

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	31
1.1 Die Geschichte von Node.js	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
1.2 Die Organisation von Node.js	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
1.3 Versionierung von Node.js	39
1.3.1 LTS-Releases	40
1.4 Vorteile von Node.js	41
1.5 Einsatzgebiete von Node.js	42
1.6 Das Herzstück – die V8-Engine	43
1.6.1 Das Speichermodell	44
1.6.2 Zugriff auf Eigenschaften	45
1.6.3 Maschinencodgenerierung	47
1.6.4 Garbage Collection	49
1.7 Bibliotheken um die Engine	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
1.8	Zusammenfassung	57

2 Installation 59

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

3 Ein erstes Beispiel 79

3.1	Der interaktive Modus	79
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
3.2	Die erste Applikation	86
3.2.1	Ein Webserver in Node.js	87
3.2.2	Erweiterung des Webserver	91
3.2.3	Erstellen einer HTML-Antwort	93
3.2.4	Dynamische Antworten generieren	94
3.3	Debuggen von Node.js-Applikationen	96
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
3.4	Entwicklungswerkzeug »nodemon«	107
3.5	Zusammenfassung	108

4 Node.js-Module 109

4.1	Modularer Aufbau	109
4.2	Kernmodule	112
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
4.3	JavaScript-Modulsysteme	132
4.3.1	CommonJS	132
4.3.2	ECMAScript-Module	133
4.4	Eigene Module erzeugen und verwenden	135
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
4.5	Zusammenfassung	148

5 HTTP 149

5.1	Der Webserver	149
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

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

6 Express 191

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

6.10.2	HTTP/2	226
6.11	Zusammenfassung	228

7 **Template-Engines** 229

7.1	Eine eigene Template-Engine	230
7.2	Template-Engines in der Praxis – Pug	232
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
7.3	Handlebars	247
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
7.4	Zusammenfassung und Ausblick	257

8 **Anbindung von Datenbanken** 259

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

9	Authentifizierung und Sessionhandling	295
<hr/>		
9.1	Passport	295
9.2	Setup und Konfiguration	296
9.2.1	Installation	296
9.2.2	Konfiguration	296
9.2.3	Konfiguration der Strategy	298
9.3	Anmeldung an der Applikation	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
9.4	Zugriff auf Ressourcen	308
9.4.1	Zugriffsbeschränkung	308
9.4.2	Bewertungen abgeben	313
9.5	Zusammenfassung	317
10	REST-Server	319
<hr/>		
10.1	REST – eine kurze Einführung und wie es in Webapplikationen verwendet wird	319
10.2	Zugriff auf die Applikation	320
10.2.1	Postman	320
10.2.2	cURL	321
10.3	Anpassungen an der Applikationsstruktur	322
10.4	Lesende Anfragen	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
10.5	Schreibende Anfragen	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
10.6	Authentifizierung mit JSON Web Tokens	342
10.6.1	Anmeldung	343

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

11 GraphQL 363

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

12 Echtzeit-Webapplikationen 383

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

13	Typsichere Applikationen in Node.js	411
<hr/>		
13.1	Typsysteme für Node.js	412
13.1.1	Flow	412
13.1.2	TypeScript	417
13.2	Werkzeuge und Konfiguration	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
13.3	Grundlagen	425
13.3.1	Datentypen	425
13.3.2	Funktionen	427
13.3.3	Module	429
13.4	Klassen	430
13.4.1	Methoden	431
13.4.2	Zugriffsmodifikatoren	432
13.4.3	Vererbung	433
13.5	Interfaces	433
13.6	Type Aliases in TypeScript	435
13.7	Generics	436
13.8	TypeScript im Einsatz in einer Node.js-Applikation	438
13.8.1	Typdefinitionen	438
13.8.2	Eigene Typdefinitionen erzeugen	438
13.8.3	Beispiel einer Express-Applikation	439
13.9	Zusammenfassung	440
14	Webapplikationen mit Nest	441
<hr/>		
14.1	Installation und erste Schritte mit Nest	442
14.2	Die Nest CLI	444
14.2.1	Kommandos für den Betrieb und das Ausführen der Applikation	445
14.2.2	Erstellen von Strukturen in der Applikation	446
14.3	Struktur der Applikation	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

14.4	Module – logische Einheiten im Quellcode	450
14.4.1	Module erzeugen	451
14.4.2	Der Module-Decorator	452
14.5	Controller – die Endpunkte einer Applikation	453
14.5.1	Einen Controller erzeugen	453
14.5.2	Implementierung eines Controllers	454
14.5.3	Einbindung und Überprüfung des Controllers	456
14.6	Providers – die Businesslogik der Applikation	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
14.7	Zugriff auf Datenbanken	462
14.7.1	Setup und Installation	463
14.7.2	Zugriff auf die Datenbank	465
14.8	Dokumentation der Endpunkte mit OpenAPI	470
14.9	Authentifizierung	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
14.10	Ausblick: Testen in Nest	480
14.11	Zusammenfassung	483
15	Node auf der Kommandozeile	485
15.1	Grundlagen	485
15.1.1	Aufbau	486
15.1.2	Ausführbarkeit	487
15.2	Der Aufbau einer Kommandozeilenapplikation	488
15.2.1	Datei und Verzeichnisstruktur	488
15.2.2	Paketdefinition	489
15.2.3	Die Mathe-Trainer-Applikation	490
15.3	Zugriff auf Ein- und Ausgabe	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

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

16 Asynchrone Programmierung 515

16.1	Grundlagen asynchroner Programmierung	515
16.2	Externe Kommandos asynchron ausführen	521
16.2.1	Die »exec«-Methode	521
16.2.2	Die »spawn«-Methode	524
16.3	Node.js-Kindprozesse erzeugen mit »fork«	527
16.4	Das »cluster«-Modul	532
16.4.1	Der Hauptprozess	533
16.4.2	Die Workerprozesse	537
16.5	Worker-Threads	541
16.5.1	Geteilter Speicher im »worker_threads«-Modul	542
16.6	Promises in Node.js	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
16.7	Async Functions	552
16.7.1	Toplevel await	554
16.8	Zusammenfassung	555

17 RxJS	557
17.1 Grundlagen	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
17.2 Operatoren	564
17.2.1 Erstellende Operatoren	566
17.2.2 Transformierende Operatoren	568
17.2.3 Filteroperatoren	571
17.2.4 Kombinerende 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
17.3 Subjects	581
17.4 Scheduler	582
17.5 Zusammenfassung	583
18 Streams	585
18.1 Einleitung	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
18.2 Readable Streams	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
18.3	Writable Streams	601
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
18.4	Duplex-Streams	609
18.4.1	Duplex-Streams im Einsatz	609
18.4.2	Eigene Duplex-Streams	609
18.4.3	Duplex-Shortcut	610
18.5	Transform-Streams	611
18.5.1	Eigene Transform-Streams	611
18.5.2	Transform-Shortcut	612
18.6	Gulp	613
18.6.1	Installation	613
18.6.2	Beispiel für einen Build-Prozess mit Gulp	613
18.7	Zusammenfassung	615
19	Arbeiten mit Dateien	617

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

19.7.2	Zugriffsberechtigungen	638
19.8	Zusammenfassung	640
20	Socket-Server	641
20.1	Unix-Sockets	642
20.1.1	Zugriff auf den Socket	645
20.1.2	Bidirektionale Kommunikation	647
20.2	Windows Pipes	649
20.3	TCP-Sockets	650
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
20.4	UDP-Sockets	657
20.4.1	Grundlagen eines UDP-Servers	658
20.4.2	Beispiel zum UDP-Server	660
20.5	Zusammenfassung	662
21	Package Manager	663
21.1	Die häufigsten Operationen	664
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
21.2	Weiterführende Operationen	677
21.2.1	Der Aufbau eines Moduls	677
21.2.2	Eigene Pakete erstellen	680
21.2.3	NPM-Skripte	683
21.3	Werkzeuge für NPM	685
21.3.1	Node License Finder	685

21.3.2	Verdaccio	686
21.3.3	»npm-check-updates«	686
21.3.4	»npx«	687
21.4	Yarn	688
21.5	Zusammenfassung	689

22 Qualitätssicherung 691

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

23 Testing 705

23.1	Unittesting	705
23.1.1	Verzeichnisstruktur	706
23.1.2	Unittests und Node.js	707
23.1.3	Triple-A	707
23.2	Assertion Testing	708
23.2.1	Exceptions	711
23.2.2	Promises testen	712
23.3	Jasmine	714
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
23.4	Jest	723
23.4.1	Installation	723
23.4.2	Ein erster Test	723
23.5	Praktisches Beispiel von Unittests mit »Jest«	726
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
23.6	Umgang mit Abhängigkeiten – Mocking	731
23.7	Zusammenfassung	734
24	Sicherheitsaspekte	735

24.1	»filter input« und »escape output«	736
24.1.1	»filter input«	736
24.1.2	Black- und Whitelisting	736
24.1.3	»escape output«	737
24.2	Absicherung des Servers	739
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
24.3	NPM-Sicherheit	758
24.3.1	Berechtigungen	758
24.3.2	Node Security Platform	759
24.3.3	Qualitätsaspekt	759
24.3.4	NPM-Skripte	761

24.4	Schutz des Clients	761
24.4.1	Cross-Site-Scripting	762
24.4.2	Cross-Site-Request-Forgery	763
24.5	Zusammenfassung	766

25 Skalierbarkeit und Deployment 767

25.1	Deployment	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
25.2	Toolunterstützung	775
25.2.1	Grunt	776
25.2.2	Gulp	776
25.2.3	NPM	776
25.3	Skalierung	777
25.3.1	Kindprozesse	778
25.3.2	Loadbalancer	782
25.3.3	Node in der Cloud	784
25.4	»pm2« – Prozessmanagement	786
25.5	Docker	787
25.5.1	Das Dockerfile	788
25.5.2	Container starten	788
25.6	Zusammenfassung	789

26 Performance 791

26.1	YAGNI – You Ain’t Gonna Need It	791
26.2	CPU	792
26.2.1	CPU-blockierende Operationen	792
26.2.2	Die CPU-Last messen	793
26.2.3	CPU-Profilng 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
26.3	Arbeitsspeicher	800
26.3.1	Memory Leaks	801
26.3.2	Speicheranalyse in den DevTools	802
26.3.3	Speicherstatistik von Node.js	805
26.4	Netzwerk	806
26.5	Zusammenfassung	810
27	Microservices mit Node.js	811
<hr/>		
27.1	Grundlagen	811
27.1.1	Monolithische Architektur	811
27.1.2	Microservice-Architektur	813
27.2	Architektur	814
27.2.1	Kommunikation zwischen den einzelnen Services	815
27.3	Die Infrastruktur	816
27.3.1	Docker Compose	817
27.4	Ein asynchroner Microservice mit RabbitMQ	818
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
27.5	API-Gateway	827
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
27.6	Synchroner Microservice mit Express	840
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
27.7	Zusammenfassung	849

28 Deno	851
28.1 Die zehn Dinge, die Ryan Dahl bereut	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 Dateieindung 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
28.2 Installation von Deno	854
28.2.1 Die Deno CLI	854
28.3 Ausführung	855
28.3.1 Ausführung einer TypeScript-Applikation	856
28.4 Arbeiten mit Dateien	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
28.5 Ein Webserver mit Deno	864
28.6 Das Modulsystem	866
28.6.1 Externe Module in Deno laden	867
28.6.2 deno.land/x	869
28.6.3 NPM-Pakete verwenden	869
28.7 Zusammenfassung	871
Index	873