

Angular

Das Praxisbuch – von den Grundlagen
bis zur professionellen Entwicklung
mit Signals

DAS INHALTS- VERZEICHNIS



» Hier geht's
direkt
zum Buch

Inhaltsverzeichnis

Vorwort 13

I Einführung 19

1 Benötigte Werkzeuge: Editor, Node.js und Co. 21

1.1 Konsole, Terminal und Shell 21

1.2 Paketverwaltung mit Node.js und NPM 21

1.3 Visual Studio Code 25

1.4 Google Chrome 26

1.5 Angular Developer Tools 27

1.6 Codebeispiele in diesem Buch 27

2 Angular CLI: der Codegenerator für unser Projekt 29

2.1 Das offizielle Tool für Angular 29

2.2 Installation 30

2.3 Autovervollständigung einrichten 31

3 Syntax & Konzepte: Grundlagen für modernes Angular 33

3.1 TypeScript 33

3.2 Template Strings 34

3.3 Arrow Functions 34

3.4 Immutability 36

3.5 Spread-Syntax und Rest-Parameter 37

3.6 Private Class Fields 39

3.7 Property Modifiers: readonly und protected 40

3.8 Decorators 41

3.9 Generic Types 42

3.10 Promises und async/await 43

II	BookManager: Schritt für Schritt zur App	45
4	Projektvorstellung und Einrichtung	47
4.1	Unser Projekt: BookManager	47
4.2	Struktur und Quellcode	48
4.3	Projekt mit der Angular CLI initialisieren	50
4.4	Aufbau des neuen Projekts	51
4.5	Das Projekt starten	56
4.6	Softwaretests ausführen	57
4.7	Beispiel-Template entfernen	57
4.8	Globale Styles einbinden: @angular-buch/styles	59
5	Komponenten und Signals: die Grundbausteine	61
5.1	Komponenten	61
5.2	Das Template einer Komponente	62
5.3	Komponenten in der Anwendung verwenden	63
5.4	Komponenten generieren mit der Angular CLI	65
5.5	Reaktivität mit Signals	66
5.6	Template-Syntax	68
5.7	Der Style einer Komponente	77
6	BookManager: Buchliste anzeigen	81
6.1	Praktische Umsetzung	81
6.2	Unit- und Integrationstests mit Vitest	90
6.3	Komponenten testen	93
7	Property Bindings und Component Inputs	97
7.1	Property Bindings für native Elemente	97
7.2	Spezielle Property Bindings	98
7.3	Wiederholung: Komponenten verschachteln	100
7.4	Datenfluss von Eltern- zu Kindkomponenten	101
7.5	Syntax für Property Bindings	103
7.6	Lebenszyklus von Komponenten	104
8	BookManager: Komponenten aufteilen	107
8.1	Praktische Umsetzung	107
8.2	Verschachtelte Komponenten testen	112
9	Event Bindings und Component Outputs	117
9.1	Native Events aus dem DOM	118
9.2	Eigene Events aus Komponenten	120

10	BookManager: Favoritenliste erstellen	125
10.1	Praktische Umsetzung	125
10.2	Komponenten mit Outputs testen	130
11	Dependency Injection: Code in Services auslagern	135
11.1	Services in Angular	135
11.2	Abhängigkeiten anfordern mit <code>inject()</code>	137
11.3	Providers: Abhängigkeiten registrieren	138
11.3.1	Tree-Shakable Providers: Abhängigkeiten automatisch registrieren	138
11.3.2	Manuelle Providers: Abhängigkeiten explizit registrieren	139
11.3.3	Aufrufreihenfolge	140
11.4	Abhängigkeiten ersetzen	140
11.5	Eigene Tokens definieren mit <code>InjectionToken</code>	142
12	BookManager: Services nutzen	145
12.1	Praktische Umsetzung	145
12.2	Services testen	148
13	Fortgeschrittene Konzepte für Signals	151
13.1	Reactive Context: Wann ändert sich ein Signal?	151
13.2	Effect: auf Änderungen reagieren	152
13.3	Computed Signal: Werte berechnen	153
13.4	Linked Signal: schreibbares Signal mit Berechnung	154
13.5	Model Signal: Kombination von Input und Output	156
13.6	<code>untracked()</code> : den Reactive Context verlassen	158
14	BookManager: Bücher lokal suchen	161
14.1	Praktische Umsetzung	161
14.2	Computed Signals testen	165
15	Routing: durch die Anwendung navigieren	167
15.1	Routen konfigurieren	168
15.2	Routen einbinden: die Datei <code>app.routes.ts</code>	169
15.3	Routing in Features verwenden	170
15.4	Komponenten anzeigen	171
15.5	Root-Route	173
15.6	Weiterleitung auf eine andere Route	173
15.7	Wildcard-Route	174
15.8	Links setzen	174
15.9	Routenparameter auslesen	176
15.10	Aktive Links stylen	179
15.11	Route programmatisch wechseln	180

15.12	Seitentitel setzen	181
15.13	Pfade in Single-Page-Applikationen	184
16	BookManager: Routing	187
16.1	Praktische Umsetzung	187
16.2	Komponenten mit Routing testen	198
17	Routing: Component Input Binding	207
17.1	Inputs setzen mit dem Router	208
17.2	Inputs transformieren	209
18	BookManager: Component Input Binding	213
18.1	Praktische Umsetzung	213
18.2	Routenparameter als Inputs testen	215
19	HTTP-Kommunikation: ein Server-Backend anbinden	217
19.1	Daten per HTTP abrufen	217
19.2	HTTP mit der Fetch API	218
19.3	Der HttpClient von Angular	219
19.4	HttpClient global konfigurieren	223
19.5	Optionen für den HttpClient	224
19.6	Ausblick: Codegenerierung mit OpenAPI	227
20	BookManager: Daten per HTTP abrufen	229
20.1	Praktische Umsetzung	229
20.2	HTTP-Aufrufe mocken und testen	238
21	Daten laden mit der Resource API	249
21.1	Die Resource API	250
21.2	Auf die Daten zugreifen	251
21.3	Status verarbeiten	252
21.4	Daten neu laden	253
21.5	Werte lokal überschreiben	253
21.6	Loader mit Parameter	254
21.7	rxResource: Resource mit Observables	255
21.8	httpResource: Resource für HTTP-Requests	257
21.9	Abgrenzung: Resource API vs HttpClient	258
21.10	Diskussion zur Architektur	258
22	BookManager: HTTP mit der Resource API	261
22.1	Praktische Umsetzung: Buchliste	261
22.2	Praktische Umsetzung: Detailseite	266
22.3	HTTP-Resources testen	269

23	Pipes: Daten im Template transformieren	281
23.1	Pipes verwenden	281
23.2	Eingebaute Pipes für den sofortigen Einsatz	282
23.3	Eigene Pipes entwickeln	290
24	BookManager: Pipes verwenden	293
24.1	Praktische Umsetzung: Datum formatiert anzeigen	293
24.2	Praktische Umsetzung: ISBN formatieren	294
24.3	Pipes testen	297
25	Formularverarbeitung mit Signal Forms	301
25.1	Angulars Ansätze für Formulare	301
25.2	Datenmodell und Feldstruktur	303
25.3	Formularmodell mit dem Template verknüpfen	306
25.4	Werte und Zustände verarbeiten	307
25.5	Schema für Validierung und Feldlogik	309
25.5.1	Eingebaute Validatoren	310
25.5.2	Zustände anzeigen	311
25.5.3	Fehlernachrichten anzeigen	312
25.5.4	FieldContext: Feldinformationen für die Validierung	313
25.5.5	Eigene Validierungsregeln	314
25.5.6	Asynchrone Validierung	315
25.5.7	Formularänderungen entprellen	316
25.5.8	Verfügbarkeit von Feldern steuern	317
25.6	Schema-Komposition	318
25.6.1	apply(): Schemas zusammenfügen	319
25.6.2	applyEach(): Listen validieren	319
25.6.3	applyWhen(): bedingte Validierung	320
25.7	Formular absenden	321
25.7.1	Absendelogik definieren	321
25.7.2	Der manuelle Weg: die Funktion submit()	323
25.7.3	Der elegante Weg: die Direktive FormRoot	324
25.7.4	Fehlermeldungen vom Server anzeigen	325
25.7.5	Verhalten beim Absenden konfigurieren	326
25.7.6	Formular zurücksetzen mit reset()	327
25.7.7	Best Practices zur Barrierefreiheit	327
25.8	CSS-Klassen automatisch setzen	328
26	BookManager: Buchdaten im Formular erfassen	333
26.1	Praktische Umsetzung: Buchdaten erfassen	333
26.2	Praktische Umsetzung: Formulardaten speichern	340
26.3	Signal Forms testen	345

27	BookManager: Formulareingaben validieren	351
27.1	Praktische Umsetzung	351
27.2	Formularvalidierung testen	359
28	BookManager: Suche auf dem Server	365
28.1	Praktische Umsetzung	365
28.2	Query-Parameter testen	372
29	Lazy Loading	379
29.1	Lazy Loading mit dem Router	381
29.1.1	loadComponent: Komponenten asynchron laden	382
29.1.2	loadChildren: Features asynchron laden	383
29.1.3	Preloading: Routen asynchron vorladen	386
29.2	Lazy Loading mit Deferrable Views	387
29.3	Abgrenzung: Lazy Loading vs Deferrable Views	391
30	BookManager: Lazy Loading implementieren	395
30.1	Praktische Umsetzung	395
30.2	Hinweise zum Testing	399
31	Reaktive Programmierung mit RxJS	401
31.1	Alles ist ein Datenstrom	401
31.2	Observables sind Funktionen	403
31.3	Das Observable aus RxJS	406
31.4	Observables abonnieren	407
31.5	Grundbegriffe	409
31.6	Observables erzeugen	409
31.7	Observables und Promises	412
31.8	Operatoren: Datenströme modellieren	413
31.9	Heiße Observables, Multicasting und Subjects	416
31.10	Subscriptions verwalten & Memory Leaks vermeiden	422
31.11	Observables subscriben mit der AsyncPipe	427
31.12	Observables und Signals: toSignal() und toObservable()	428
31.13	Fehler behandeln	431
31.14	Flattening-Strategien für Higher-Order Observables	433
32	BookManager: Typeahead-Suche	443
32.1	Praktische Umsetzung	443
32.2	RxJS-Pipelines testen	453

III	Wissenswertes	459
33	Barrierefreiheit (a11y)	461
33.1	Gesetze und Standards	461
33.2	Semantic HTML	463
33.3	ARIA-Attribute	465
33.4	Host Bindings	467
33.5	Routing	468
33.6	Formularverarbeitung	473
33.7	Angular Aria: barrierefreie Direktiven	475
33.8	Angular CDK	476
34	AI-Unterstützung für Angular	479
34.1	Herausforderung: veraltetes Wissen	480
34.2	AI-Konfigurationsdateien	481
34.3	Herausforderung: das Kontextfenster	482
34.4	Der MCP-Server von Angular	483
34.5	Empfehlungen für die Praxis	485
35	Softwaretests	489
35.1	Testarten: Wie sollte man testen?	489
35.2	Unit- und Integrationstests mit Vitest	491
35.2.1	Tests starten	491
35.2.2	Der Aufbau eines Tests	493
35.2.3	Testdoubles	495
35.2.4	Test Pollution vermeiden	497
35.2.5	Fake Timers	499
35.3	Testing mit Angular: das TestBed	501
35.3.1	Services testen	501
35.3.2	Komponenten testen mit ComponentFixture	502
35.3.3	Auf Aktualisierungen warten	503
35.3.4	Kindkomponenten mocken mit overrideComponent()	503
35.3.5	Weitere Testing-APIs von Angular	504
35.4	E2E-Tests	505
36	Deployment	507
36.1	Den Build konfigurieren (angular.json)	507
36.1.1	Application Builder	509
36.1.2	Konfigurationen	510

- 36.2 Build ausführen 511
 - 36.2.1 Spezifische Konfigurationen beim Build laden 512
 - 36.2.2 Bundles..... 512
 - 36.2.3 Budgets konfigurieren 514
 - 36.2.4 Basispfad setzen 515
 - 36.2.5 Erfolgreichen Build testen 515
- 36.3 Umgebungen konfigurieren mit File Replacements 516
- 36.4 InjectionTokens: direkte Abhängigkeiten zur Umgebung vermeiden 518
- 36.5 Produktive Anwendung ausliefern 519
- 36.6 Automatisiertes Deployment (CI/CD) 522
- 37 Bildoptimierung mit NgOptimizedImage 525**
 - 37.1 Wichtige Bilder priorisieren 526
 - 37.2 Bildgrößen handhaben 527
- 38 Angular aktualisieren 529**
 - 38.1 ng update: Update mit der Angular CLI 530
 - 38.2 Automatisches Patch-Management 531
 - 38.3 Angular Update Guide 531
- Nachwort 533**
- IV Anhang 535**
- A Abkürzungsverzeichnis 537**
- B Linkliste 539**
- API- und Sprachreferenz 543**
- Index 549**