

Inhalt

Geleitwort des Fachgutachters	19
Vorwort	21

1 Grundlagen 25

1.1 Die Geschichte von Node.js	26
1.1.1 Die Ursprünge	26
1.1.2 Die Geburt von Node.js	27
1.1.3 Der Durchbruch von Node.js	28
1.1.4 Node.js erobert Windows	29
1.1.5 io.js – der Fork von Node.js	29
1.1.6 Node.js wieder vereint	30
1.2 Vorteile von Node.js	30
1.3 Einsatzgebiete von Node.js	31
1.4 Das Herzstück – die V8-Engine	32
1.4.1 Das Speichermodell	33
1.4.2 Zugriff auf Eigenschaften	34
1.4.3 Maschinencodgenerierung	36
1.4.4 Garbage Collection	37
1.5 Bibliotheken um die Engine	39
1.5.1 Eventloop	40
1.5.2 Eingabe und Ausgabe	41
1.5.3 libuv	42
1.5.4 DNS	43
1.5.5 Crypto	44
1.5.6 Zlib	44
1.5.7 HTTP-Parser	45
1.6 Zusammenfassung	45

2 Installation 47

2.1 Installation von Paketen	48
2.1.1 Linux	49

2.1.2	Windows	52
2.1.3	OS X	57
2.2	Kompilieren und Installieren	63
2.3	Zusammenfassung	65
3	Ein erstes Beispiel	67
<hr/>		
3.1	Der interaktive Modus	67
3.1.1	Generelle Benutzung	67
3.1.2	Zusätzliche REPL-Befehle	69
3.1.3	Speichern und Laden im REPL	70
3.1.4	Kontext der REPL	71
3.1.5	REPL-Historie	71
3.1.6	REPL-Modus	72
3.2	Die erste Applikation	72
3.2.1	Ein Webserver in Node.js	73
3.2.2	Erweiterung des Webserver	76
3.2.3	Erstellen einer HTML-Antwort	78
3.2.4	Dynamische Antworten generieren	79
3.3	Zusammenfassung	81
4	Node-Module	83
<hr/>		
4.1	Modularer Ansatz	83
4.2	Stabilität	86
4.2.1	Stabilitätsindex	86
4.2.2	Verfügbare Module	87
4.3	Basis-Module	90
4.3.1	Globale Objekte	90
4.3.2	Utility	93
4.3.3	Events	95
4.3.4	OS	97
4.3.5	Path	97
4.4	Eigene Module erstellen und einbinden	98
4.4.1	Eigene Module in Node.js	98
4.4.2	Eigene Node.js-Module	99

4.4.3	Verschiedene Datentypen exportieren	101
4.4.4	Das modules-Modul	102
4.4.5	Der Modulloader	104
4.4.6	Die »require«-Funktionalität	107
4.5	Zusammenfassung	108
5	NPM	109
<hr/>		
5.1	Pakete suchen	109
5.1.1	Pakete installieren	110
5.1.2	Installierte Pakete anzeigen	114
5.1.3	Pakete verwenden	115
5.1.4	Pakete aktualisieren	116
5.1.5	Pakete entfernen	117
5.1.6	Die wichtigsten Kommandos im Überblick	118
5.1.7	Der Aufbau eines Moduls	119
5.1.8	Eigene Pakete erstellen	123
5.1.9	NPM Scripts	125
5.2	Werkzeuge für NPM	126
5.2.1	Node License Finder	126
5.2.2	Sinopia	127
5.3	Zusammenfassung	128
6	ECMAScript 6	129
<hr/>		
6.1	Ein kurzer Rückblick	129
6.1.1	Die wichtigsten ES5 Features	130
6.2	Node.js und ECMAScript 6	131
6.3	String- und Array-Erweiterungen	132
6.4	Scoping	133
6.4.1	Gültigkeitsbereiche in JavaScript	133
6.4.2	Module Scope in Node.js	134
6.4.3	Block Scoping	134
6.4.4	Konstante Werte	136
6.4.5	Function in Blocks	136

6.5	Klassen	137
6.5.1	Eine Klasse definieren	137
6.5.2	Vererbung	139
6.6	Template Strings	140
6.6.1	Standard Template Strings	140
6.6.2	Tagged Template Strings	141
6.7	Collections	142
6.7.1	Map	142
6.7.2	Set	143
6.7.3	WeakMaps und WeakSets	143
6.8	Arrow Functions	144
6.9	Generators	145
6.10	Promises	146
6.11	Symbols	148
6.12	Typed Arrays	150
6.13	Spread-Operator	150
6.14	Rest Parameters	151
6.15	Destructuring	151
6.16	Binary und Octal Literals	152
6.17	Erweiterung der Object Literals	152
6.18	»Object.assign«	153
6.19	»new.target«	153
6.20	Zusammenfassung	153

7 Node auf der Kommandozeile 155

7.1	Grundlagen	155
7.1.1	Aufbau	156
7.1.2	Ausführbarkeit	157
7.2	Der Aufbau einer Kommandozeilenapplikation	158
7.2.1	Datei und Verzeichnisstruktur	159
7.2.2	Paketdefinition	160
7.3	Zugriff auf Ein- und Ausgabe	161
7.3.1	Ausgabe	161

7.3.2	Eingabe	162
7.3.3	Optionen und Argumente	164
7.3.4	Signale und Exit Codes	166
7.4	Werkzeuge	169
7.4.1	Commander.js	169
7.4.2	Chalk	171
7.5	Zusammenfassung	172
8	HTTP	173
<hr/>		
8.1	Der Webserver	173
8.1.1	Das »Server«-Objekt	173
8.1.2	Server-Events	175
8.1.3	Das »Request«-Objekt	177
8.1.4	Das »Response«-Objekt	180
8.1.5	Ausliefern von HTML	184
8.1.6	Ausliefern von statischen Dateien	184
8.1.7	Umgang mit »GET« und »POST«	185
8.1.8	Dateiupload	186
8.2	HTTP-Client mit Node.js	187
8.2.1	Die »get«-Methode	188
8.2.2	Der »http.Agent«	188
8.2.3	Die Anfrage-Optionen	189
8.2.4	Die Klasse »ClientRequest«	191
8.2.5	Die Antwort des Servers	193
8.2.6	HTML-Parser	195
8.3	Umgang mit URLs	196
8.4	Sichere Kommunikation mit HTTPS	198
8.5	Zusammenfassung	199
9	Express.js	201
<hr/>		
9.1	Aufbau	201
9.2	Installation	203
9.3	Grundlagen	204

9.3.1	»Request«	204
9.3.2	»Response«	205
9.4	Setup	206
9.4.1	Verzeichnisstruktur	207
9.4.2	Router	210
9.4.3	Controller	211
9.4.4	Models	212
9.4.5	View	213
9.5	Routing	215
9.5.1	HTTP-Methoden	215
9.5.2	Muster in Routen	216
9.5.3	Variablen in Routen	217
9.5.4	Routing-Callbacks	218
9.5.5	»Route«-Objekt	219
9.5.6	Das »Router«-Objekt	220
9.6	Middleware	220
9.6.1	Eigene Middleware	221
9.6.2	Morgan	222
9.6.3	Serve-static	223
9.6.4	Body-Parser	224
9.7	Zusammenfassung	227

10 Template-Engines 229

10.1	Eine eigene Template-Engine	230
10.2	Template-Engines in der Praxis – Jade	231
10.2.1	Installation und grundsätzliche Verwendung	232
10.2.2	Jade und Express.js	233
10.2.3	Variablen in Jade	236
10.2.4	Die Besonderheiten von Jade	238
10.2.5	Bedingungen und Schleifen	242
10.2.6	Extends und Includes	245
10.2.7	Mixins	248
10.2.8	Compiling	250
10.3	Handlebars	251
10.3.1	Installation und erstes Beispiel	251
10.3.2	Bedingungen und Schleifen	253
10.3.3	Partials	254

10.3.4	Eigene Helper	255
10.3.5	Integration in Express.js	256
10.4	Zusammenfassung	258
11	REST Server	259
<hr/>		
11.1	»GET« – lesender Zugriff	261
11.2	»POST« – Anlegen neuer Ressourcen	265
11.3	»PUT« – Aktualisierung bestehender Daten	267
11.4	»DELETE« – Löschen vorhandener Daten	269
11.5	»Accept«-Header	271
11.6	Zusammenfassung	273
12	Single-Page-Webapplikationen mit Express.js und Angular.js	275
<hr/>		
12.1	Die Aufgabenstellung	275
12.2	Setup	276
12.2.1	Ordnerstruktur	276
12.2.2	Die Datenbank	277
12.2.3	Abhängigkeiten	277
12.2.4	Clientbibliotheken	278
12.3	Die Applikation	281
12.3.1	Liste von Datensätzen	282
12.3.2	Anlegen neuer Datensätze	289
12.3.3	Datensätze verändern	293
12.3.4	Löschen von Datensätzen	297
12.4	Zusammenfassung	299
13	Echtzeit-Webapplikationen	301
<hr/>		
13.1	Die Beispielapplikation	302
13.2	Setup	303

13.3	Websockets	308
13.3.1	Die Serverseite	309
13.3.2	Die Clientseite	312
13.3.3	Userliste	314
13.3.4	Logout	319
13.4	Socket.IO	320
13.4.1	Installation und Einbindung	321
13.4.2	Socket.IO-API	321
13.5	Zusammenfassung	325

14 Asynchrone Programmierung 327

14.1	Grundlagen asynchroner Programmierung	327
14.1.1	Das child_process-Modul	330
14.2	Externe Kommandos asynchron ausführen	332
14.2.1	Die »exec«-Methode	332
14.2.2	Die »spawn«-Methode	335
14.3	»fork«	337
14.4	Das cluster-Modul	341
14.4.1	Der Masterprozess	342
14.4.2	Die Workerprozesse	346
14.5	Promises in Node.js	349
14.5.1	Flusssteuerung mit Promises	351
14.6	Zusammenfassung	353

15 Streams 355

15.1	Einleitung	355
15.1.1	Was ist ein Stream?	355
15.1.2	Wozu verwendet man Streams?	356
15.1.3	Welche Streams gibt es?	357
15.1.4	Stream-Versionen in Node.js	357
15.1.5	Streams sind EventEmitter	358
15.2	Readable Streams	358
15.2.1	Einen Readable Stream erstellen	359

15.2.2	Die Readable-Stream-Schnittstelle	360
15.2.3	Die Events eines Readable Streams	361
15.2.4	Fehlerbehandlung in Readable Streams	362
15.2.5	Methoden	363
15.2.6	Piping	363
15.2.7	Readable-Stream-Modi	364
15.2.8	Wechsel in den Flowing Mode	364
15.2.9	Wechsel in den Paused Mode	365
15.2.10	Eigene Readable Streams	365
15.2.11	Beispiel für einen Readable Stream	366
15.2.12	Readable-Shortcut	368
15.3	Writable Streams	369
15.3.1	Einen Writable Stream erstellen	369
15.3.2	Die Writable-Stream-Schnittstelle	370
15.3.3	Events	370
15.3.4	Fehlerbehandlung in Writable Streams	372
15.3.5	Methoden	372
15.3.6	Schreiboperationen puffern	373
15.3.7	Flusssteuerung	374
15.3.8	Eigene Writable Streams	375
15.3.9	Writable-Shortcut	376
15.4	Duplex Streams	376
15.4.1	Duplex Streams im Einsatz	377
15.4.2	Eigene Duplex Streams	377
15.4.3	Duplex-Shortcut	378
15.5	Transform Streams	378
15.5.1	Eigene Transform Streams	379
15.5.2	Transform-Shortcut	380
15.6	Gulp	380
15.6.1	Installation	381
15.6.2	Beispiel für einen Build-Prozess mit Gulp	381
15.7	Zusammenfassung	382
16	Arbeiten mit Dateien	383
16.1	Synchrone und asynchrone Funktionen	383
16.2	Existenz von Dateien	384

16.3 Dateien lesen	385
16.4 Fehlerbehandlung	390
16.5 In Dateien schreiben	391
16.6 Verzeichnisoperationen	394
16.7 Weiterführende Operationen	398
16.7.1 »watch«	400
16.7.2 Zugriffsberechtigungen	401
16.8 Zusammenfassung	402

17 Socket-Server 405

17.1 UNIX-Sockets	406
17.1.1 Zugriff auf den Socket	408
17.1.2 Bidirektionale Kommunikation	410
17.2 Windows Pipes	413
17.3 TCP-Sockets	414
17.3.1 Datenübertragung	416
17.3.2 Dateiübertragung	417
17.3.3 Flusssteuerung	418
17.3.4 Duplex	420
17.3.5 Pipe	420
17.4 UDP-Sockets	421
17.4.1 Grundlagen eines UDP-Servers	422
17.4.2 Beispiel zum UDP-Server	424
17.5 Zusammenfassung	426

18 Anbindung von Datenbanken 427

18.1 Node.js und relationale Datenbanken	428
18.1.1 MySQL	428
18.1.2 SQLite	434
18.2 Node.js und nicht relationale Datenbanken	439
18.2.1 Redis	440
18.2.2 MongoDB	444

18.3	Datenbanken und Promises	449
18.4	Datenbanken und Streams	450
18.5	Zusammenfassung	451
19	Qualitätssicherung	453
19.1	Unittesting	453
19.1.1	Verzeichnisstruktur	454
19.1.2	Unittests und Node.js	455
19.1.3	Tripple-A	455
19.2	Assertion Testing	456
19.3	Jasmine	459
19.3.1	Installation	460
19.3.2	Konfiguration	460
19.3.3	Tests in Jasmine	461
19.3.4	Assertions	462
19.3.5	Spies	464
19.3.6	»beforeEach« und »afterEach«	465
19.4	nodeunit	465
19.4.1	Installation	465
19.4.2	Ein erster Test	466
19.4.3	Assertions	468
19.4.4	Gruppierung	468
19.4.5	»setUp« und »tearDown«	470
19.5	Praktisches Beispiel von Unittests mit nodeunit	470
19.5.1	Der Test	471
19.5.2	Die Implementierung	472
19.5.3	Der zweite Test	472
19.5.4	Verbesserung der Implementierung	473
19.6	Statische Codeanalyse	474
19.6.1	ESLint	475
19.6.2	PMD CPD	478
19.6.3	Plato	480
19.7	Node.js Debugger	482
19.7.1	Navigation im Debugger	483
19.7.2	Informationen im Debugger	484
19.7.3	Breakpoints	486

19.8 Node Inspector	487
19.8.1 Installation	488
19.8.2 Features	488
19.9 Debugging in der Entwicklungsumgebung	489
19.10 Zusammenfassung	489
20 Sicherheitsaspekte	491
<hr/>	
20.1 Filter Input und Escape Output	492
20.1.1 Filter Input	492
20.1.2 Black- und Whitelisting	492
20.1.3 Escape Output	493
20.2 Absicherung des Servers	494
20.2.1 Benutzerberechtigungen	495
20.2.2 Single-Threaded-Ansatz	496
20.2.3 Denial of Service	499
20.2.4 Reguläre Ausdrücke	500
20.2.5 HTTP-Header	501
20.2.6 Fehlermeldungen	504
20.2.7 SQL-Injections	504
20.2.8 »eval«	506
20.2.9 Method Invocation	508
20.2.10 Überschreiben von Built-ins	510
20.3 NPM-Sicherheit	511
20.3.1 Berechtigungen	512
20.3.2 Qualitätsaspekt	512
20.3.3 NPM Scripts	513
20.4 Schutz des Clients	514
20.4.1 Cross-Site-Scripting	515
20.4.2 Cross-Site-Request-Forgery	516
20.5 Zusammenfassung	518

21 Skalierbarkeit und Deployment	519
21.1 Deployment	519
21.1.1 Einfaches Deployment	520
21.1.2 Dateisynchronisierung mit rsync	521
21.1.3 Die Applikation als Dienst	522
21.1.4 node_modules beim Deployment	525
21.1.5 Applikationen mit dem Node Package Manager installieren	526
21.1.6 Pakete lokal installieren	527
21.2 Toolunterstützung	528
21.2.1 Grunt	528
21.2.2 Gulp	533
21.2.3 NPM	534
21.3 Skalierung	534
21.3.1 Kindprozesse	535
21.3.2 Loadbalancer	538
21.3.3 Node in der Cloud	541
21.4 pm2 – Prozessmanagement	543
21.5 Zusammenfassung	544
Index	545