

Contents

Foreword	25
Preface	27

1 Basic Principles 31

1.1 The Story of Node.js	33
1.1.1 Origins	33
1.1.2 Birth of Node.js	34
1.1.3 Breakthrough of Node.js	34
1.1.4 Node.js Conquers Windows	35
1.1.5 io.js: The Fork of Node.js	36
1.1.6 Node.js Reunited	36
1.1.7 Deno: A New Star in the JavaScript Sky	36
1.1.8 OpenJS Foundation	37
1.2 Organization of Node.js	37
1.2.1 Technical Steering Committee	37
1.2.2 Collaborators	37
1.2.3 Community Committee	38
1.2.4 Work Groups	38
1.2.5 OpenJS Foundation	38
1.3 Versioning of Node.js	38
1.3.1 Long-Term Support Releases	39
1.4 Benefits of Node.js	40
1.5 Areas of Use for Node.js	40
1.6 The Core: V8 Engine	41
1.6.1 Memory Model	42
1.6.2 Accessing Properties	43
1.6.3 Machine Code Generation	45
1.6.4 Garbage Collection	46
1.7 Libraries around the Engine	47
1.7.1 Event Loop	48
1.7.2 Input and Output	50
1.7.3 libuv	50
1.7.4 Domain Name System	51
1.7.5 Crypto	52
1.7.6 Zlib	52

1.7.7	HTTP Parser	52
1.8	Summary	53

2 Installation 55

2.1	Installing Packages	56
2.1.1	Linux	57
2.1.2	Windows	60
2.1.3	macOS	63
2.2	Compiling and Installing	68
2.3	Node Version Manager	71
2.4	Node and Docker	71
2.5	Summary	72

3 Developing Your First Application 73

3.1	Interactive Mode	73
3.1.1	General Use	74
3.1.2	Other REPL Commands	75
3.1.3	Saving and Loading in the REPL	76
3.1.4	Context of the REPL	77
3.1.5	REPL History	77
3.1.6	REPL Mode	78
3.1.7	Searching in the REPL	78
3.1.8	Asynchronous Operations in the REPL	79
3.2	The First Application	79
3.2.1	Web Server in Node.js	80
3.2.2	Extending the Web Server	83
3.2.3	Creating an HTML Response	85
3.2.4	Generating Dynamic Responses	86
3.3	Debugging Node.js Applications	88
3.3.1	Navigating in the Debugger	90
3.3.2	Information in the Debugger	91
3.3.3	Breakpoints	93
3.3.4	Debugging with Chrome Developer Tools	96
3.3.5	Debugging in the Development Environment	97

3.4	nodemon Development Tool	98
3.5	Summary	99

4 Node.js Modules 101

4.1	Modular Structure	101
4.2	Core Modules	103
4.2.1	Stability	104
4.2.2	List of Core Modules	105
4.2.3	Loading Core Modules	108
4.2.4	Global Objects	111
4.3	JavaScript Module Systems	121
4.3.1	CommonJS	121
4.3.2	ECMAScript Modules	122
4.4	Creating and Using Your Own Modules	124
4.4.1	Modules in Node.js: CommonJS	125
4.4.2	Custom Node.js Modules	126
4.4.3	Modules in Node.js: ECMAScript	127
4.4.4	Exporting Different Types of Data	129
4.4.5	The modules Module	130
4.4.6	Module Loader	131
4.5	Summary	135

5 HTTP 137

5.1	Web Server	137
5.1.1	Server Object	137
5.1.2	Server Events	142
5.1.3	Request Object	145
5.1.4	Handling the Request Body (Update)	152
5.1.5	Delivering Static Content	157
5.1.6	File Upload	159
5.1.7	Fine-Tuning the Frontend	163
5.2	Node.js as HTTP Client	164
5.2.1	Requests with the http Module	164
5.2.2	The request Package	165
5.2.3	HTML Parser	167

5.3	Secure Communication with HTTPS	168
5.3.1	Creating Certificates	168
5.3.2	Using HTTPS in the Web Server	169
5.4	HTTP/2	170
5.4.1	HTTP/2 Server	170
5.4.2	HTTP/2 Client	173
5.5	Summary	175

6 Express 177

6.1	Structure	177
6.2	Installation	178
6.3	Basic Principles	179
6.3.1	Request	180
6.3.2	Response	180
6.4	Setup	181
6.4.1	Structure of an Application	182
6.5	Movie Database	185
6.5.1	Routing	186
6.5.2	Controller	189
6.5.3	Model	190
6.5.4	View	192
6.6	Middleware	193
6.6.1	Custom Middleware	194
6.6.2	Morgan: Logging Middleware for Express	195
6.6.3	Delivering Static Content	197
6.7	Extended Routing: Deleting Data Records	199
6.8	Creating and Editing Data Records: Body Parser	201
6.8.1	Handling Form Input: Body Parser	205
6.9	Express 5	208
6.10	HTTPS and HTTP/2	209
6.10.1	HTTPS	209
6.10.2	HTTP/2	210
6.11	Summary	212

7 Template Engines 213

7.1	Custom Template Engine	214
7.2	Template Engines in Practice: Pug	215
7.2.1	Installation	215
7.2.2	Pug and Express: Integration	216
7.2.3	Variables in Pug	219
7.2.4	Specific Features of Pug	221
7.2.5	Conditions and Loops	222
7.2.6	Extends and Includes	223
7.2.7	Mixins	226
7.2.8	Using Pug without Express	228
7.2.9	Compiling	228
7.3	Handlebars	229
7.3.1	Installation	230
7.3.2	Integration with Express	230
7.3.3	Conditions and Loops	232
7.3.4	Partials	234
7.3.5	Custom Helpers	236
7.3.6	Handlebars without Express	238
7.4	Summary	239

8 Connecting Databases 241

8.1	Node.js and Relational Databases	242
8.1.1	MySQL	242
8.1.2	SQLite	251
8.1.3	Object-Relational Mapping	257
8.2	Node.js and Nonrelational Databases	260
8.2.1	Redis	260
8.2.2	MongoDB	265
8.3	Summary	272

9	Authentication and Session Handling	273
9.1	Passport	273
9.2	Setup and Configuration	274
9.2.1	Installation	274
9.2.2	Configuration	274
9.2.3	Strategy Configuration	275
9.3	Logging In to the Application	277
9.3.1	Login Form	277
9.3.2	Securing Resources	280
9.3.3	Logging Out	281
9.3.4	Connecting to the Database	282
9.4	Accessing Resources	285
9.4.1	Access Restriction	285
9.4.2	Submitting Ratings	289
9.5	Summary	294
10	REST Server	295
10.1	Introduction to REST and Usage in Web Applications	295
10.2	Accessing the Application	296
10.2.1	Postman	296
10.2.2	cURL	297
10.3	Adaptations to the Application Structure	297
10.4	Read Requests	298
10.4.1	Reading All Data Records of a Resource	298
10.4.2	Accessing a Data Record	301
10.4.3	Error Handling	302
10.4.4	Sorting the List	304
10.4.5	Controlling the Output Format	307
10.5	Write Requests	309
10.5.1	POST: Creating New Data Records	309
10.5.2	PUT: Modifying Existing Data Records	312
10.5.3	DELETE: Deleting Data Records	314
10.6	Authentication via JWTs	316
10.6.1	Login	317
10.6.2	Safeguarding Resources	319

10.6.3	Accessing User Information in the Token	321
10.7	OpenAPI Specification: Documentation with Swagger	324
10.8	Validation	329
10.8.1	Installation and First Validation	330
10.8.2	Checking Requests with a Validation Schema	332
10.9	Summary	335

11 GraphQL 337

11.1	GraphQL Libraries	338
11.2	Integration with Express	339
11.3	GraphiQL	341
11.4	Reading Data via the Interface	342
11.4.1	Parameterizing Queries	345
11.5	Write Accesses to the GraphQL Interface	347
11.5.1	Creating New Data Records	347
11.5.2	Updating and Deleting Data Records	350
11.6	Authentication for the GraphQL Interface	353
11.7	Summary	355

12 Real-Time Web Applications 357

12.1	The Sample Application	358
12.2	Setup	358
12.3	WebSockets	364
12.3.1	The Server Side	366
12.3.2	The Client Side	367
12.3.3	User List	370
12.3.4	Logout	373
12.4	Socket.IO	377
12.4.1	Installation and Integration	378
12.4.2	Socket.IO API	379
12.5	Summary	383

13 Type-Safe Applications in Node.js 385

13.1	Type Systems for Node.js	386
13.1.1	Flow	386
13.1.2	TypeScript	390
13.2	Tools and Configuration	392
13.2.1	Configuring the TypeScript Compiler	393
13.2.2	Integration into the Development Environment	394
13.2.3	ESLint	395
13.2.4	ts-node	396
13.3	Basic Principles	398
13.3.1	Data Types	398
13.3.2	Functions	400
13.3.3	Modules	402
13.4	Classes	403
13.4.1	Methods	404
13.4.2	Access Modifiers	405
13.4.3	Inheritance	405
13.5	Interfaces	406
13.6	Type Aliases in TypeScript	408
13.7	Generics	409
13.8	TypeScript in Use in a Node.js Application	410
13.8.1	Type Definitions	410
13.8.2	Creating Custom Type Definitions	410
13.8.3	Sample Express Application	411
13.9	Summary	412

14 Web Applications with Nest 413

14.1	Installation and Getting Started with Nest	414
14.2	Nest Command-Line Interface	416
14.2.1	Commands for Operating and Running the Application	416
14.2.2	Creating Structures in the Application	418
14.3	Structure of the Application	419
14.3.1	Root Directory with the Configuration Files	419
14.3.2	src Directory: Core of the Application	420
14.3.3	Other Directories of the Application	420

14.4	Modules: Logical Units in the Source Code	421
14.4.1	Creating Modules	422
14.4.2	Module Decorator	423
14.5	Controllers: Endpoints of an Application	423
14.5.1	Creating a Controller	424
14.5.2	Implementing a Controller	424
14.5.3	Integrating and Checking the Controller	426
14.6	Providers: Business Logic of the Application	428
14.6.1	Creating and Including a Service	428
14.6.2	Implementing the Service	429
14.6.3	Integrating the Service via Nest's Dependency Injection	431
14.7	Accessing Databases	432
14.7.1	Setup and Installation	432
14.7.2	Accessing the Database	435
14.8	Documenting the Endpoints with OpenAPI	439
14.9	Authentication	442
14.9.1	Setup	442
14.9.2	Authentication Service	443
14.9.3	Login Controller: Endpoint for User Login	445
14.9.4	Protecting Routes	446
14.10	Outlook: Testing in Nest	449
14.11	Summary	451

15 Node on the Command Line 453

15.1	Basic Principles	453
15.1.1	Structure	454
15.1.2	Executability	455
15.2	Structure of a Command-Line Application	456
15.2.1	File and Directory Structure	456
15.2.2	Package Definition	456
15.2.3	Math Trainer Application	457
15.3	Accessing Input and Output	461
15.3.1	Output	461
15.3.2	Input	462
15.3.3	User Interaction with the readline Module	463
15.3.4	Options and Arguments	467

15.4	Tools	469
15.4.1	Commander	469
15.4.2	Chalk	471
15.4.3	node-emoji	473
15.5	Signals	476
15.6	Exit Codes	478
15.7	Summary	479

16 Asynchronous Programming 481

16.1	Basic Principles of Asynchronous Programming	481
16.1.1	The <code>child_process</code> Module	485
16.2	Running External Commands Asynchronously	486
16.2.1	The <code>exec</code> Method	487
16.2.2	The <code>spawn</code> Method	489
16.3	Creating Node.js Child Processes with <code>fork</code> Method	492
16.4	The <code>cluster</code> Module	496
16.4.1	Main Process	497
16.4.2	Worker Processes	501
16.5	Worker Threads	504
16.5.1	Shared Memory in the <code>worker_threads</code> Module	505
16.6	Promises in Node.js	507
16.6.1	Using <code>util.promisify</code> to Use Promises Where None Actually Exist	510
16.6.2	Concatenating Promises	511
16.6.3	Multiple Parallel Operations with <code>Promise.all</code>	512
16.6.4	Fastest Asynchronous Operation with <code>Promise.race</code>	513
16.6.5	Overview of the Promise Functions	514
16.7	Async Functions	514
16.7.1	Top-Level <code>Await</code>	516
16.8	Summary	517

17 RxJS 519

17.1	Basic Principles	520
17.1.1	Installation and Integration	521

17.1.2	Observable	521
17.1.3	Observer	522
17.1.4	Operator	523
17.1.5	Example of RxJS in Node.js	523
17.2	Operators	525
17.2.1	Creation Operators	527
17.2.2	Transformation Operators	529
17.2.3	Filtering Operators	532
17.2.4	Join Operators	534
17.2.5	Error Handling Operators	535
17.2.6	Utility Operators	537
17.2.7	Conditional Operators	538
17.2.8	Connection Operators	539
17.2.9	Conversion Operators	540
17.3	Subjects	540
17.4	Schedulers	542
17.5	Summary	543

18 Streams 545

18.1	Introduction	545
18.1.1	What Is a Stream?	545
18.1.2	Stream Usages	546
18.1.3	Available Streams	546
18.1.4	Stream Versions in Node.js	547
18.1.5	Streams Are EventEmitters	548
18.2	Readable Streams	548
18.2.1	Creating a Readable Stream	548
18.2.2	Readable Stream Interface	550
18.2.3	Events of a Readable Stream	550
18.2.4	Error Handling in Readable Streams	552
18.2.5	Methods	553
18.2.6	Piping	553
18.2.7	Readable Stream Modes	554
18.2.8	Switching to Flowing Mode	554
18.2.9	Switching to the Paused Mode	555
18.2.10	Custom Readable Streams	555
18.2.11	Example of a Readable Stream	556
18.2.12	Readable Shortcut	558

18.3	Writable Streams	559
18.3.1	Creating a Writable Stream	560
18.3.2	Events	560
18.3.3	Error Handling in Writable Streams	562
18.3.4	Methods	562
18.3.5	Buffering Write Operations	563
18.3.6	Flow Control	564
18.3.7	Custom Writable Streams	565
18.3.8	Writable Shortcut	566
18.4	Duplex Streams	566
18.4.1	Duplex Streams in Use	566
18.4.2	Custom Duplex Streams	567
18.4.3	Duplex Shortcut	567
18.5	Transform Streams	568
18.5.1	Custom Transform Streams	568
18.5.2	Transform Shortcut	569
18.6	Gulp	570
18.6.1	Installation	570
18.6.2	Example of a Build Process with Gulp	571
18.7	Summary	572

19 Working with Files 573

19.1	Synchronous and Asynchronous Functions	573
19.2	Existence of Files	575
19.3	Reading Files	576
19.3.1	Promise-Based API	581
19.4	Error Handling	582
19.5	Writing to Files	582
19.6	Directory Operations	586
19.7	Advanced Operations	589
19.7.1	The watch Method	591
19.7.2	Access Permissions	592
19.8	Summary	594

20 Socket Server 595

20.1 Unix Sockets	596
20.1.1 Accessing the Socket	598
20.1.2 Bidirectional Communication	600
20.2 Windows Pipes	602
20.3 TCP Sockets	603
20.3.1 Data Transfer	605
20.3.2 File Transfer	606
20.3.3 Flow Control	607
20.3.4 Duplex	609
20.3.5 Pipe	609
20.4 UDP Sockets	610
20.4.1 Basic Principles of a UDP Server	611
20.4.2 Example Illustrating the UDP Server	612
20.5 Summary	614

21 Package Manager 615

21.1 Most Common Operations	616
21.1.1 Searching Packages	616
21.1.2 Installing Packages	617
21.1.3 Viewing Installed Packages	622
21.1.4 Using Packages	623
21.1.5 Updating Packages	624
21.1.6 Removing Packages	625
21.1.7 Overview of the Most Important Commands	626
21.2 Advanced Operations	627
21.2.1 Structure of a Module	627
21.2.2 Creating Custom Packages	630
21.2.3 Node Package Manager Scripts	632
21.3 Tools for Node Package Manager	634
21.3.1 Node License Finder	634
21.3.2 Verdaccio	635
21.3.3 npm-check-updates	636
21.3.4 npx	637
21.4 Yarn	637
21.5 Summary	638

22 Quality Assurance 641

22.1	Style Guides	642
22.1.1	Airbnb Style Guide	642
22.2	Linters	643
22.2.1	ESLint	644
22.3	Prettier	648
22.3.1	Installation	649
22.3.2	Execution	649
22.4	Programming Mistake Detector: Copy/Paste Detector	649
22.4.1	Installation	650
22.4.2	Execution	651
22.5	Husky	652
22.6	Summary	653

23 Testing 655

23.1	Unit Testing	655
23.1.1	Directory Structure	656
23.1.2	Unit Tests and Node.js	656
23.1.3	Arrange, Act, Assert	657
23.2	Assertion Testing	658
23.2.1	Exceptions	661
23.2.2	Testing Promises	662
23.3	Jasmine	663
23.3.1	Installation	664
23.3.2	Configuration	664
23.3.3	Tests in Jasmine	665
23.3.4	Assertions	667
23.3.5	Spies	670
23.3.6	beforeEach and afterEach	671
23.4	Jest	671
23.4.1	Installation	671
23.4.2	First Test	672
23.5	Practical Example of Unit Tests with Jest	674
23.5.1	The Test	675
23.5.2	Implementation	676

23.5.3	Triangulation: Second Test	677
23.5.4	Optimizing the Implementation	678
23.6	Dealing with Dependencies: Mocking	679
23.7	Summary	681

24 Security 683

24.1	Filter Input and Escape Output	684
24.1.1	Filter Input	684
24.1.2	Blacklisting and Whitelisting	684
24.1.3	Escape Output	685
24.2	Protecting the Server	686
24.2.1	User Permissions	686
24.2.2	Problems Caused by the Single-Threaded Approach	688
24.2.3	Denial-of-Service Attacks	690
24.2.4	Regular Expressions	692
24.2.5	HTTP Header	693
24.2.6	Error Messages	695
24.2.7	SQL Injections	695
24.2.8	eval	699
24.2.9	Method Invocation	700
24.2.10	Overwriting Built-Ins	702
24.3	Node Package Manager Security	704
24.3.1	Permissions	704
24.3.2	Node Security Platform	705
24.3.3	Quality Aspect	705
24.3.4	Node Package Manager Scripts	706
24.4	Client Protection	707
24.4.1	Cross-Site Scripting	707
24.4.2	Cross-Site Request Forgery	709
24.5	Summary	711

25 Scalability and Deployment 713

25.1	Deployment	713
25.1.1	Simple Deployment	713
25.1.2	File Synchronization via rsync	715

25.1.3	Application as a Service	716
25.1.4	node_modules in Deployment	718
25.1.5	Installing Applications Using Node Package Manager	718
25.1.6	Installing Packages Locally	720
25.2	Tool Support	720
25.2.1	Grunt	721
25.2.2	Gulp	721
25.2.3	Node Package Manager	721
25.3	Scaling	721
25.3.1	Child Processes	722
25.3.2	Load Balancer	726
25.3.3	Node in the Cloud	728
25.4	pm2: Process Management	730
25.5	Docker	730
25.5.1	Dockerfile	731
25.5.2	Starting the Container	732
25.6	Summary	732

26 Performance 733

26.1	You Aren't Gonna Need It	733
26.2	CPU	734
26.2.1	CPU-Blocking Operations	734
26.2.2	Measuring the CPU Load	735
26.2.3	CPU Profiling with Chrome DevTools	736
26.2.4	Alternatives to the Profiler: console.time	738
26.2.5	Alternatives to the Profiler: Performance-Hooks Interface	739
26.3	Memory	741
26.3.1	Memory Leaks	742
26.3.2	Memory Analysis in DevTools	743
26.3.3	Node.js Memory Statistics	745
26.4	Network	747
26.5	Summary	751

27 Microservices with Node.js 753

27.1 Basic Principles	753
27.1.1 Monolithic Architecture	753
27.1.2 Microservice Architecture	755
27.2 Architecture	756
27.2.1 Communication between Individual Services	756
27.3 Infrastructure	758
27.3.1 Docker Compose	759
27.4 Asynchronous Microservice with RabbitMQ	759
27.4.1 Installation and Setup	760
27.4.2 Connecting to the RabbitMQ Server	762
27.4.3 Handling Incoming Messages	763
27.4.4 Database Connection	764
27.4.5 Docker Setup	765
27.5 API Gateway	768
27.5.1 Connecting the User Service	768
27.5.2 Asynchronous Communication with the User Service	770
27.5.3 Docker Setup of the API Gateway	774
27.5.4 Authentication	776
27.6 Synchronous Microservice with Express	780
27.6.1 Setup	781
27.6.2 Controller	782
27.6.3 Model Implementation	782
27.6.4 Docker Setup	784
27.6.5 Integration into the API Gateway	786
27.7 Summary	789

28 Deno 791

28.1 The Ten Things Ryan Dahl Regrets about Node.js	791
28.1.1 Promises	791
28.1.2 Security	792
28.1.3 The Generate Your Projects Build System	792
28.1.4 Package.json	792
28.1.5 Node_modules	792
28.1.6 Optional File Extension When Loading Modules	793
28.1.7 Index.js	793

28.1.8	What's Going on Now with Node.js	793
28.2	Installing Deno	793
28.2.1	Deno Command-Line Interface	794
28.3	Execution	795
28.3.1	Running a TypeScript Application	796
28.4	Handling Files	796
28.4.1	The Task: Copying a File	797
28.4.2	Processing Command-Line Options	797
28.4.3	Reading Files	798
28.4.4	Permissions in Deno	800
28.4.5	readTextFile Function	801
28.4.6	Writing Files with Deno	801
28.5	Web Server with Deno	803
28.6	Module System	804
28.6.1	Loading External Modules into Deno	806
28.6.2	deno.land/x	807
28.6.3	Using Node Package Manager Packages	807
28.7	Summary	809
The Author		811
Index		813