

Inhaltsverzeichnis

Vorwort	vii
----------------------	------------

Aktualisierungen in der dritten Auflage	xvii
--	-------------

I Einführung	1
---------------------------	----------

1 Schnellstart	3
1.1 Das HTML-Grundgerüst	6
1.2 Die Startdatei für das Bootstrapping	6
1.3 Das zentrale Angular-Modul	7
1.4 Die erste Komponente	8
2 Haben Sie alles, was Sie benötigen?	11
2.1 Visual Studio Code	11
2.2 Google Chrome	14
2.3 Paketverwaltung mit Node.js und NPM	14
2.4 Codebeispiele in diesem Buch	17
3 Angular CLI: der Codegenerator für unser Projekt	21
3.1 Das offizielle Tool für Angular	21
3.2 Installation	22
3.3 Die wichtigsten Befehle	23

II TypeScript	25
----------------------------	-----------

4 Einführung in TypeScript	27
4.1 Was ist TypeScript und wie setzen wir es ein?	27
4.2 Variablen: const, let und var	30
4.3 Die wichtigsten Basistypen	32
4.4 Klassen	35
4.5 Interfaces	39
4.6 Template-Strings	40

4.7	Arrow-Funktionen/Lambda-Ausdrücke	40
4.8	Spread-Operator und Rest-Syntax	42
4.9	Union Types	45
4.10	Destrukturierende Zuweisungen	45
4.11	Decorators	47
4.12	Optional Chaining	47
4.13	Nullish Coalescing	48

III BookMonkey 4: Schritt für Schritt zur App **51**

5	Projekt- und Prozessvorstellung	53
5.1	Unser Projekt: BookMonkey	53
5.2	Projekt mit Angular CLI initialisieren	57
5.3	Style-Framework Semantic UI einbinden	70
6	Komponenten & Template-Syntax: Iteration I	73
6.1	Komponenten: die Grundbausteine der Anwendung	73
6.1.1	Komponenten	74
6.1.2	Komponenten in der Anwendung verwenden	79
6.1.3	Template-Syntax	80
6.1.4	Den BookMonkey erstellen	90
6.2	Property Bindings: mit Komponenten kommunizieren	102
6.2.1	Komponenten verschachteln	102
6.2.2	Eingehender Datenfluss mit Property Bindings	103
6.2.3	Andere Arten von Property Bindings	106
6.2.4	DOM-Propertys in Komponenten auslesen	109
6.2.5	Den BookMonkey erweitern	110
6.3	Event Bindings: auf Ereignisse in Komponenten reagieren	114
6.3.1	Native DOM-Events	114
6.3.2	Eigene Events definieren	117
6.3.3	Den BookMonkey erweitern	119
7	Powertipp: Styleguide	129
8	Services & Routing: Iteration II	131
8.1	Dependency Injection: Code in Services auslagern	131
8.1.1	Abhängigkeiten anfordern	133
8.1.2	Services in Angular	134
8.1.3	Abhängigkeiten registrieren	134
8.1.4	Abhängigkeiten ersetzen	137
8.1.5	Eigene Tokens definieren mit InjectionToken	140
8.1.6	Abhängigkeiten anfordern mit @Inject()	141

8.1.7	Multiprovider: mehrere Abhängigkeiten im selben Token	141
8.1.8	Zirkuläre Abhängigkeiten auflösen mit forwardRef ...	142
8.1.9	Provider in Komponenten registrieren	142
8.1.10	Den BookMonkey erweitern	143
8.2	Routing: durch die Anwendung navigieren	147
8.2.1	Routen konfigurieren	149
8.2.2	Routing-Modul einbauen	149
8.2.3	Komponenten anzeigen	152
8.2.4	Root-Route	153
8.2.5	Routen verlinken	154
8.2.6	Routenparameter	156
8.2.7	Verschachtelung von Routen	158
8.2.8	Routenweiterleitung	161
8.2.9	Wildcard-Route	162
8.2.10	Aktive Links stylen	162
8.2.11	Route programmatisch wechseln	163
8.2.12	Den BookMonkey erweitern	165
9	Powertipp: Chrome Developer Tools	177
10	HTTP & reaktive Programmierung: Iteration III	189
10.1	HTTP-Kommunikation: ein Server-Backend anbinden	189
10.1.1	Modul einbinden	191
10.1.2	Requests mit dem HttpClient durchführen	191
10.1.3	Optionen für den HttpClient	193
10.1.4	Den BookMonkey erweitern	196
10.2	Reaktive Programmierung mit RxJS	206
10.2.1	Alles ist ein Datenstrom	207
10.2.2	Observables sind Funktionen	208
10.2.3	Das Observable aus RxJS	211
10.2.4	Observables abonnieren	212
10.2.5	Observables erzeugen	214
10.2.6	Operatoren: Datenströme modellieren	217
10.2.7	Heiße Observables, Multicasting und Subjects	222
10.2.8	Subscriptions verwalten & Memory Leaks vermeiden	227
10.2.9	Flattening-Strategien für Higher-Order Observables .	231
10.2.10	Den BookMonkey erweitern: Daten vom Server typisieren und umwandeln	237
10.2.11	Den BookMonkey erweitern: Fehlerbehandlung	242
10.2.12	Den BookMonkey erweitern: Typeahead-Suche	246
10.3	Interceptoren: HTTP-Requests abfangen und transformieren .	257
10.3.1	Warum HTTP-Interceptoren nutzen?	257

10.3.2	Funktionsweise der Interceptoren	257
10.3.3	Interceptoren anlegen	258
10.3.4	Interceptoren einbinden	260
10.3.5	OAuth 2 und OpenID Connect	262
10.3.6	Den BookMonkey erweitern	265
11	Powertipp: Komponenten untersuchen mit Augury	271
12	Formularverarbeitung & Validierung: Iteration IV	275
12.1	Angulars Ansätze für Formulare	276
12.2	Template-Driven Forms	276
12.2.1	FormsModule einbinden	277
12.2.2	Datenmodell in der Komponente	277
12.2.3	Template mit Two-Way Binding und ngModel	278
12.2.4	Formularzustand verarbeiten	279
12.2.5	Eingaben validieren	280
12.2.6	Formular abschicken	281
12.2.7	Formular zurücksetzen	282
12.2.8	Den BookMonkey erweitern	284
12.3	Reactive Forms	303
12.3.1	Warum ein zweiter Ansatz für Formulare?	303
12.3.2	Modul einbinden	304
12.3.3	Formularmodell in der Komponente	304
12.3.4	Template mit dem Modell verknüpfen	307
12.3.5	Formularzustand verarbeiten	309
12.3.6	Eingebaute Validatoren nutzen	310
12.3.7	Formular abschicken	311
12.3.8	Formular zurücksetzen	312
12.3.9	Formularwerte setzen	312
12.3.10	FormBuilder verwenden	313
12.3.11	Änderungen überwachen	314
12.3.12	Den BookMonkey erweitern	315
12.4	Eigene Validatoren entwickeln	335
12.4.1	Validator für einzelne Formularfelder	335
12.4.2	Validator für Formulargruppen und -Arrays	338
12.4.3	Asynchrone Validatoren	340
12.4.4	Den BookMonkey erweitern	343
12.5	Welcher Ansatz ist der richtige?	351
13	Pipes & Direktiven: Iteration V	353
13.1	Pipes: Daten im Template formatieren	353
13.1.1	Pipes verwenden	353
13.1.2	Die Sprache fest einstellen	354

13.1.3	Eingebaute Pipes für den sofortigen Einsatz	356
13.1.4	Eigene Pipes entwickeln	367
13.1.5	Pipes in Komponenten nutzen	370
13.1.6	Den BookMonkey erweitern: Datum formatieren mit der DatePipe	371
13.1.7	Den BookMonkey erweitern: Observable mit der AsyncPipe auflösen	373
13.1.8	Den BookMonkey erweitern: eigene Pipe für die ISBN implementieren	376
13.2	Direktiven: das Vokabular von HTML erweitern	380
13.2.1	Was sind Direktiven?	380
13.2.2	Eigene Direktiven entwickeln	381
13.2.3	Attributdirektiven	383
13.2.4	Strukturdirektiven	388
13.2.5	Den BookMonkey erweitern: Attributdirektive für vergrößerte Darstellung	393
13.2.6	Den BookMonkey erweitern: Strukturdirektive für zeitverzögerte Sterne	396
14	Module & fortgeschrittenes Routing: Iteration VI	401
14.1	Die Anwendung modularisieren: das Modulkonzept von Angular	401
14.1.1	Module in Angular	401
14.1.2	Grundaufbau eines Moduls	402
14.1.3	Bestandteile eines Moduls deklarieren	403
14.1.4	Anwendung in Feature-Module aufteilen	405
14.1.5	Aus Modulen exportieren: Shared Module	408
14.1.6	Den BookMonkey erweitern	409
14.2	Lazy Loading: Angular-Module asynchron laden	419
14.2.1	Warum Module asynchron laden?	420
14.2.2	Lazy Loading verwenden	420
14.2.3	Module asynchron vorladen: Preloading	424
14.2.4	Den BookMonkey erweitern	425
14.3	Guards: Routen absichern	430
14.3.1	Grundlagen zu Guards	431
14.3.2	Guards implementieren	432
14.3.3	Guards verwenden	435
14.3.4	Den BookMonkey erweitern	435
14.4	Routing: Wie geht's weiter?	441
14.4.1	Resolver: asynchrone Daten beim Routing vorladen .	441
14.4.2	Mehrere Router-Outlets verwenden	445
14.4.3	Erweiterte Konfigurationen für den Router	446

15	Internationalisierung: Iteration VII	449
15.1	i18n: mehrere Sprachen und Kulturen anbieten	449
15.1.1	Was bedeutet Internationalisierung?	449
15.1.2	Eingebaute Pipes mehrsprachig verwenden	450
15.1.3	Texte übersetzen: Vorgehen in fünf Schritten	451
15.1.4	@angular/localize hinzufügen	452
15.1.5	Nachrichten im HTML mit dem i18n-Attribut markieren	452
15.1.6	Nachrichten im TypeScript-Code mit \$localize markieren	453
15.1.7	Nachrichten extrahieren und übersetzen	453
15.1.8	Feste IDs vergeben	454
15.1.9	Die App mit Übersetzungen bauen	455
15.1.10	Übersetzte Apps mit unterschiedlichen Konfigurationen bauen	459
15.1.11	Ausblick: Übersetzungen dynamisch zur Startzeit bereitstellen	466
15.1.12	Den BookMonkey erweitern	469
16	Powertipp: POEditor	477
17	Qualität fördern mit Softwaretests	483
17.1	Softwaretests	483
17.1.1	Testabdeckung: Was sollte man testen?	484
17.1.2	Testart: Wie sollte man testen?	486
17.1.3	Test-Framework Jasmine	487
17.1.4	»Arrange, Act, Assert« mit Jasmine	490
17.1.5	Test-Runner Karma	492
17.1.6	E2E-Test-Runner Protractor	492
17.1.7	Weitere Frameworks	494
17.2	Unit- und Integrationstests mit Karma	495
17.2.1	TestBed: die Testbibliothek von Angular	495
17.2.2	Isolierte Unit-Tests: Services testen	496
17.2.3	Isolierte Unit-Tests: Pipes testen	498
17.2.4	Isolierte Unit-Tests: Komponenten testen	500
17.2.5	Shallow Unit-Tests: einzelne Komponenten testen	503
17.2.6	Integrationstests: mehrere Komponenten testen	506
17.2.7	Abhängigkeiten durch Stubs ersetzen	508
17.2.8	Abhängigkeiten durch Mocks ersetzen	513
17.2.9	Leere Komponenten als Stubs oder Mocks einsetzen	515
17.2.10	HTTP-Requests testen	516
17.2.11	Komponenten mit Routen testen	520
17.2.12	Asynchronen Code testen	524

17.3	Oberflächentests mit Protractor	526
17.3.1	Protractor verwenden	527
17.3.2	Elemente selektieren: Locators	528
17.3.3	Aktionen durchführen	529
17.3.4	Asynchron mit Warteschlange	530
17.3.5	Redundanz durch Page Objects vermeiden	531
17.3.6	Eine Angular-Anwendung testen	532

IV Das Projekt ausliefern: Deployment 537

18	Build und Deployment mit der Angular CLI	539
18.1	Build-Konfiguration	540
18.2	Build erzeugen	542
18.3	Umgebungen konfigurieren	544
18.3.1	Abhängigkeit zur Umgebung vermeiden	547
18.3.2	Konfigurationen und Umgebungen am Beispiel: BookMonkey	548
18.4	Produktivmodus aktivieren	550
18.5	Die Templates kompilieren	551
18.5.1	Ahead-of-Time-Kompilierung (AOT)	552
18.5.2	Just-in-Time-Kompilierung (JIT)	552
18.6	Bundles analysieren mit source-map-explorer	554
18.7	Webserver konfigurieren und die Anwendung ausliefern	555
18.7.1	ng deploy: Deployment mit der Angular CLI	558
18.7.2	Ausblick: Deployment mit einem Build-Service	560
19	Angular-Anwendungen mit Docker bereitstellen	563
19.1	Docker	564
19.2	Docker Registry	565
19.3	Lösungsskizze	565
19.4	Eine Angular-App über Docker bereitstellen	566
19.5	Build Once, Run Anywhere: Konfiguration über Docker verwalten	571
19.6	Multi-Stage Builds	577
19.7	Grenzen der vorgestellten Lösung	582
19.8	Fazit	583

V	Fortgeschrittene Themen	585
20	Server-Side Rendering mit Angular Universal	587
20.1	Single-Page-Anwendungen, Suchmaschinen und Start-Performance	588
20.2	Dynamisches Server-Side Rendering	591
20.3	Statisches Pre-Rendering	597
20.4	Hinter den Kulissen von Angular Universal	599
20.5	Browser oder Server? Die Plattform bestimmen	601
20.6	Routen ausschließen	602
20.7	Wann setze ich serverseitiges Rendering ein?	603
20.8	Ausblick: Pre-Rendering mit Scully	605
21	State Management mit Redux und NgRx	607
21.1	Ein Modell für zentrales State Management	608
21.2	Das Architekturmodell Redux	619
21.3	NgRx: Reactive Extensions for Angular	621
21.3.1	Projekt vorbereiten	621
21.3.2	Store einrichten	622
21.3.3	Schematics nutzen	622
21.3.4	Grundstruktur	622
21.3.5	Feature anlegen	624
21.3.6	Struktur des Feature-States definieren	626
21.3.7	Actions: Kommunikation mit dem Store	627
21.3.8	Dispatch: Actions in den Store senden	629
21.3.9	Reducer: den State aktualisieren	630
21.3.10	Selektoren: Daten aus dem State lesen	634
21.3.11	Effects: Seiteneffekte ausführen	639
21.4	Redux und NgRx: Wie geht's weiter?	644
21.4.1	Routing	644
21.4.2	Entity Management	645
21.4.3	Testing	647
21.4.4	Hilfsmittel für Komponenten: @ngrx/component	656
21.5	Ausblick: Lokaler State mit RxAngular	659
22	Powertipp: Redux DevTools	663

VI Angular-Anwendungen für Mobilgeräte 667

23 Der Begriff App und die verschiedenen Arten von Apps .. 669

- 23.1 Plattformspezifische Apps 669
- 23.2 Apps nach ihrer Umsetzungsart 670

24 Progressive Web Apps (PWA) 673

- 24.1 Die Charakteristiken einer PWA 673
- 24.2 Service Worker 674
- 24.3 Eine bestehende Angular-Anwendung in eine PWA
verwandeln 674
- 24.4 Add to Homescreen 676
- 24.5 Offline-Funktionalität 680
- 24.6 Push-Benachrichtigungen 685

25 NativeScript: mobile Anwendungen entwickeln 695

- 25.1 Mobile Apps entwickeln 695
- 25.2 Was ist NativeScript? 696
- 25.3 Warum NativeScript? 696
- 25.4 Hinter den Kulissen 698
- 25.5 Plattformspezifischer Code 699
- 25.6 Komponenten und Layouts 701
- 25.7 Styling 702
- 25.8 NativeScript und Angular 703
- 25.9 Angular als Native App 704
- 25.10 NativeScript installieren 705
- 25.11 Ein Shared Project erstellen mit der Angular CLI 705
- 25.12 Den BookMonkey mit NativeScript umsetzen 709
 - 25.12.1 Das Projekt mit den NativeScript Schematics
erweitern 709
 - 25.12.2 Die Anwendung starten 709
 - 25.12.3 Das angepasste Bootstrapping für NativeScript 713
 - 25.12.4 Das Root-Modul anpassen 713
 - 25.12.5 Das Routing anpassen 716
 - 25.12.6 Die Templates der Komponenten für NativeScript
anlegen 717

26 Powertipp: Android-Emulator Genymotion 729

VII Weiterführende Themen 733

27 Fortgeschrittene Konzepte der Angular CLI	735
27.1 Workspace und Monorepo: Heimat für Apps und Bibliotheken	735
27.1.1 Applikationen: Angular-Apps im Workspace	736
27.1.2 Bibliotheken: Code zwischen Anwendungen teilen ..	738
27.2 Schematics: Codegenerierung mit der Angular CLI	740
28 Wissenswertes	743
28.1 Web Components mit Angular Elements	743
28.2 Container und Presentational Components	751
28.3 Else-Block für die Direktive ngIf	755
28.4 TrackBy-Funktion für die Direktive ngFor	756
28.5 Angular-Anwendungen dokumentieren und analysieren	758
28.6 Angular Material und weitere UI-Komponentensammlungen	762
28.7 Content Projection: Inhalt des Host-Elements verwenden	765
28.8 Lifecycle-Hooks	766
28.9 Change Detection	770
28.10 Plattformen und Renderer	784
28.11 Angular updaten	785
28.12 Upgrade von AngularJS	788

VIII Anhang 795

A Befehle der Angular CLI	797
B Operatoren von RxJS	805
C Matcher von Jasmine	809
D Abkürzungsverzeichnis	813
E Linkliste	815
Index	825
Weiterführende Literatur	835
Nachwort	837