 Module swapping 374 Module swapping 374 Module swapping 377 Design patterns for cross-platform development 378 Stateful components 385 Creating a Universal JavaScript app 392 Server-side rendering 393 Asynchronous data retrieval 411 Two-pass rendering 412 Async pages 414 Implementing async pages 414 Implementing async pages 414	Async generators	334
In the wild 336   Middleware in Express 337   Middleware in Express 337   Middleware as a pattern 338   Creating a middleware framework for ZeroMQ 340   The Middleware Manager 340   Implementing the middleware to process messages 342   Using the ZeroMQ middleware framework 344   In the wild 347   The Task pattern 349   A more complex command 349   Summary 353   Exercises 354   Chapter 10: Universal JavaScript for Web Applications 357   Sharing code with the browser 358   JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 363   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 372   Challenges of runtime code branching 374   Module swapping 377   Design patterns for cross-platform development 378   A brief introduction to React 381   Alte	Async iterators and Node.js streams	335
Middleware 337   Middleware in Express 337   Middleware as a pattern 338   Creating a middleware framework for ZeroMQ 340   The Middleware Manager 340   Implementing the middleware framework 344   In the wild 347   Command 349   A more complex command 349   A more complex command 349   Summary 353   Exercises 354   Chapter 10: Universal JavaScript for Web Applications 357   Sharing code with the browser 358   JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 360   How a module bundler works 360   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 373   Build-time code branching 374   Module swapping 377   Design patterns for cross-platform development 377   A brief introduction to React 379   Hello React 381	In the wild	336
Middleware in Express 337   Middleware in Express 338   Creating a middleware framework for ZeroMQ 340   The Middleware Manager 340   Implementing the middleware to process messages 342   Using the ZeroMQ middleware framework 344   In the wild 347   Command 344   The Task pattern 349   A more complex command 349   Summary 353   Exercises 354   Chapter 10: Universal JavaScript for Web Applications 357   Sharing code with the browser 358   JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 363   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 377   Design patterns for cross-platform development 378   A brief introduction to React 379   Hello React 381   Alternatives to react.createElement 383   Stateful components 385   Croating a Univer	Middleware	337
Middleware as a pattern338Creating a middleware framework for ZeroMQ340The Middleware Manager340Implementing the middleware to process messages342Using the ZeroMQ middleware framework344In the wild347Command347The Task pattern349A more complex command349Summary353Exercises354Chapter 10: Universal JavaScript for Web Applications357Sharing code with the browser358JavaScript modules in a cross-platform context369Houda wodules of cross-platform development371Runtime code branching372Challenges of runtime code branching373Build-time code branching377Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement385Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering392Server-side rendering392Server-side rendering392Server-side rendering392Server-side rendering392Server-side rendering392Server-side rendering392Server-side rendering392Server-side rendering392Server-side rendering412Async pages416 <tr< td=""><td>Middleware in Express</td><td>337</td></tr<>	Middleware in Express	337
Creating a middleware tramework for ZeroMQ 340   The Middleware Manager 340   Implementing the middleware to process messages 342   Using the ZeroMQ middleware framework 344   In the wild 347   Command 349   A more complex command 349   A more complex command 349   Summary 353   Exercises 354   Chapter 10: Universal JavaScript for Web Applications 357   Sharing code with the browser 358   JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 360   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 372   Challenges of runtime code branching 373   Build-time code branching 374   Module swapping 377   Design patterns for cross-platform development 378   A brief introduction to React 379   Hello React 381   Stateful components 385   Creating a U	Middleware as a pattern	338
The Middleware Manager 340   Implementing the middleware to process messages 342   Using the ZeroMQ middleware framework 344   In the wild 347   Command 349   A more complex command 349   Summary 353   Exercises 354   Chapter 10: Universal JavaScript for Web Applications 357   Sharing code with the browser 358   JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 363   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 372   Challenges of runtime code branching 373   Build-time code branching 377   Design patterns for cross-platform development 378   A brief introduction to React 383   Stateful components 383   Stateful components 383   Stateful components 383   Challementing a Universal JavaScript app 399   A prief introduction to React 381	Creating a middleware framework for ZeroMQ	340
Implementing the middleware to process messages 342   Using the ZeroMQ middleware framework 344   In the wild 347   Command 347   The Task pattern 349   A more complex command 349   Summary 353   Exercises 354   Chapter 10: Universal JavaScript for Web Applications 357   Sharing code with the browser 358   JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 363   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 373   Build-time code branching 374   Module swapping 377   Design patterns for cross-platform development 378   A brief introduction to React 381   Alternatives to react.createElement 383   Stateful components 385   Creating a Universal JavaScript app 391   Frontend-only app 392   Server-side rendering 392   Asynchronous data ret	I he Middleware Manager	340
Using the ZeroNU middleware tranework344In the wild347Command347The Task pattern349A more complex command349Summary353Exercises354Chapter 10: Universal JavaScript for Web Applications357Sharing code with the browser358JavaScript modules in a cross-platform context359Module bundlers360How a module bundler works363Using webpack363Fundamentals of cross-platform development371Runtime code branching372Challenges of runtime code branching374Module swapping377Design patterns for cross-platform development378A brief introduction to React381A brief introduction to React381A ternatives to react.createElement383Stateful components392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Implementing the middleware to process messages	342
In the wind347Command347The Task pattern349A more complex command349Summary353Exercises354Chapter 10: Universal JavaScript for Web Applications357Sharing code with the browser358JavaScript modules in a cross-platform context359Module bundlers360How a module bundler works363Using webpack369Fundamentals of cross-platform development371Runtime code branching372Challenges of runtime code branching373Build-time code branching374Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages414Implementing async pages416Summary425Exercises426	Using the ZeroMQ middleware framework	344
Command347The Task pattern349A more complex command349Summary353Exercises354Chapter 10: Universal JavaScript for Web Applications357Sharing code with the browser358JavaScript modules in a cross-platform context359Module bundlers360How a module bundler works363Using webpack369Fundamentals of cross-platform development371Runtime code branching372Challenges of runtime code branching377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement385Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval414Implementing async pages414Implementing async pages414Implementing async pages414Implementing async pages416Summary425Exercises426	In the wild	347 347
A more complex command 349 Summary 353 Exercises 354 Chapter 10: Universal JavaScript for Web Applications 357 Sharing code with the browser 358 JavaScript modules in a cross-platform context 359 Module bundlers 360 How a module bundler works 363 Using webpack 369 Fundamentals of cross-platform development 371 Runtime code branching 372 Challenges of runtime code branching 373 Build-time code branching 374 Module swapping 377 Design patterns for cross-platform development 378 A brief introduction to React 379 Hello React 381 Alternatives to react.createElement 383 Stateful components 385 Creating a Universal JavaScript app 392 Server-side rendering 399 Asynchronous data retrieval 405 Universal data retrieval 405 Universal data retrieval 411 Two-pass rendering 412 Async pages 414 Implementing async pages 416 Summary 425 Exercises 426	The Teak pettern	<b>34</b> 7
Summary 353   Summary 353   Exercises 354   Chapter 10: Universal JavaScript for Web Applications 357   Sharing code with the browser 358   JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 363   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 373   Build-time code branching 374   Module swapping 377   Design patterns for cross-platform development 379   Hello React 379   Hello React 383   Stateful components 385   Creating a Universal JavaScript app 391   Frontend-only app 392   Server-side rendering 405   Universal data retrieval 405   Universal data retrieval 405   Universal data retrieval 411   Two-pass rendering 4125   Async pages 414   Implementing async pages 414   Implementing asy	A more complex command	349
Summary353Exercises354Chapter 10: Universal JavaScript for Web Applications357Sharing code with the browser358JavaScript modules in a cross-platform context359Module bundlers360How a module bundler works363Using webpack369Fundamentals of cross-platform development371Runtime code branching373Build-time code branching373Build-time code branching377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering405Universal data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Summary	349
Chapter 10: Universal JavaScript for Web Applications 357   Sharing code with the browser 358   JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 363   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 372   Challenges of runtime code branching 373   Build-time code branching 374   Module swapping 377   Design patterns for cross-platform development 378   A brief introduction to React 379   Hello React 381   Alternatives to react.createElement 383   Stateful components 385   Creating a Universal JavaScript app 399   Asynchronous data retrieval 405   Universal data retrieval 411   Two-pass rendering 412   Async pages 414   Implementing async pages 416   Summary 425   Exercises 426	Summary	353
Sharing code with the browser 357   JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 363   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 372   Challenges of runtime code branching 374   Module swapping 377   Design patterns for cross-platform development 378   A brief introduction to React 379   Hello React 381   Alternatives to react.createElement 383   Stateful components 385   Creating a Universal JavaScript app 399   Asynchronous data retrieval 405   Universal data retrieval 411   Two-pass rendering 412   Async pages 414   Implementing async pages 416   Summary 425   Exercises 426	Chanter 10: Universal JavaScrint for Web Applications	357
JavaScript modules in a cross-platform context 359   Module bundlers 360   How a module bundler works 363   Using webpack 369   Fundamentals of cross-platform development 371   Runtime code branching 372   Challenges of runtime code branching 373   Build-time code branching 377   Design patterns for cross-platform development 378   A brief introduction to React 379   Hello React 381   Alternatives to react.createElement 383   Stateful components 385   Creating a Universal JavaScript app 391   Frontend-only app 392   Server-side rendering 399   Asynchronous data retrieval 405   Universal data retrieval 411   Two-pass rendering 412   Async pages 414   Implementing async pages 416   Summary 425   Exercises 426	Chapter 10. Oniversal JavaScript for Web Applications	
Module bundlers360How a module bundler works363Using webpack369Fundamentals of cross-platform development371Runtime code branching372Challenges of runtime code branching373Build-time code branching374Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages416Summary425Exercises426	lavaScript modules in a cross platform context	350
Module bundlers363How a module bundler works363Using webpack369Fundamentals of cross-platform development371Runtime code branching373Build-time code branching373Build-time code branching374Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426		360
Tow a module works360Using webpack369Fundamentals of cross-platform development371Runtime code branching372Challenges of runtime code branching373Build-time code branching374Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages416Summary425Exercises426	How a module hundler works	363
Fundamentals of cross-platform development371Runtime code branching372Challenges of runtime code branching373Build-time code branching374Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises[vii]	Lising webnack	369
Runtime code branching372Challenges of runtime code branching373Build-time code branching374Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises[vii]	Fundamentals of cross-platform development	371
Challenges of runtime code branching373Build-time code branching374Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Runtime code branching	372
Build-time code branching374Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Challenges of runtime code branching	373
Module swapping377Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Build-time code branching	374
Design patterns for cross-platform development378A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Module swapping	377
A brief introduction to React379Hello React381Alternatives to react.createElement383Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Design patterns for cross-platform development	378
Hello React381Alternatives to react.createElement383Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	A brief introduction to React	379
Alternatives to react.createElement383Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Hello React	381
Stateful components385Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Alternatives to react.createElement	383
Creating a Universal JavaScript app391Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Stateful components	385
Frontend-only app392Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Creating a Universal JavaScript app	391
Server-side rendering399Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Frontend-only app	392
Asynchronous data retrieval405Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Server-side rendering	399
Universal data retrieval411Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Asynchronous data retrieval	405
Two-pass rendering412Async pages414Implementing async pages416Summary425Exercises426	Universal data retrieval	411
Async pages 414 Implementing async pages 416 Summary 425 Exercises 426	Two-pass rendering	412
Implementing async pages 416   Summary 425   Exercises 426    [vii]	Async pages	414
Summary   425     Exercises   426    [vii]	Implementing async pages	416
Exercises   426	Summary	425
[ vii ]	Exercises	426
	[ vii ]	

Dealing with asynchronously initialized components428The issue with asynchronously initialized components428Local initialization check429Delayed startup430Pre-initialization queues431In the wild435Asynchronous request batching and caching435What's asynchronous request batching?436Optimal asynchronous request caching437An API server without caching or batching439Batching and caching with promises441Batching requests in the total sales web server442Caching requests in the total sales web server445Canceling asynchronous operations445A basic recipe for creating cachelable functions446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Considerations on the interleaving approach459Using external processes460Deleaging the subset sum task to an external process461Considerations for the multi-process approach473Exercises473Charler 12: Scalability and Architectural Patterns475Choning the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Choriderations of the multi-process approach461Considerations of scalability	Chapter 11: Advanced Recipes	427
The issue with asynchronously initialized components 428   Local initialization check 429   Deleged startup 430   Pre-initialization queues 431   In the wild 435   Asynchronous request batching and caching 435   What's asynchronous request batching? 436   Optimal asynchronous request batching? 437   An API server without caching or batching 439   Batching and caching with promises 441   Batching requests in the total sales web server 443   Notes about implementing caching mechanisms 445   Canceling asynchronous operations 447   Cancelable async functions with generators 449   Running CPU-bound tasks 453   Solving the subset sum problem 453   Interleaving with setImmediate 457   Interleaving with setImmediate 457   Obling external processes 460   Delegating the subset sum task to an external process 461   Quising worker threads 468   Running CPU-bound tasks in production 472   Subing external processes 460   Delegating th	Dealing with asynchronously initialized components	428
Local initialization check429Delayed startup430Pre-initialization queues431In the wild435Asynchronous request batching and caching435Optimal asynchronous request batching?436Optimal asynchronous request caching437An API server without caching or batching439Batching and caching with promises441Batching requests in the total sales web server442Caching requests in the total sales web server443Notes about implementing caching mechanisms445Canceling asynchronous operations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem457Interleaving with setImmediate457Interleaving with setImmediate457Considerations on the interleaving approach459Using worker threads468Running CPU-bound tasks in a worker thread468Running CPU-bound tasks in production472Summary473Chaiderations for the multi-process approach467Using worker threads468Running CPU-bound tasks in production472Summary473Chaiderations for the multi-process approach467Using worker threads468Running CPU-bound tasks in production472Summary473Chaiderations for the multi-process approach467Considerations for the multi-process approach467 <tr< td=""><td>The issue with asynchronously initialized components</td><td>428</td></tr<>	The issue with asynchronously initialized components	428
Delayed startup430Pre-initialization queues431In the wild435Asynchronous request batching and caching435What's asynchronous request batching?436Optimal asynchronous request caching437An API server without caching or batching439Batching and caching with promises441Batching requests in the total sales web server442Caching requests in the total sales web server442Caching requests in the total sales web server443Notes about implementing caching mechanisms445Canceling asynchronous operations446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving with setImmediate457Using external processes460Delegating the subset sum task to an external process461Considerations of the multi-process approach473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling477Cloning and load balancing477The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling visit he total seles readule480Notes on data as the subset module480Notes on the behavior of the cluster module481Building a simple H	Local initialization check	429
Pre-initialization queues431In the wild435Asynchronous request batching and caching435What's asynchronous request batching?436Optimal asynchronous request caching437An API server without caching or batching439Batching and caching with promises441Batching requests in the total sales web server442Caching requests in the total sales web server443Notes about implementing caching mechanisms445Canceling asynchronous operations4447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving with setImmediate457Interleaving the subset sum problem458Quality generators460Delegating the subset sum task to an external process461Considerations for the multi-process approach469Running CPU-bound tasks in a worker thread469Running CPU-bound tasks in production472Summary473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling477Cloning and load balancing479The three dimensions of scalability477Cloning and load balancing479The cluster module481Building a simple HTTP server482Scaling via thit betoker module484Resiliency and availability with the cluster module	Delayed startup	430
In the wild435Asynchronous request batching and caching435What's asynchronous request batching?436Optimal asynchronous request caching437An API server without caching or batching439Batching and caching with promises441Batching requests in the total sales web server442Caching requests in the total sales web server443Notes about implementing caching mechanisms445Canceling asynchronous operations445A basic recipe for creating cancelable functions446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving the stubset sum algorithm457Considerations for the multi-process approach460Delegating the subset sum task to an external process460Delegating the subset sum task in a worker thread468Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to applications scaling476Scaling Node_js applications477The three dimensions of scalability477The cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486An int	Pre-initialization queues	431
Asynchronous request batching and caching435What's asynchronous request batching?436Optimal asynchronous request caching437An API server without caching or batching439Batching and caching with promises441Batching requests in the total sales web server442Caching requests in the total sales web server443Notes about implementing caching mechanisms445Canceling asynchronous operations4445A basic recipe for creating cancelable functions446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving with setImmediate457Considerations on the interleaving approach460Delegating the subset sum task to an external process461Considerations for the multi-process approach468Running CPU-bound tasks in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module482Resiliency and availability with the cluster module486Running CPU-bound tasks in production477 </td <td>In the wild</td> <td>435</td>	In the wild	435
What's asynchronous request batching?436Optimal asynchronous request caching437An API server without caching or batching439Batching and caching with promises441Batching requests in the total sales web server442Caching requests in the total sales web server443Notes about implementing caching mechanisms445Canceling asynchronous operations446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving with setImmediate457Considerations of the multi-process approach460Delegating the subset sum task to an external process461Considerations of the multi-process approach462Using worker threads468Running CPU-bound tasks in production477Summary473Exercises473Chapter 12: Scalability and Architectural Patterns476Scaling Node js applications477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module481Building a simple HTTP server482Scaling with the cluster module486Running CPU-bound tasks in production477Considerations of scalability477	Asynchronous request batching and caching	435
Optimal asynchronous request caching437An API server without caching or batching439Batching and caching with promises441Batching requests in the total sales web server442Caching requests in the total sales web server443Notes about implementing caching mechanisms445 <b>Canceling asynchronous operations</b> 445A basic recipe for creating cancelable functions446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving the steps of the subset sum algorithm457Considerations on the interleaving approach458Using external processes460Delegating the subset sum task to an external process461Considerations on the interleaving approach459Using worker threads468Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to applications caling477The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module484Resiliency and availability with the cluster module484Areacodurutime restert485<	What's asynchronous request batching?	436
An API server without caching or batching 439   Batching and caching with promises 441   Batching requests in the total sales web server 442   Caching requests in the total sales web server 443   Notes about implementing caching mechanisms 445   Canceling asynchronous operations 445   A basic recipe for creating cancelable functions 446   Wrapping asynchronous invocations 447   Cancelable async functions with generators 449   Running CPU-bound tasks 453   Solving the subset sum problem 453   Interleaving with setImmediate 457   Interleaving the steps of the subset sum algorithm 459   Using external processes 460   Delegating the subset sum task to an external process 461   Using worker threads 468   Running CPU-bound tasks in a worker thread 469   Running the subset sum task in a worker thread 469   Running CPU-bound tasks in production 472   Saing Worker threads 468   Running CPU-bound tasks in production 473   Exercises 473   Chapter 12: Scalability and Architectur	Optimal asynchronous request caching	437
Batching and caching with promises441Batching requests in the total sales web server442Caching requests in the total sales web server443Notes about implementing caching mechanisms445 <b>Canceling asynchronous operations</b> 445A basic recipe for creating cancelable functions446Wrapping asynchronous invocations447Cancelable async functions with generators449 <b>Running CPU-bound tasks</b> 453Solving the subset sum problem453Interleaving with setImmediate457Interleaving the steps of the subset sum algorithm457Considerations for the multi-process approach468Running CPU-bound tasks in a worker thread469Using external processes460Delegating the subset sum task in a worker thread469Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Resiliency and availability with the cluster module482Arear chammer areat481Running the subset sum task in a worker thread469Running the subset sum task in a worker thread469Running the subset sum task in a worker thread473Exercises473 <td>An API server without caching or batching</td> <td>439</td>	An API server without caching or batching	439
Batching requests in the total sales web server 442   Caching requests in the total sales web server 443   Notes about implementing caching mechanisms 445 <b>Canceling asynchronous operations</b> 445   A basic recipe for creating cancelable functions 447   Cancelable async functions with generators 449 <b>Running CPU-bound tasks</b> 453   Solving the subset sum problem 453   Interleaving with setImmediate 457   Interleaving the steps of the subset sum algorithm 457   Considerations for the multi-process approach 460   Delegating the subset sum task to an external process 460   Delegating the subset sum task in a worker thread 468   Running CPU-bound tasks in production 472   Considerations for the multi-process approach 467   Using worker threads 468   Running CPU-bound tasks in production 472   Summary 473   Exercises 473   Chapter 12: Scalability and Architectural Patterns 475   An introduction to applications 477   The three dimensions of scalability 477   Choning and l	Batching and caching with promises	441
Caching requests in the total sales web server443Notes about implementing caching mechanisms445Canceling asynchronous operations445A basic recipe for creating cancelable functions446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving the steps of the subset sum algorithm457Considerations on the interleaving approach460Delegating the subset sum task to an external process460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns476Scaling Node is applications477The three dimensions of scalability477Cloning and load balancing479The cluster module481Building a simple HTTP server482Scaling with the cluster module481Resiliency and availability with the cluster module481	Batching requests in the total sales web server	442
Notes about implementing caching mechanisms445Canceling asynchronous operations445A basic recipe for creating cancelable functions446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving the steps of the subset sum algorithm457Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running CPU-bound tasks in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zerno-downtime astern486Zerno-downtime astern486Zerno-downtime astern486Zerno-downtime astern486Zerno-downtime astern486Running CPU-bound tasks in production479The three dimensions of scalability477<	Caching requests in the total sales web server	443
Canceling asynchronous operations445A basic recipe for creating cancelable functions446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving the steps of the subset sum algorithm457Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to applications scaling477The three dimensions of scalability477Cloning and load balancing479The cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zern-downtime astern486Area of the cluster module486Area o	Notes about implementing caching mechanisms	445
A basic recipe for creating cancelable functions446Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving the subset sum algorithm457Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running CPU-bound tasks in a worker thread469Running CPU-bound tasks in production472Summary473Exercises475An introduction to applications caling476Scaling Node is applications477The three dimensions of scalability477The cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module484Zero-downtime restart488	Canceling asynchronous operations	445
Wrapping asynchronous invocations447Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving with setImmediate457Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module484Zero-downtime restart488	A basic recipe for creating cancelable functions	446
Cancelable async functions with generators449Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving the steps of the subset sum algorithm457Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to applications scaling477Cloning and load balancing479The cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module484	Wrapping asynchronous invocations	447
Running CPU-bound tasks453Solving the subset sum problem453Interleaving with setImmediate457Interleaving the steps of the subset sum algorithm457Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to applications477The three dimensions of scalability477Cloning and load balancing479The cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module484Zerodowning rastart486	Cancelable async functions with generators	449
Solving the subset sum problem453Interleaving with setImmediate457Interleaving the steps of the subset sum algorithm457Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to applications477The three dimensions of scalability477Cloning and load balancing479The cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zerodowning restart486	Running CPU-bound tasks	453
Interleaving with setImmediate457Interleaving the steps of the subset sum algorithm457Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns476Scaling Node is applications477The three dimensions of scalability477Cloning and load balancing479The cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module484	Solving the subset sum problem	453
Interleaving the steps of the subset sum algorithm457Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns476Scaling Node.js application scaling477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486	Interleaving with setImmediate	457
Considerations on the interleaving approach459Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling477Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zaro-downtime restart486Zaro-downtime restart486	Interleaving the steps of the subset sum algorithm	457
Using external processes460Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling477Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module484Zero-downtime restart488	Considerations on the interleaving approach	459
Delegating the subset sum task to an external process461Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486	Using external processes	460
Considerations for the multi-process approach467Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart486	Delegating the subset sum task to an external process	461
Using worker threads468Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module484Zero-downtime restart488	Considerations for the multi-process approach	467
Running the subset sum task in a worker thread469Running CPU-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486	Using worker threads	468
Running CPO-bound tasks in production472Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	Running the subset sum task in a worker thread	469
Summary473Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486	Running CPU-bound tasks in production	472
Exercises473Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486	Summary	4/3
Chapter 12: Scalability and Architectural Patterns475An introduction to application scaling476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	Exercises	473
An introduction to application scaling476Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	Chapter 12: Scalability and Architectural Patterns	475
Scaling Node.js applications477The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	An introduction to application scaling	476
The three dimensions of scalability477Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	Scaling Node.js applications	477
Cloning and load balancing479The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	The three dimensions of scalability	477
The cluster module480Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	Cloning and load balancing	479
Notes on the behavior of the cluster module481Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	The cluster module	480
Building a simple HTTP server482Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	Notes on the behavior of the cluster module	481
Scaling with the cluster module484Resiliency and availability with the cluster module486Zero-downtime restart488	Building a simple HTTP server	482
Resiliency and availability with the cluster module 486 Zero-downtime restart 488	Scaling with the cluster module	484
	Resiliency and availability with the Cluster module Zero-downtime restart	486 288

Dealing with stateful communications	490
Sharing the state across multiple instances	491
Slicky load balancing	492
Load balancing with Nginy	494
Dynamic borizontal scaling	501
Lising a service registry	501
Implementing a dynamic load balancer with http-proxy and Consul	503
Peer-to-peer load balancing	510
Implementing an HTTP client that can balance requests across multiple servers	511
Scaling applications using containers	513
What is a container?	513
Creating and running a container with Docker	514
What is Kubernetes?	517
Deploying and scaling an application on Kubernetes	519
Decomposing complex applications	523
Monolithic architecture	524
The microservice architecture	526
An example of a microservice architecture	526
Microservices – advantages and disadvantages	528
Integration patterns in a microservice architecture	530
API orchestration	532
Integration with a message broker	536
Summary	538
Summary Exercises	538 539
Summary Exercises Chapter 13: Messaging and Integration Patterns	538 539 541
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system	538 539 <u>541</u> 542
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns	538 539 541 542 542
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 544
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 544 544 545
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 544 544 545 545
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Asynchronous messaging, queues, and streams	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 544 545 545 545
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 544 545 545 545 545 547
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging Publish/Subscribe pattern	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 544 545 545 545 545 547 <b>549</b>
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging Publish/Subscribe pattern Building a minimalist real-time chat application	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 544 545 545 545 545 545 547 <b>549</b> 550
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging Publish/Subscribe pattern Building a minimalist real-time chat application Implementing the server side	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 545 545 545 545 545 545 545 <b>545</b> 547 <b>549</b> 550 550
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging Publish/Subscribe pattern Building a minimalist real-time chat application Implementing the server side Implementing the server side Implementing the server side	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 545 545 545 545 545 547 <b>549</b> 550 550 550
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging Publish/Subscribe pattern Building a minimalist real-time chat application Implementing the server side Implementing the client side Running and scaling the chat application	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 544 545 545 545 545 547 <b>549</b> 550 550 551 553
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging Publish/Subscribe pattern Building a minimalist real-time chat application Implementing the server side Implementing the client side Running and scaling the chat application Using Redis as a simple message broker Dear to poer Dublish/Subacribe with ZarrMO	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 544 545 545 545 545 547 <b>549</b> 550 551 553 554 554
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging Publish/Subscribe pattern Building a minimalist real-time chat application Implementing the server side Implementing the client side Running and scaling the chat application Using Redis as a simple message broker Peer-to-peer Publish/Subscribe with ZeroMQ	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 545 545 545 545 545 547 <b>549</b> 550 550 551 553 554 557
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging Publish/Subscribe pattern Building a minimalist real-time chat application Implementing the server side Implementing the client side Running and scaling the chat application Using Redis as a simple message broker Peer-to-peer Publish/Subscribe with ZeroMQ Introducing ZeroMQ Designing a peer-to-peer architecture for the chat server	538 539 541 542 542 544 545 545 545 545 545 545 545
Summary Exercises Chapter 13: Messaging and Integration Patterns Fundamentals of a messaging system One way versus request/reply patterns Message types Command Messages Event Messages Document Messages Document Messages Asynchronous messaging, queues, and streams Peer-to-peer or broker-based messaging Publish/Subscribe pattern Building a minimalist real-time chat application Implementing the server side Implementing the server side Implementing the client side Running and scaling the chat application Using Redis as a simple message broker Peer-to-peer Publish/Subscribe with ZeroMQ Introducing ZeroMQ Designing a peer-to-peer architecture for the chat server Using the ZeroMQ PUB/SUB sockets	<b>538</b> <b>539</b> <b>541</b> <b>542</b> 542 544 545 545 545 545 545 547 <b>549</b> 550 550 550 551 553 554 553 554 557 557 558 559

Reliable message delivery with queues	562
Introducing AMQP	564
Durable subscribers with AMQP and RabbitMQ	566
Reliable messaging with streams	571
Characteristics of a streaming platform	571
Streams versus message queues	573
Implementing the chat application using Redis Streams	573
Task distribution patterns	577
The ZeroMQ Fanout/Fanin pattern	579
PUSH/PULL sockets	580
Building a distributed hashsum cracker with ZeroMQ	580
Pipelines and competing consumers in AMQP	587
Point-to-point communications and competing consumers	588
Implementing the hashsum cracker using AMQP	588
Distributing tasks with Redis Streams	592
Redis consumer groups	593
Implementing the hashsum cracker using Redis Streams	594
Request/Reply patterns	598
Correlation Identifier	598
Implementing a request/reply abstraction using correlation identifiers	599
Return address	605
Implementing the Return Address pattern in AMQP	605
Summary	611
Exercises	612
Other Books You May Enjoy	615
Index	619