

# Auf einen Blick

1	Grundlagen .....	31
2	Installation .....	59
3	Ein erstes Beispiel .....	79
4	Node.js-Module .....	109
5	HTTP .....	149
6	Express .....	191
7	Template-Engines .....	229
8	Anbindung von Datenbanken .....	259
9	Authentifizierung und Sessionhandling .....	295
10	REST-Server .....	319
11	GraphQL .....	363
12	Echtzeit-Webapplikationen .....	383
13	Typsichere Applikationen in Node.js .....	411
14	Webapplikationen mit Nest .....	441
15	Node auf der Kommandozeile .....	485
16	Asynchrone Programmierung .....	515
17	RxJS .....	557
18	Streams .....	585
19	Arbeiten mit Dateien .....	617
20	Socket-Server .....	641
21	Package Manager .....	663
22	Qualitätssicherung .....	691
23	Testing .....	705
24	Sicherheitsaspekte .....	735
25	Skalierbarkeit und Deployment .....	767
26	Performance .....	791
27	Microservices mit Node.js .....	811
28	Deno .....	851

# Inhalt

Materialien zum Buch .....	23
Geleitwort .....	25
Vorwort .....	27

## 1 Grundlagen

<b>1.1 Die Geschichte von Node.js .....</b>	33
1.1.1 Die Ursprünge .....	33
1.1.2 Die Geburt von Node.js .....	34
1.1.3 Der Durchbruch von Node.js .....	35
1.1.4 Node.js erobert Windows .....	36
1.1.5 io.js – der Fork von Node.js .....	36
1.1.6 Node.js wieder vereint .....	37
1.1.7 Deno – ein neuer Stern am JavaScript-Himmel .....	37
1.1.8 Die OpenJS Foundation .....	37
<b>1.2 Die Organisation von Node.js .....</b>	38
1.2.1 Das Technical Steering Committee .....	38
1.2.2 Collaborators .....	38
1.2.3 Das Community Committee .....	39
1.2.4 Arbeitsgruppen .....	39
1.2.5 Die OpenJS Foundation .....	39
<b>1.3 Versionierung von Node.js .....</b>	39
1.3.1 LTS-Releases .....	40
<b>1.4 Vorteile von Node.js .....</b>	41
<b>1.5 Einsatzgebiete von Node.js .....</b>	42
<b>1.6 Das Herzstück – die V8-Engine .....</b>	43
1.6.1 Das Speichermodell .....	44
1.6.2 Zugriff auf Eigenschaften .....	45
1.6.3 Maschinencodegenerierung .....	47
1.6.4 Garbage Collection .....	49
<b>1.7 Bibliotheken um die Engine .....</b>	50
1.7.1 Event-Loop .....	51
1.7.2 Eingabe und Ausgabe .....	53
1.7.3 libuv .....	54

1.7.4	DNS .....	55
1.7.5	Crypto .....	55
1.7.6	Zlib .....	56
1.7.7	HTTP-Parser .....	56
<b>1.8</b>	<b>Zusammenfassung .....</b>	<b>57</b>

---

## 2 Installation

---

<b>2.1</b>	<b>Installation von Paketen .....</b>	<b>60</b>
2.1.1	Linux .....	61
2.1.2	Windows .....	64
2.1.3	macOS .....	68
<b>2.2</b>	<b>Komplizieren und installieren .....</b>	<b>74</b>
<b>2.3</b>	<b>Node Version Manager .....</b>	<b>76</b>
<b>2.4</b>	<b>Node und Docker .....</b>	<b>77</b>
<b>2.5</b>	<b>Zusammenfassung .....</b>	<b>78</b>

---

## 3 Ein erstes Beispiel

---

<b>3.1</b>	<b>Der interaktive Modus .....</b>	<b>79</b>
3.1.1	Generelle Benutzung .....	80
3.1.2	Weitere REPL-Befehle .....	81
3.1.3	Speichern und Laden im REPL .....	83
3.1.4	Kontext des REPL .....	83
3.1.5	REPL-Historie .....	84
3.1.6	REPL-Modus .....	84
3.1.7	Suche im REPL .....	85
3.1.8	Asynchrone Operationen im REPL .....	85
<b>3.2</b>	<b>Die erste Applikation .....</b>	<b>86</b>
3.2.1	Ein Webserver in Node.js .....	87
3.2.2	Erweiterung des Webservers .....	91
3.2.3	Erstellen einer HTML-Antwort .....	93
3.2.4	Dynamische Antworten generieren .....	94
<b>3.3</b>	<b>Debuggen von Node.js-Applikationen .....</b>	<b>96</b>
3.3.1	Navigation im Debugger .....	98
3.3.2	Informationen im Debugger .....	99

---

3.3.3	Breakpoints .....	101
3.3.4	Debuggen mit den Chrome Developer Tools .....	104
3.3.5	Debugging in der Entwicklungsumgebung .....	106
<b>3.4</b>	<b>Entwicklungswerzeug »nodemon« .....</b>	<b>107</b>
<b>3.5</b>	<b>Zusammenfassung .....</b>	<b>108</b>
<b>4</b>	<b>Node.js-Module</b>	<b>109</b>
<b>4.1</b>	<b>Modularer Aufbau .....</b>	<b>109</b>
<b>4.2</b>	<b>Kernmodule .....</b>	<b>112</b>
4.2.1	Stabilität .....	112
4.2.2	Liste der Kernmodule .....	114
4.2.3	Laden von Kernmodulen .....	117
4.2.4	Globale Objekte .....	120
<b>4.3</b>	<b>JavaScript-Modulsysteme .....</b>	<b>132</b>
4.3.1	CommonJS .....	132
4.3.2	ECMAScript-Module .....	133
<b>4.4</b>	<b>Eigene Module erzeugen und verwenden .....</b>	<b>135</b>
4.4.1	Module in Node.js – CommonJS .....	136
4.4.2	Eigene Node.js-Module .....	137
4.4.3	Module in Node.js – ECMAScript .....	138
4.4.4	Verschiedene Datentypen exportieren .....	140
4.4.5	Das »modules«-Modul .....	142
4.4.6	Der Modulloader .....	143
<b>4.5</b>	<b>Zusammenfassung .....</b>	<b>148</b>
<b>5</b>	<b>HTTP</b>	<b>149</b>
<b>5.1</b>	<b>Der Webserver .....</b>	<b>149</b>
5.1.1	Das »Server«-Objekt .....	149
5.1.2	Server-Events .....	155
5.1.3	Das »Request«-Objekt .....	158
5.1.4	Umgang mit dem Request-Body (Update) .....	165
5.1.5	Ausliefern von statischen Inhalten .....	170
5.1.6	Dateiupload .....	173
5.1.7	Feinschliff am Frontend .....	177

<b>5.2 Node.js als HTTP-Client .....</b>	178
5.2.1 Requests mit dem »http«-Modul .....	178
5.2.2 Das »request«-Paket .....	179
5.2.3 HTML-Parser .....	181
<b>5.3 Sichere Kommunikation mit HTTPS .....</b>	182
5.3.1 Zertifikate erstellen .....	183
5.3.2 HTTPS im Webserver verwenden .....	183
<b>5.4 HTTP/2 .....</b>	184
5.4.1 Der HTTP/2-Server .....	185
5.4.2 Der HTTP/2-Client .....	188
<b>5.5 Zusammenfassung .....</b>	189

---

## **6 Express**

---

<b>6.1 Aufbau .....</b>	191
<b>6.2 Installation .....</b>	192
<b>6.3 Grundlagen .....</b>	194
6.3.1 Request .....	194
6.3.2 Response .....	195
<b>6.4 Setup .....</b>	196
6.4.1 Struktur einer Applikation .....	196
<b>6.5 Movie-Datenbank .....</b>	199
6.5.1 Routing .....	201
6.5.2 Controller .....	203
6.5.3 Model .....	205
6.5.4 View .....	207
<b>6.6 Middleware .....</b>	209
6.6.1 Eigene Middleware .....	209
6.6.2 Morgan – Logging-Middleware für Express .....	210
6.6.3 Statische Inhalte ausliefern .....	212
<b>6.7 Erweitertes Routing – Löschen von Datensätzen .....</b>	214
<b>6.8 Anlegen und Bearbeiten von Datensätzen – Body-Parser .....</b>	217
6.8.1 Umgang mit Formulareingaben – Body-Parser .....	220
<b>6.9 Express 5 .....</b>	224
<b>6.10 HTTPS und HTTP/2 .....</b>	225
6.10.1 HTTPS .....	225

---

6.10.2	HTTP/2 .....	226
<b>6.11</b>	<b>Zusammenfassung .....</b>	<b>228</b>

---

## **7 Template-Engines**

---

<b>7.1</b>	<b>Eine eigene Template-Engine .....</b>	<b>230</b>
<b>7.2</b>	<b>Template-Engines in der Praxis – Pug .....</b>	<b>232</b>
7.2.1	Installation .....	232
7.2.2	Pug und Express.js – Integration .....	232
7.2.3	Variablen in Pug .....	236
7.2.4	Die Besonderheiten von Pug .....	237
7.2.5	Bedingungen und Schleifen .....	238
7.2.6	Extends und Includes .....	240
7.2.7	Mixins .....	243
7.2.8	Pug unabhängig von Express verwenden .....	245
7.2.9	Compiling .....	245
<b>7.3</b>	<b>Handlebars .....</b>	<b>247</b>
7.3.1	Installation .....	247
7.3.2	Integration in Express.js .....	247
7.3.3	Bedingungen und Schleifen .....	250
7.3.4	Partials .....	251
7.3.5	Eigene Helper .....	254
7.3.6	Handlebars ohne Express .....	256
<b>7.4</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>257</b>

---

## **8 Anbindung von Datenbanken**

---

<b>8.1</b>	<b>Node.js und relationale Datenbanken .....</b>	<b>260</b>
8.1.1	MySQL .....	261
8.1.2	SQLite .....	270
8.1.3	ORM .....	277
<b>8.2</b>	<b>Node.js und nicht relationale Datenbanken .....</b>	<b>280</b>
8.2.1	Redis .....	280
8.2.2	MongoDB .....	286
<b>8.3</b>	<b>Zusammenfassung .....</b>	<b>293</b>

<b>9 Authentifizierung und Sessionhandling</b>	295
<b>9.1 Passport</b> .....	295
<b>9.2 Setup und Konfiguration</b> .....	296
9.2.1 Installation .....	296
9.2.2 Konfiguration .....	296
9.2.3 Konfiguration der Strategy .....	298
<b>9.3 Anmeldung an der Applikation</b> .....	300
9.3.1 Anmeldeformular .....	300
9.3.2 Absicherung von Ressourcen .....	303
9.3.3 Abmelden .....	304
9.3.4 Anbindung an die Datenbank .....	305
<b>9.4 Zugriff auf Ressourcen</b> .....	308
9.4.1 Zugriffsbeschränkung .....	308
9.4.2 Bewertungen abgeben .....	313
<b>9.5 Zusammenfassung</b> .....	317
 <b>10 REST-Server</b>	319
<b>10.1 REST – eine kurze Einführung und wie es in Webapplikationen verwendet wird</b> .....	319
<b>10.2 Zugriff auf die Applikation</b> .....	320
10.2.1 Postman .....	320
10.2.2 cURL .....	321
<b>10.3 Anpassungen an der Applikationsstruktur</b> .....	322
<b>10.4 Lesende Anfragen</b> .....	323
10.4.1 Alle Datensätze einer Ressource auslesen .....	323
10.4.2 Zugriff auf einen Datensatz .....	325
10.4.3 Fehlerbehandlung .....	327
10.4.4 Sortieren der Liste .....	329
10.4.5 Steuern des Ausgabeformats .....	332
<b>10.5 Schreibende Anfragen</b> .....	334
10.5.1 POST – Erstellen von neuen Datensätzen .....	334
10.5.2 PUT – bestehende Datensätze modifizieren .....	337
10.5.3 DELETE – Datensätze löschen .....	340
<b>10.6 Authentifizierung mit JSON Web Tokens</b> .....	342
10.6.1 Anmeldung .....	343

10.6.2	Absichern von Ressourcen .....	345
10.6.3	Zugriff auf Benutzerinformationen im Token .....	348
<b>10.7</b>	<b>OpenAPI-Spezifikation – Dokumentation mit Swagger .....</b>	<b>351</b>
<b>10.8</b>	<b>Validierung .....</b>	<b>355</b>
10.8.1	Installation und erste Überprüfung .....	356
10.8.2	Anfragen mit einem Validierungsschema überprüfen .....	358
<b>10.9</b>	<b>Zusammenfassung .....</b>	<b>362</b>

---

## **11 GraphQL**

---

<b>11.1</b>	<b>GraphQL-Bibliotheken .....</b>	<b>364</b>
<b>11.2</b>	<b>Integration in Express .....</b>	<b>365</b>
11.2.1	GraphiQL .....	367
<b>11.3</b>	<b>Daten über die Schnittstelle auslesen .....</b>	<b>369</b>
11.3.1	Abfragen parametrisieren .....	371
<b>11.4</b>	<b>Schreibende Zugriffe auf die GraphQL-Schnittstelle .....</b>	<b>374</b>
11.4.1	Neue Datensätze erstellen .....	374
11.4.2	Aktualisieren und Löschen von Datensätzen .....	377
<b>11.5</b>	<b>Authentifizierung für die GraphQL-Schnittstelle .....</b>	<b>380</b>
<b>11.6</b>	<b>Zusammenfassung .....</b>	<b>382</b>

---

## **12 Echtzeit-Webapplikationen**

---

<b>12.1</b>	<b>Die Beispielapplikation .....</b>	<b>384</b>
<b>12.2</b>	<b>Setup .....</b>	<b>385</b>
<b>12.3</b>	<b>Websockets .....</b>	<b>391</b>
12.3.1	Die Serverseite .....	393
12.3.2	Die Clientseite .....	394
12.3.3	User-Liste .....	397
12.3.4	Logout .....	400
<b>12.4</b>	<b>Socket.IO .....</b>	<b>404</b>
12.4.1	Installation und Einbindung .....	405
12.4.2	Socket.IO-API .....	406
<b>12.5</b>	<b>Zusammenfassung .....</b>	<b>410</b>

<b>13 Typsichere Applikationen in Node.js</b>	411
<b>13.1 Typsysteme für Node.js</b>	412
13.1.1 Flow .....	412
13.1.2 TypeScript .....	417
<b>13.2 Werkzeuge und Konfiguration</b> .....	419
13.2.1 Konfiguration des TypeScript-Compilers .....	419
13.2.2 Integration in die Entwicklungsumgebung .....	421
13.2.3 ESLint .....	421
13.2.4 »ts-node« .....	423
<b>13.3 Grundlagen</b> .....	425
13.3.1 Datentypen .....	425
13.3.2 Funktionen .....	427
13.3.3 Module .....	429
<b>13.4 Klassen</b> .....	430
13.4.1 Methoden .....	431
13.4.2 Zugriffsmodifikatoren .....	432
13.4.3 Vererbung .....	433
<b>13.5 Interfaces</b> .....	433
<b>13.6 Type Aliases in TypeScript</b> .....	435
<b>13.7 Generics</b> .....	436
<b>13.8 TypeScript im Einsatz in einer Node.js-Applikation</b> .....	438
13.8.1 Typdefinitionen .....	438
13.8.2 Eigene Typdefinitionen erzeugen .....	438
13.8.3 Beispiel einer Express-Applikation .....	439
<b>13.9 Zusammenfassung</b> .....	440
<b>14 Webapplikationen mit Nest</b>	441
<b>14.1 Installation und erste Schritte mit Nest</b> .....	442
<b>14.2 Die Nest CLI</b> .....	444
14.2.1 Kommandos für den Betrieb und das Ausführen der Applikation .....	445
14.2.2 Erstellen von Strukturen in der Applikation .....	446
<b>14.3 Struktur der Applikation</b> .....	448
14.3.1 Das Wurzelverzeichnis mit den Konfigurationsdateien .....	448
14.3.2 Das src-Verzeichnis – das Herzstück der Applikation .....	450
14.3.3 Weitere Verzeichnisse der Applikation .....	450

---

<b>14.4</b>	<b>Module – logische Einheiten im Quellcode</b>	450
14.4.1	Module erzeugen .....	451
14.4.2	Der Module-Decorator .....	452
<b>14.5</b>	<b>Controller – die Endpunkte einer Applikation</b>	453
14.5.1	Einen Controller erzeugen .....	453
14.5.2	Implementierung eines Controllers .....	454
14.5.3	Einbindung und Überprüfung des Controllers .....	456
<b>14.6</b>	<b>Providers – die Businesslogik der Applikation</b>	458
14.6.1	Einen Service erzeugen und einbinden .....	458
14.6.2	Die Implementierung des Service .....	459
14.6.3	Einbindung des Service über die Dependency Injection von Nest .....	461
<b>14.7</b>	<b>Zugriff auf Datenbanken</b>	462
14.7.1	Setup und Installation .....	463
14.7.2	Zugriff auf die Datenbank .....	465
<b>14.8</b>	<b>Dokumentation der Endpunkte mit OpenAPI</b>	470
<b>14.9</b>	<b>Authentifizierung</b>	473
14.9.1	Setup .....	474
14.9.2	Authentifizierungsservice .....	475
14.9.3	Der Login-Controller – der Endpunkt für die Benutzeranmeldung .....	476
14.9.4	Routen absichern .....	478
<b>14.10</b>	<b>Ausblick: Testen in Nest</b>	480
<b>14.11</b>	<b>Zusammenfassung</b>	483

---

## 15 Node auf der Kommandozeile

		485
--	--	-----

<b>15.1</b>	<b>Grundlagen</b>	485
15.1.1	Aufbau .....	486
15.1.2	Ausführbarkeit .....	487
<b>15.2</b>	<b>Der Aufbau einer Kommandozeilenapplikation</b>	488
15.2.1	Datei und Verzeichnisstruktur .....	488
15.2.2	Paketdefinition .....	489
15.2.3	Die Mathe-Trainer-Applikation .....	490
<b>15.3</b>	<b>Zugriff auf Ein- und Ausgabe</b>	493
15.3.1	Ausgabe .....	494
15.3.2	Eingabe .....	495
15.3.3	Benutzerinteraktion mit dem »readline«-Modul .....	496
15.3.4	Optionen und Argumente .....	500

<b>15.4 Werkzeuge .....</b>	503
15.4.1 Commander .....	503
15.4.2 Chalk .....	505
15.4.3 node-emoji .....	508
<b>15.5 Signale .....</b>	510
<b>15.6 Exit Codes .....</b>	513
<b>15.7 Zusammenfassung .....</b>	514

---

## **16 Asynchrone Programmierung**

---

<b>16.1 Grundlagen asynchroner Programmierung .....</b>	515
<b>16.2 Externe Kommandos asynchron ausführen .....</b>	521
16.2.1 Die »exec«-Methode .....	521
16.2.2 Die »spawn«-Methode .....	524
<b>16.3 Node.js-Kindprozesse erzeugen mit »fork« .....</b>	527
<b>16.4 Das »cluster«-Modul .....</b>	532
16.4.1 Der Hauptprozess .....	533
16.4.2 Die Workerprozesse .....	537
<b>16.5 Worker-Threads .....</b>	541
16.5.1 Geteilter Speicher im »worker_threads«-Modul .....	542
<b>16.6 Promises in Node.js .....</b>	544
16.6.1 Mit »util.promisify« Promises dort verwenden, wo es eigentlich keine gibt .....	547
16.6.2 Verkettung von Promises .....	548
16.6.3 Mehrere parallele Operationen mit »Promise.all« .....	550
16.6.4 Die schnellste asynchrone Operation mit »Promise.race« .....	551
16.6.5 Die Promise-Funktionen im Überblick .....	551
<b>16.7 Async Functions .....</b>	552
16.7.1 Toplevel await .....	554
<b>16.8 Zusammenfassung .....</b>	555

---

## 17 RxJS

557

---

<b>17.1 Grundlagen</b> .....	558
17.1.1 Observable .....	559
17.1.2 Observer .....	560
17.1.3 Operator .....	562
17.1.4 Beispiel für RxJS in Node .....	562
<b>17.2 Operatoren</b> .....	564
17.2.1 Erstellende Operatoren .....	566
17.2.2 Transformierende Operatoren .....	568
17.2.3 Filteroperatoren .....	571
17.2.4 Kombinierende Operatoren .....	573
17.2.5 Operatoren zur Fehlerbehandlung .....	575
17.2.6 Hilfsoperatoren .....	576
17.2.7 Bedingungsoperatoren .....	578
17.2.8 Verbindungsoperatoren .....	579
17.2.9 Konvertierungsoperator .....	580
<b>17.3 Subjects</b> .....	581
<b>17.4 Scheduler</b> .....	582
<b>17.5 Zusammenfassung</b> .....	583

---

## 18 Streams

585

---

<b>18.1 Einleitung</b> .....	585
18.1.1 Was ist ein Stream? .....	585
18.1.2 Wozu verwendet man Streams? .....	586
18.1.3 Welche Streams gibt es? .....	587
18.1.4 Streamversionen in Node.js .....	587
18.1.5 Streams sind EventEmitter .....	588
<b>18.2 Readable Streams</b> .....	589
18.2.1 Einen Readable Stream erstellen .....	589
18.2.2 Die Readable-Stream-Schnittstelle .....	590
18.2.3 Die Events eines Readable Streams .....	591
18.2.4 Fehlerbehandlung in Readable Streams .....	592
18.2.5 Methoden .....	594
18.2.6 Piping .....	594
18.2.7 Readable-Stream-Modi .....	595
18.2.8 Wechsel in den Flowing Mode .....	595

18.2.9 Wechsel in den Paused Mode .....	596
18.2.10 Eigene Readable Streams .....	596
18.2.11 Beispiel für einen Readable Stream .....	597
18.2.12 Readable-Shortcut .....	600
<b>18.3 Writable Streams .....</b>	<b>601</b>
18.3.1 Einen Writable Stream erstellen .....	601
18.3.2 Events .....	602
18.3.3 Fehlerbehandlung in Writable Streams .....	604
18.3.4 Methoden .....	604
18.3.5 Schreiboperationen puffern .....	605
18.3.6 Flusssteuerung .....	606
18.3.7 Eigene Writable Streams .....	607
18.3.8 Writable-Shortcut .....	608
<b>18.4 Duplex-Streams .....</b>	<b>609</b>
18.4.1 Duplex-Streams im Einsatz .....	609
18.4.2 Eigene Duplex-Streams .....	609
18.4.3 Duplex-Shortcut .....	610
<b>18.5 Transform-Streams .....</b>	<b>611</b>
18.5.1 Eigene Transform-Streams .....	611
18.5.2 Transform-Shortcut .....	612
<b>18.6 Gulp .....</b>	<b>613</b>
18.6.1 Installation .....	613
18.6.2 Beispiel für einen Build-Prozess mit Gulp .....	613
<b>18.7 Zusammenfassung .....</b>	<b>615</b>

---

<b>19 Arbeiten mit Dateien</b>	<b>617</b>
<b>19.1 Synchrone und asynchrone Funktionen</b> .....	617
<b>19.2 Existenz von Dateien</b> .....	619
<b>19.3 Dateien lesen</b> .....	620
19.3.1 Die promisebasierte API .....	625
<b>19.4 Fehlerbehandlung</b> .....	626
<b>19.5 In Dateien schreiben</b> .....	627
<b>19.6 Verzeichnisoperationen</b> .....	631
<b>19.7 Weiterführende Operationen</b> .....	634
19.7.1 »watch« .....	637

19.7.2 Zugriffsberechtigungen .....	638
<b>19.8 Zusammenfassung .....</b>	<b>640</b>

---

## **20 Socket-Server**

---

<b>20.1 Unix-Sockets .....</b>	<b>642</b>
20.1.1 Zugriff auf den Socket .....	645
20.1.2 Bidirektionale Kommunikation .....	647
<b>20.2 Windows Pipes .....</b>	<b>649</b>
<b>20.3 TCP-Sockets .....</b>	<b>650</b>
20.3.1 Datenübertragung .....	652
20.3.2 Dateiübertragung .....	653
20.3.3 Flusssteuerung .....	654
20.3.4 Duplex .....	656
20.3.5 Pipe .....	656
<b>20.4 UDP-Sockets .....</b>	<b>657</b>
20.4.1 Grundlagen eines UDP-Servers .....	658
20.4.2 Beispiel zum UDP-Server .....	660
<b>20.5 Zusammenfassung .....</b>	<b>662</b>

---

## **21 Package Manager**

---

<b>21.1 Die häufigsten Operationen .....</b>	<b>664</b>
21.1.1 Pakete suchen .....	664
21.1.2 Pakete installieren .....	665
21.1.3 Installierte Pakete anzeigen .....	671
21.1.4 Pakete verwenden .....	672
21.1.5 Pakete aktualisieren .....	673
21.1.6 Pakete entfernen .....	675
21.1.7 Die wichtigsten Kommandos im Überblick .....	676
<b>21.2 Weiterführende Operationen .....</b>	<b>677</b>
21.2.1 Der Aufbau eines Moduls .....	677
21.2.2 Eigene Pakete erstellen .....	680
21.2.3 NPM-Skripte .....	683
<b>21.3 Werkzeuge für NPM .....</b>	<b>685</b>
21.3.1 Node License Finder .....	685

21.3.2	Verdaccio .....	686
21.3.3	»npm-check-updates« .....	686
21.3.4	»npx« .....	687
<b>21.4</b>	<b>Yarn .....</b>	<b>688</b>
<b>21.5</b>	<b>Zusammenfassung .....</b>	<b>689</b>

---

## **22 Qualitätssicherung**

---

<b>22.1</b>	<b>Styleguides .....</b>	<b>692</b>
22.1.1	Der Airbnb-Styleguide .....	692
<b>22.2</b>	<b>Linter .....</b>	<b>693</b>
22.2.1	ESLint .....	694
<b>22.3</b>	<b>Prettier .....</b>	<b>699</b>
22.3.1	Installation .....	699
22.3.2	Ausführung .....	699
<b>22.4</b>	<b>PMD CPD .....</b>	<b>700</b>
22.4.1	Installation .....	701
22.4.2	Ausführung .....	702
<b>22.5</b>	<b>Husky .....</b>	<b>703</b>
<b>22.6</b>	<b>Zusammenfassung .....</b>	<b>704</b>

---

## **23 Testing**

---

<b>23.1</b>	<b>Unittesting .....</b>	<b>705</b>
23.1.1	Verzeichnisstruktur .....	706
23.1.2	Unitests und Node.js .....	707
23.1.3	Triple-A .....	707
<b>23.2</b>	<b>Assertion Testing .....</b>	<b>708</b>
23.2.1	Exceptions .....	711
23.2.2	Promises testen .....	712
<b>23.3</b>	<b>Jasmine .....</b>	<b>714</b>
23.3.1	Installation .....	715
23.3.2	Konfiguration .....	715
23.3.3	Tests in Jasmine .....	716

---

23.3.4 Assertions .....	719
23.3.5 Spys .....	721
23.3.6 »beforeEach« und »afterEach« .....	722
<b>23.4 Jest .....</b>	<b>723</b>
23.4.1 Installation .....	723
23.4.2 Ein erster Test .....	723
<b>23.5 Praktisches Beispiel von Unitests mit »Jest« .....</b>	<b>726</b>
23.5.1 Der Test .....	727
23.5.2 Die Implementierung .....	728
23.5.3 Triangulation - der zweite Test .....	729
23.5.4 Verbesserung der Implementierung .....	731
<b>23.6 Umgang mit Abhängigkeiten – Mocking .....</b>	<b>731</b>
<b>23.7 Zusammenfassung .....</b>	<b>734</b>

---

## 24 Sicherheitsaspekte 735

<b>24.1 »filter input« und »escape output« .....</b>	<b>736</b>
24.1.1 »filter input« .....	736
24.1.2 Black- und Whitelisting .....	736
24.1.3 »escape output« .....	737
<b>24.2 Absicherung des Servers .....</b>	<b>739</b>
24.2.1 Benutzerberechtigungen .....	739
24.2.2 Probleme durch den Single-threaded-Ansatz .....	740
24.2.3 Denial of Service .....	743
24.2.4 Reguläre Ausdrücke .....	744
24.2.5 HTTP-Header .....	745
24.2.6 Fehlermeldungen .....	748
24.2.7 SQL-Injections .....	748
24.2.8 »eval« .....	752
24.2.9 Method Invocation .....	754
24.2.10 Überschreiben von Built-ins .....	756
<b>24.3 NPM-Sicherheit .....</b>	<b>758</b>
24.3.1 Berechtigungen .....	758
24.3.2 Node Security Platform .....	759
24.3.3 Qualitätsaspekt .....	759
24.3.4 NPM-Skripte .....	761

<b>24.4 Schutz des Clients .....</b>	761
24.4.1 Cross-Site-Scripting .....	762
24.4.2 Cross-Site-Request-Forgery .....	763
<b>24.5 Zusammenfassung .....</b>	766

## 25 Skalierbarkeit und Deployment

767

---

<b>25.1 Deployment .....</b>	767
25.1.1 Einfaches Deployment .....	768
25.1.2 Dateisynchronisierung mit »rsync« .....	769
25.1.3 Die Applikation als Dienst .....	770
25.1.4 »node_modules« beim Deployment .....	773
25.1.5 Applikationen mit dem Node Package Manager installieren .....	773
25.1.6 Pakete lokal installieren .....	775
<b>25.2 Toolunterstützung .....</b>	775
25.2.1 Grunt .....	776
25.2.2 Gulp .....	776
25.2.3 NPM .....	776
<b>25.3 Skalierung .....</b>	777
25.3.1 Kindprozesse .....	778
25.3.2 Loadbalancer .....	782
25.3.3 Node in der Cloud .....	784
<b>25.4 »pm2« – Prozessmanagement .....</b>	786
<b>25.5 Docker .....</b>	787
25.5.1 Das Dockerfile .....	788
25.5.2 Container starten .....	788
<b>25.6 Zusammenfassung .....</b>	789

## 26 Performance

791

---

<b>26.1 YAGNI – You Ain't Gonna Need It .....</b>	791
<b>26.2 CPU .....</b>	792
26.2.1 CPU-blockierende Operationen .....	792
26.2.2 Die CPU-Last messen .....	793
26.2.3 CPU-Profilierung mit den Chrome DevTools .....	794

26.2.4	Alternativen zum Profiler – console.time .....	796
26.2.5	Alternativen zum Profiler – die Performance-Hooks-Schnittstelle ....	797
<b>26.3</b>	<b>Arbeitsspeicher .....</b>	<b>800</b>
26.3.1	Memory Leaks .....	801
26.3.2	Speicheranalyse in den DevTools .....	802
26.3.3	Speicherstatistik von Node.js .....	805
<b>26.4</b>	<b>Netzwerk .....</b>	<b>806</b>
<b>26.5</b>	<b>Zusammenfassung .....</b>	<b>810</b>

---

## **27 Microservices mit Node.js**

---

<b>27.1</b>	<b>Grundlagen .....</b>	<b>811</b>
27.1.1	Monolithische Architektur .....	811
27.1.2	Microservice-Architektur .....	813
<b>27.2</b>	<b>Architektur .....</b>	<b>814</b>
27.2.1	Kommunikation zwischen den einzelnen Services .....	815
<b>27.3</b>	<b>Die Infrastruktur .....</b>	<b>816</b>
27.3.1	Docker Compose .....	817
<b>27.4</b>	<b>Ein asynchroner Microservice mit RabbitMQ .....</b>	<b>818</b>
27.4.1	Installation und Setup .....	818
27.4.2	Verbindung mit dem RabbitMQ-Server .....	821
27.4.3	Mit eingehenden Nachrichten umgehen .....	822
27.4.4	Datenbankanbindung .....	823
27.4.5	Docker-Setup .....	824
<b>27.5</b>	<b>API-Gateway .....</b>	<b>827</b>
27.5.1	Anbindung des User-Service .....	827
27.5.2	Asynchrone Kommunikation mit dem User-Service .....	830
27.5.3	Docker-Setup des API-Gateways .....	834
27.5.4	Authentifizierung .....	836
<b>27.6</b>	<b>Synchroner Microservice mit Express .....</b>	<b>840</b>
27.6.1	Setup .....	841
27.6.2	Controller .....	842
27.6.3	Model-Implementierung .....	842
27.6.4	Docker-Setup .....	844
27.6.5	Einbindung ins API-Gateway .....	846
<b>27.7</b>	<b>Zusammenfassung .....</b>	<b>849</b>

## 28 Deno

851

<b>28.1 Die zehn Dinge, die Ryan Dahl bereut</b> .....	851
28.1.1 Promises .....	852
28.1.2 Sicherheit .....	852
28.1.3 Das GYP-Build-System (GYP) .....	852
28.1.4 package.json .....	852
28.1.5 node_modules .....	853
28.1.6 Optionale Dateiendung beim Laden von Modulen .....	853
28.1.7 Index.js .....	853
28.1.8 Und wie sieht es jetzt auf der Node.js-Seite aus? .....	853
<b>28.2 Installation von Deno</b> .....	854
28.2.1 Die Deno CLI .....	854
<b>28.3 Ausführung</b> .....	855
28.3.1 Ausführung einer TypeScript-Applikation .....	856
<b>28.4 Arbeiten mit Dateien</b> .....	857
28.4.1 Die Aufgabenstellung: Kopieren einer Datei .....	857
28.4.2 Verarbeiten von Kommandozeilenoptionen .....	857
28.4.3 Dateien lesen .....	859
28.4.4 Berechtigungen in Deno .....	860
28.4.5 Die readTextFile-Funktion .....	862
28.4.6 Dateien mit Deno schreiben .....	862
<b>28.5 Ein Webserver mit Deno</b> .....	864
<b>28.6 Das Modulsystem</b> .....	866
28.6.1 Externe Module in Deno laden .....	867
28.6.2 deno.land/x .....	869
28.6.3 NPM-Pakete verwenden .....	869
<b>28.7 Zusammenfassung</b> .....	871
 Index .....	873