Inhalt

Vorwort ... **XI**
... und ihre Motivation ... XIII
Das Zielpublikum ... XIII
Das Buch ... XIV
Die Welt von JavaScript ... XV

1 Hello, Node.js ... **1**
1.1 Einführung in Node.js ... 1
1.2 Installation ... 8
1.2.1 Windows ... 8
1.2.2 Mac OS X ... 8
1.2.3 Debian ... 9
1.2.4 Ubuntu ... 9
1.2.5 openSUSE und SLE ... 10
1.2.6 Fedora ... 11
1.2.7 RHEL und CentOS ... 11
1.3 IDEs ... 11
1.3.1 cloud9 ... 12
1.3.2 WebStorm ... 14
1.3.3 Nodeclipse ... 15
1.3.4 WebMatrix/VisualStudio ... 16
1.3.5 Atom ... 16
1.4 nvm & nodist – mit Node-Versionen jonglieren ... 17
1.4.1 *ix-Systeme ... 18
1.4.2 Windows ... 19
1.5 npm – Node Packaged Modules ... 21
1.5.1 npm install – ein Modul laden ... 22
1.5.2 Global? Lokal? ... 23
1.5.3 package.json ... 24
1.5.4 Module patchen ... 24
1.5.5 Browserify ... 28
1.6 Kein Code? ... 30

Bibliografische Informationen
http://d-nb.info/1063128455

digitalisiert durch DEUTSCHE NATIONAL BIBLIOTHEK

2 You build it ... 35
2.1 File I/O ... 36
2.1.1 Dateifunktionen in Node.js ... 36
2.1.2 Permissions ... 39
2.1.3 „watch" - Änderungen im Auge behalten ... 40
2.1.4 Erweiterungen ... 41
2.1.4.1 Modul „fs-extra" ... 42
2.1.4.2 Modul „file" ... 43
2.1.4.3 Modul „find" ... 43
2.1.4.4 Modul „properties" ... 44
2.1.4.5 Modul „token-filter" ... 45
2.2 Streams ... 46
2.2.1 Aus Streams lesen 47
2.2.1.1 Objekte und Strings ... 48
2.2.2 ... und in Streams schreiben ... 49
2.2.2.1 Streams verknüpfen ... 49
2.2.3 Eigene Streams implementieren ... 50
2.2.3.1 Ein Random-Number-Generator ... 51
2.2.3.2 Ein Daten-Lösch-Stream ... 53
2.2.3.3 Ein Verschlüsselungsserver für geheime Botschaften ... 54
2.2.4 Buffers and Strings ... 56
2.3 Daten für immer ... 57
2.3.1 Neo4j ... 57
2.3.1.1 Asynchron? ... 60
2.3.1.2 Querying Neo4j ... 61
2.3.1.3 Cypher für Abfragen ... 62
2.3.1.4 Indizes ... 64
2.3.1.5 Cypher für Batches ... 65
2.3.2 MongoDB ... 66
2.3.2.1 Wann sind Daten geschrieben? ... 67
2.3.2.2 _id ... 68
2.3.2.3 Die Mongo-API ... 68
2.4 Sichtbarkeit erzeugen - im Web ... 73
2.4.1 Middleware Framework Connect ... 73
2.4.1.1 Installation und einführendes Beispiel ... 74
2.4.1.2 Ausprägungen von Connect-Middleware-Typen ... 75
2.4.1.3 Integrierte Middleware-Komponenten ... 77
2.4.1.4 Middleware-Strukturen ... 85
2.4.2 Webentwicklung mit Express ... 90
2.4.2.1 Ready for take off: Installation und Einführungsbeispiel ... 91
2.4.2.2 Routing von HTTP-Anfragen ... 94
2.4.2.3 Views und Web-Templating ... 98
2.4.3 Express 4 ... 99

2.4.4 Jade ... 101
2.4.4.1 Einbindung in Express ... 103
2.4.4.2 Sprachelemente von Jade ... 103
2.4.5 swig ... 116
2.4.5.1 Grundeinstellungen ... 116
2.4.5.2 Einbindung in Express ... 117
2.4.5.3 Sprachelemente von swig ... 118
2.4.5.4 Filterliste ... 121
2.4.5.5 Verketten von Filtern ... 124
2.4.5.6 Die swig-API ... 124
2.4.5.7 Eigene Funktionalitäten hinzufügen ... 126
2.4.6 Sessions & Authentifizierung ... 127
2.4.6.1 Ich will Kekse und biete dafür eine Session ... 128
2.4.6.2 Authentifizierung (Authentication) ... 130
2.4.6.3 Facebook ... 133
2.4.6.4 Twitter ... 134
2.4.6.5 Google ... 135
2.5 socket.io ... 136
2.5.1 Verbindung herstellen ... 137
2.5.2 Kommunikation ... 138
2.5.3 Broadcast ... 139
2.5.4 Private Daten ... 139
2.5.5 Rückantwort und Bestätigung ... 139
2.5.6 Namespaces ... 140
2.5.7 Räume ... 141
2.5.8 Autorisierung ... 143
2.5.8.1 Globale Autorisierung ... 143
2.5.8.2 Autorierung mit Namespaceses ... 144
2.5.8.3 Benutzerdefinierte Variablen und Autorisierung ... 145
2.5.9 Sessions mit „socket.io-session“ ... 145
2.5.9.1 socket.io-bundle ... 145
2.5.9.2 socket.io-passport ... 146
2.5.10 Version 1.0 ... 147
2.6 Node.js und Webservices ... 151
2.6.1 SOAP-Services ... 151
2.6.1.1 Von und nach SOAP ... 153
2.6.2 REST-Services ... 163
2.6.2.1 Von Nomen, Verben und Routen ... 164
2.6.2.2 Ansichtssache? Verhandlungssache ... 168
2.6.2.3 Fehlermeldungen ... 170
2.6.2.4 Plug-ins ... 171
2.6.2.5 Sicherheit und Authentifizierung ... 176
2.6.3 XML-Verarbeitung ... 183
2.6.3.1 XML-Parsing ... 183

2.6.3.2 XML-Erzeugung und -Veränderung 189
2.6.3.3 Exkurs: Ein (selbst unterschriebenes) Zertifikat erstellen 191
2.7 Clustering 193
2.7.1 Methoden und Eigenschaften von cluster 197
2.7.1.1 isMaster/isWorker 197
2.7.1.2 fork/online - Event 197
2.7.1.3 exit - Event 198
2.7.1.4 workers 198
2.7.2 Der Master 198
2.7.2.1 setupMaster() 199
2.7.2.2 fork() 200
2.7.2.3 disconnect() 200
2.7.3 Der Worker 201
2.7.3.1 Die Attribute „id“ und „process“ 201
2.7.3.2 Das suicide-Attribut 201
2.7.3.3 kill() & disconnect() 201
2.8 Der Callback-Hölle entfliehen 202
2.8.1 async 203
2.8.1.1 Kontrollfluss 205
2.8.2 Q 212
2.8.2.1 then 214
2.8.2.2 fail 215
2.8.2.3 progress 215
2.9 Auf Herz und Nieren - Node.js-Anwendungen testen 216
2.9.1 Mocha 217
2.9.1.1 Asynchrone Aufrufe und Timeouts 220
2.9.1.2 Set-Up & Tear-Down 222
2.9.1.3 Only & Skip 223
2.9.1.4 Mocha im Browser 223
2.9.2 Assert & Chai 225
2.9.2.1 Assert 225
2.9.2.2 Chai 227
2.9.3 Sinon 232
2.9.3.1 Spies 234
2.9.3.2 Stubs 235
2.9.3.3 Mocks 236
2.9.3.4 Faked Timers 237
2.9.4 Jasmine 238
2.9.5 Continuous Test 239
2.9.5.1 Mocha & Jasmine im Überwachungsmodus 239
2.9.5.2 Travis-CI 240

3 ... you run it! ... 245
3.1 Eigene Module publizieren ... 245
3.1.1 Patterns & Style ... 246
3.1.1.1 package.json ... 247
3.1.1.2 Import & Export ... 248
3.1.1.3 Tests ... 249
3.1.1.4 Dokumentation ... 250
3.1.2 Ausführbare Module ... 252
3.1.3 Module mit nativen Abhängigkeiten ... 254
3.1.3.1 OS Libraries ... 255
3.1.3.2 Sourcecode Dependencies ... 256
3.1.3.3 Hands-On mit Add-On ... 257
3.1.4 It works on my machine - Dependency Hell ... 266
3.1.5 Veröffentlichung von Modulen ... 269
3.1.5.1 Einen Benutzer erzeugen ... 269
3.1.5.2 ... und das Modul publizieren ... 269
3.2 Private Repositories für npm ... 270
3.2.1 reggie ... 271
3.2.1.1 Inbetriebnahme ... 271
3.2.1.2 reggie publish ... 272
3.2.1.3 Laden von Modulen ... 272
3.2.1.4 HTTP-Abfragen ... 274
3.2.1.5 npm-Client ... 274
3.2.2 sinopia ... 275
3.3 Deployment ... 277
3.3.1 Ein eigener Server ... 278
3.3.1.1 Docker ... 278
3.3.1.2 Modul „forever" ... 280
3.3.1.3 pm2 ... 284
3.3.1.4 git-deploy ... 290
3.3.2 Cloud ... 291
3.3.2.1 PaaS-Provider ... 291
3.3.2.2 Server-Systeme ... 295
3.4 Was Node.js antreibt ... V8 Engine ... 296
3.4.1 Architektur ... 297
3.4.2 Die Performance-Tricks ... 299
3.4.2.1 „Fast Property Access" ... 300
3.4.2.2 Arrays ... 301
3.4.2.3 Kein Interpretationsspielraum ... 302
3.4.2.4 Garbage Collection ... 302
3.4.2.5 Caching Modules ... 303
3.5 Logging ... 304
3.5.1 debug ... 304
3.5.2 winston ... 307

3.5.2.1 Transportmechanismen ... 307
3.5.2.2 Logger-Instanz ... 308
3.5.2.3 Logging Levels ... 309
3.5.2.4 Strukturierte Daten loggen ... 309
3.5.2.5 Profiling ... 310
3.5.3 Bunyan ... 311
3.5.3.1 Konfiguration ... 312
3.5.3.2 Child Logger ... 313
3.5.3.3 Die „src"-Option ... 314
3.5.3.4 Streams ... 314
3.6 Debugging ... 315
3.6.1 Der Node-Debugger ... 315
3.6.2 Node-Inspector ... 318
3.7 Monitoring ... 321
3.7.1 Kommerzielle Monitoring-Services ... 323
3.7.1.1 New Relic ... 323
3.7.1.2 Nodetime ... 325
3.7.1.3 StrongOps ... 329
3.8 Alternativen zu Node.js ... 334
3.8.1 Vert.x - die polyglotte JVM-Alternative ... 335
3.8.1.1 Architektur ... 335
3.8.1.2 Hands-On ... 342
3.8.1.3 Node.js oder Vert.x? ... 347

Index ... **349**