

# Zwei Jahre Angular. In Produktion.



JAX 2018

# Karsten Sitterberg

Trainer  
Autor  
Entwickler



<https://sitterberg.com/>  
[karsten@sitterberg.com](mailto:karsten@sitterberg.com)  
[@kakulty](#)

**Typisch(e) Kunden**

# Enterprise Anwendungen

- Komplexe Anwendungen
  - Erstellung im Bereich > xx PJ
- Lange Lebensdauer erfordert gute Wartbarkeit
  - Firmen sind bereit zu investieren
- Entwickler
  - Viele Entwickler(teams)
  - Unterschiedliches Know-How
  - Produktivität steht im Vordergrund
  - Fachentwickler vs. Infrastrukturentwickler

# Angular richtige Wahl für Enterprise

- Opinionated + Full-Stack
  - Liefert vorgegebene Lösungen: Routing, Formulare, HTTP, DI
- Ökosystem
  - CLI, NgRx, Angular Material, PWA, Universal, ...
- TypeScript für langlebige Anwendungen
  - Statische Analysierbarkeit
  - Verbessert Refactoring-Möglichkeiten
  - Produktivität: IDE Support (Linter, Code Qualität)

Deutsche Bank

Osram

Citibank

Audi

Lufthansa Systems

Microsoft

Otto

Bosch

PwC

Avarto

Deutsche Post

Terradata

Provinzial

Barnes & Noble

RWE

LVM

# Angular richtige Wahl für Enterprise



Firebase



Google Analytics



Google Express

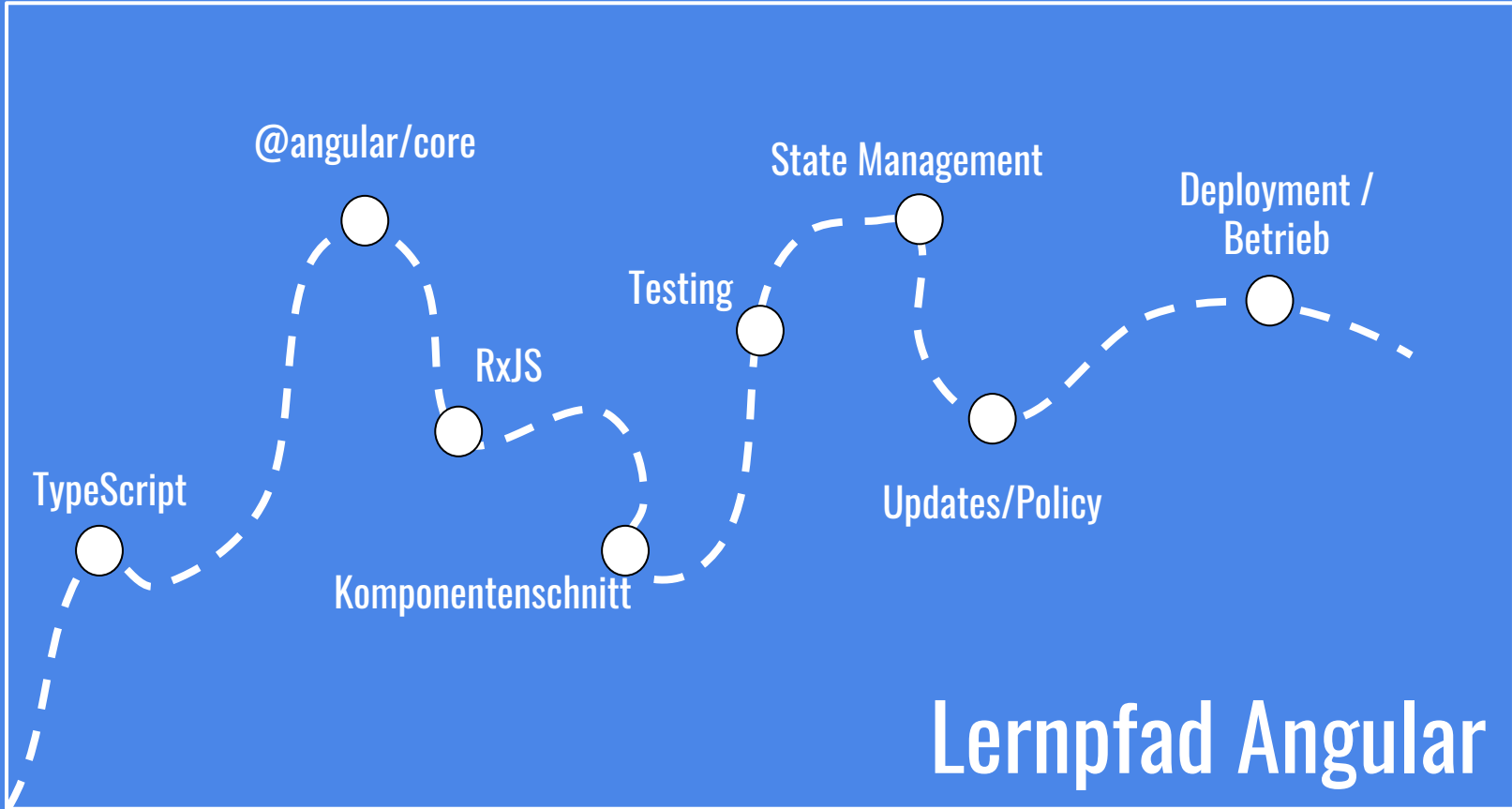


Google Cloud Platform

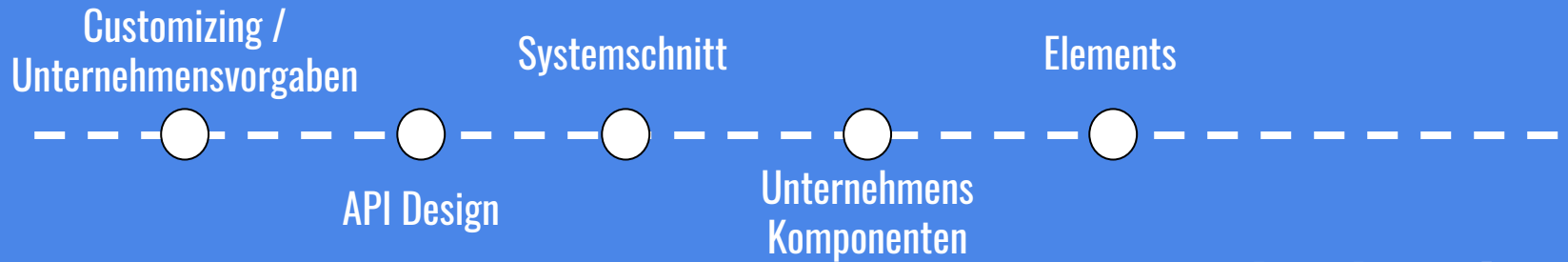
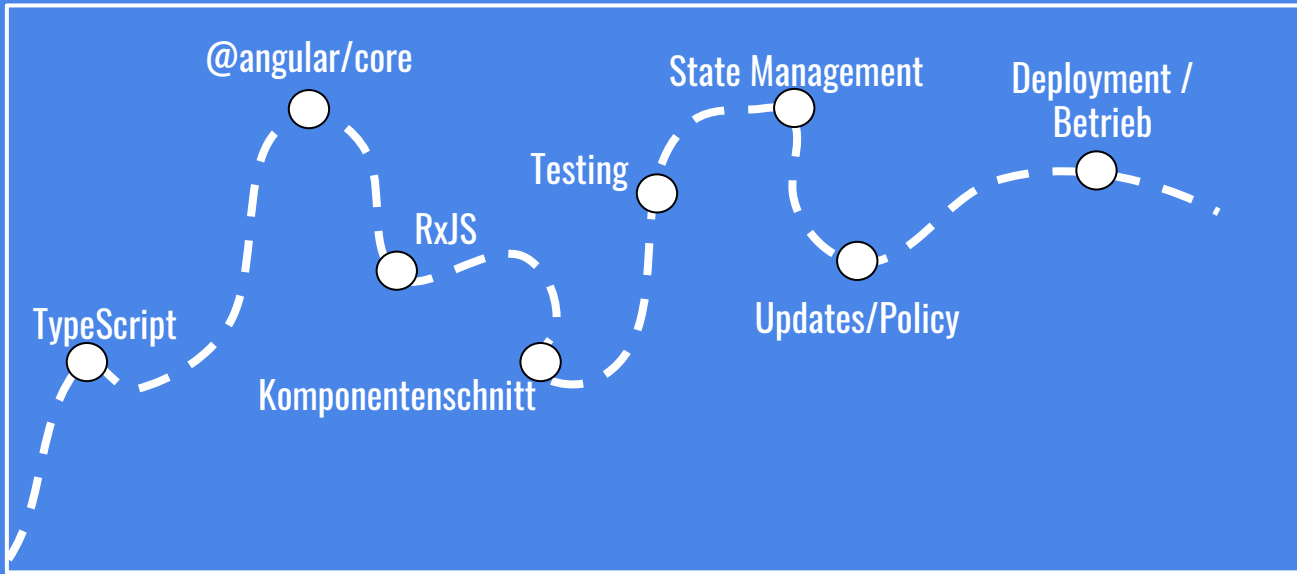
...



+600 more



# Lernpfad Angular



# Architektur



# Lessons Learned



**Beim Standard bleiben**

# Mit dem Standard beginnen

- Verwendung von Standard Tools
  - Angular CLI
  - Schematics
  - tslint/codelyzer
- Standardeinstellungen
  - Erst Verstehen
  - Dann Customizing

vs.

- Auf gewohnten Mustern bestehen
- Angular verbiegen

# Am Standard wachsen

- Best Practises beibehalten
- Firmeninterne Konventionen
  - Eigene Linter-Regeln
  - Eigene Schematics
  - Evtl. eigene Komponenten-Bibliotheken
- Nutzen
  - Interne Unterstützung (Inter-Team-Fähigkeit, Code Reviews)
  - Externe Unterstützung (Code Reviews, Dienstleistung)

Customizing /  
Unternehmensvorgaben

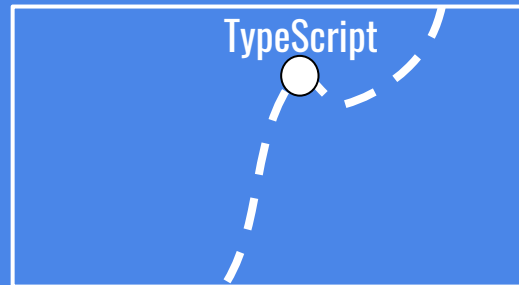


TS app.module.ts ✕

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppRoutingModule } from './app-routing.module';
5
6  import { AppComponent } from './app.component';
7
8
9  @NgModule({
10   declarations: [
11     AppComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
21
```

# Embrace TypeScript

# TypeScript

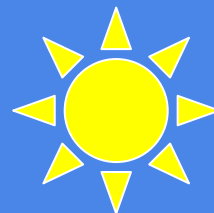


- “JavaScript that scales”
- Statische Typisierung - Statische Codeanalyse
  - Durch Compiler **und** Linter
- Sollte möglichst voll ausgeschöpft werden



Rückgabetypen immer  
angeben (**void**)

~~**var**, **any**~~



# TypeScript

- Statische Analyse
  - IDE

```
1
2  /**
3   * A simple component providing an input field, an output area and a toggle
4   */
5  @Component({
6   selector: "my-app",
7   template: `
8     <h1>Hello Angular 2!</h1>
9     <div>
10      <label [hidden]="!isCalling">I am {{name}}.</label>
11      <div>
12        <input [(ngModel)]="name" />
13        <button (click)="toggleCalling()">toggle Calling</button>
14      </div>
15    </div>
16  `
17 })
18 export class SimpleComponent {
19   name = "here";
20   private isCalling = true;
21
22   private toggleCalling() {
23     this.isCalling = !this.isCalling;
24   }
25 }
26
```

# TypeScript

- Statische Analyse
  - IDE
  - Linter
  - SonarQube

The screenshot displays the SonarQube interface for a TypeScript project. The left sidebar shows the 'Measures' tab with a 'Coverage' section. The main area shows the code for 'angular / src/app / app.component.ts' with a coverage of 57.1% and 0 issues. The code is highlighted to show coverage status: green for covered lines, yellow for uncovered lines, and red for uncovered conditions.

**Project Overview**

- > Reliability ?
- > Security ?
- > Maintainability ?
- ▼ Coverage
  - Overview
  - On new code
    - Coverage 50.0%
    - Lines to Cover 8
    - Uncovered Lines 3
    - Line Coverage 62.5%
    - Conditions to Cover 4
    - Uncovered Conditions 3
    - Condition Coverage 25.0%
  - Overall
    - Coverage 57.1%
    - Lines to Cover 10
    - Uncovered Lines 3
    - Line Coverage 70.0%
    - Conditions to Cover 4
    - Uncovered Conditions 3
    - Condition Coverage 25.0%
- > Duplications

**angular / src/app / app.component.ts** ☆

Coverage on New Code 50.0% Leak Period: since previous version

angular 22 Lines 0 Issues 57.1% Coverage

```
1 | import { Component } from '@angular/core';
2 |
3 | @Component({
4 |   selector: 'app-root',
5 |   templateUrl: './app.component.html',
6 |   styleUrls: ['./app.component.css']
7 | })
8 | export class AppComponent {
9 |
10 |   foo = true;
11 |
12 |   get title() {
13 |     if (this.foo) {
14 |       return "app"
15 |     }
16 |     if(this.foo === false) {
17 |       return "demo";
18 |     }
19 |     return "bar";
20 |   }
21 | }
22 |
```



# TypeScript

- Refactoring

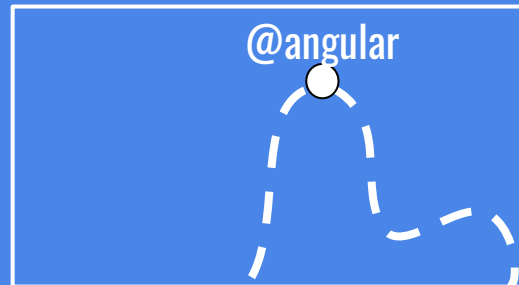
```
Departments.ts x
1  abstract class Department {
2
3      constructor(public name: string) {}
4
5
6      printName(): void { console.log("Department name: " + this.name); }
7
8  }
9
10
11  interface ReportingDepartment {
12      generateReports(): void
13  }
14
15  class AccountingDepartment extends Department implements ReportingDepartment {
16
17      constructor() { super("Accounting and Auditing"); }
18
19
20
21      printMeeting(): void { console.log("The Accounting Department meets each Monday at 10am."); }
22
23
24
25      generateReports(): void {
26          console.log("Generating accounting reports...");
27      }
28  }
29
30
31
32
33
```

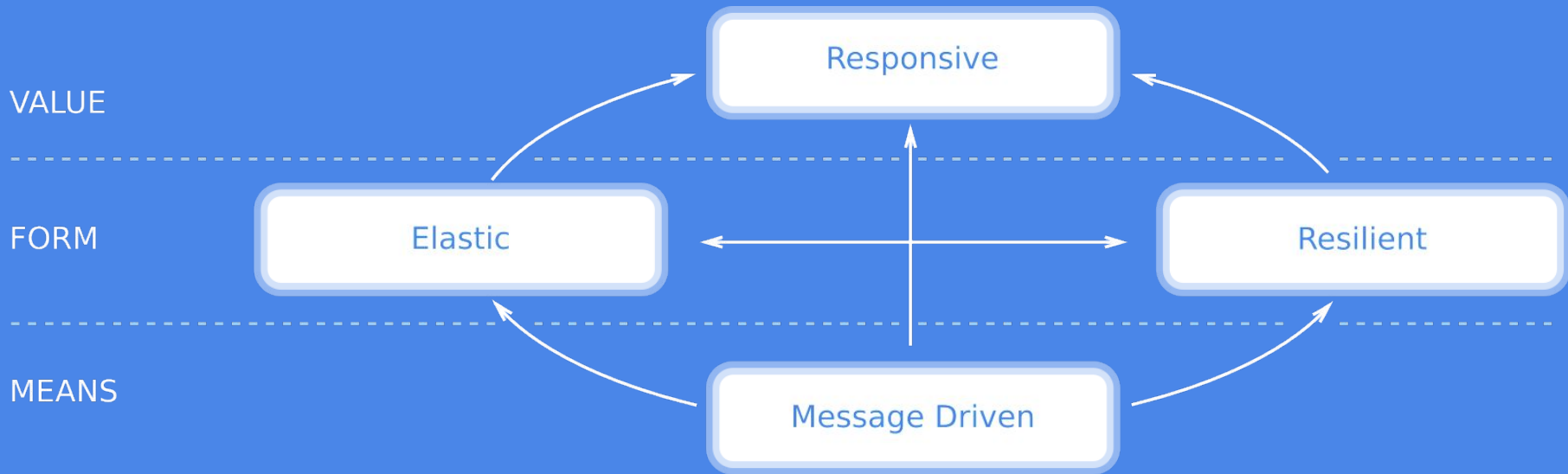


@angular

# @angular

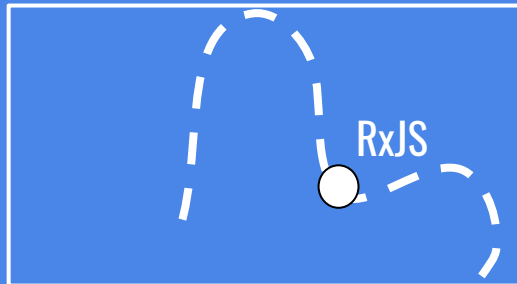
- **Vorsicht: Veraltete Dokumentationen/Tipps**
  - Datum beachten (Stackoverflow, ...)
  - Bücher altern schnell
- **Tutorials von angular.io nutzen**
- **Austausch mit Community**
  - User Groups, Meetups, Vorträge
- **Auf dem laufenden bleiben**
  - Aktuelle Magazine/Blogs





# Master RxJS

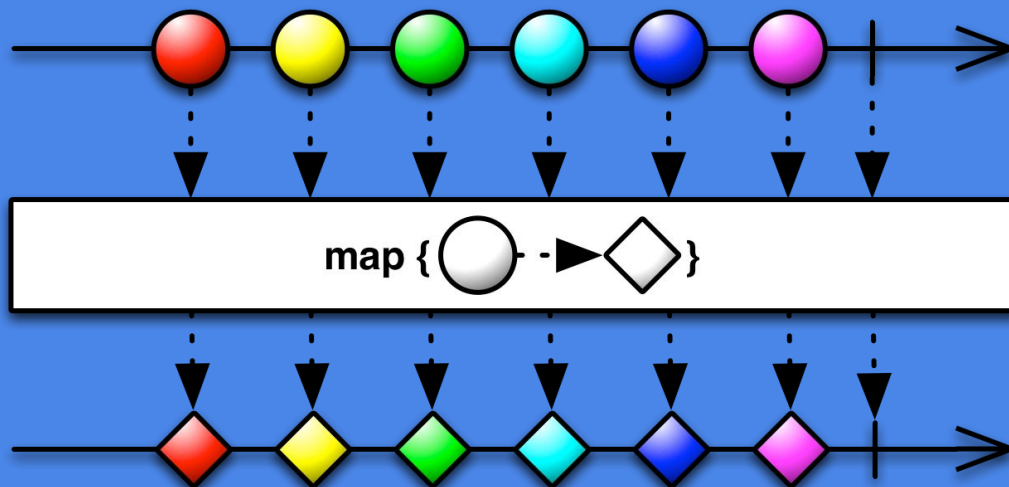
# RxJS einsetzen



- RxJS - Programmieren mit **asynchronen Datenströmen**
- Browser arbeitet asynchron
- Von Angular intern verwendet
  - HTTP
  - Router
  - Forms
  - (ngrx)

# RxJS lernen

- Verständnis: Funktionale Programmierung
  - “Filter-Map-Reduce”
- Rx-Operatoren kennenlernen
  - Marble-Diagramme



# RxJS einsetzen

- Von Angular bereitgestellte API annehmen
  - Observables **nicht** direkt auflösen
    - ~~“Subscribe und ab in die synchrone Welt”~~
    - Meist ist asynchrone Abhandlung sinnvoller
- **async**-Pipe!
  - Auflösen der Observables im Template
  - Angular erledigt subscribe/unsubscribe automatisch

# Rxjs Anti Patterns

- Unsubscribe-Leak
  - Unsubscribe vergessen - Memory Leaks
  - Abhilfe: **async-Pipe**, **ng2-rx-componentdestroyed**

```
@Component({ ... })
export class MyComponent implements OnInit, OnDestroy
{
  ngOnInit() {
    Observable.interval(1000)
      .pipe(untilComponentDestroyed(this) )
      .subscribe(console.log);
  }
  ngOnDestroy() {}
}
```



# Rxjs Anti Patterns

- **async**-Pipe - Kann zu mehrfachen (ungewollten) Requests führen, daher:
  - `share()`-Operator verwenden
  - `@ngrx` nutzen
  - `ngIfAs`-Syntax nutzen

```
<p *ngIf="userObservable | async as user">
  {{user.nachname}}, {{user.nachname}}
</p>
```

# Rxjs Anti Patterns

- Nested Subscriptions

```
this.http.get(`/api/user/42`)
  .map( user => user.id)
  .subscribe( id => {

    this.http.get(`/api/shoppingListForUser/{id}`)
      .subscribe(list => {
        this.shoppingList = list
      });

  });
```

# Rxjs Anti Patterns

- Nested Subscriptions

```
this.http.get(`/api/user/42`)  
  .map( user => user.id)  
  .subscribe( id => {  
  
    this.http.get(`/api/shoppingListForUser/${id}`)  
      .subscribe(list => {  
        this.shoppingList = list  
      });  
  
  });
```

# Rxjs Anti Patterns

- mergeMap statt Nested Subscriptions

```
this.http.get(`/api/user/42`)
  .map( user => user.id)
  .mergeMap( id => {
    return this.http.get(`/api/shoppingListForUser/${id}`)
  })
  .subscribe(list => {
    this.shoppingList = list
  });
```

# Rxjs Anti Patterns

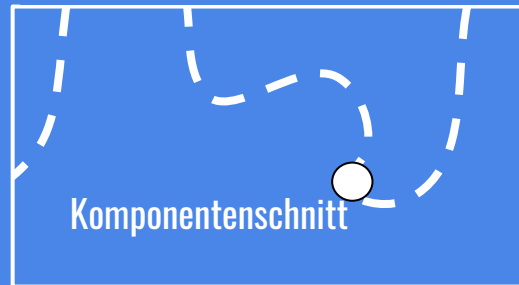
- Nicht Promise-API und Rx-API mischen
- Bei Fremdbibliotheken
  - Mappingschicht
  - `.toPromise()` / `.fromPromise()`

# Anwendungs Architektur



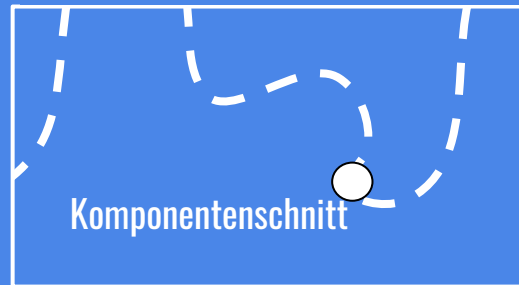
# Domain vs. View Components

- View Komponenten
  - User-Interaktionselemente  
(Dropdown, Datepicker, Modal, ...)
  - **Hohe** Wiederverwendbarkeit
  - Zustandslos
  - Auslagerung als Option
    - Eigene Komponentenbibliothek



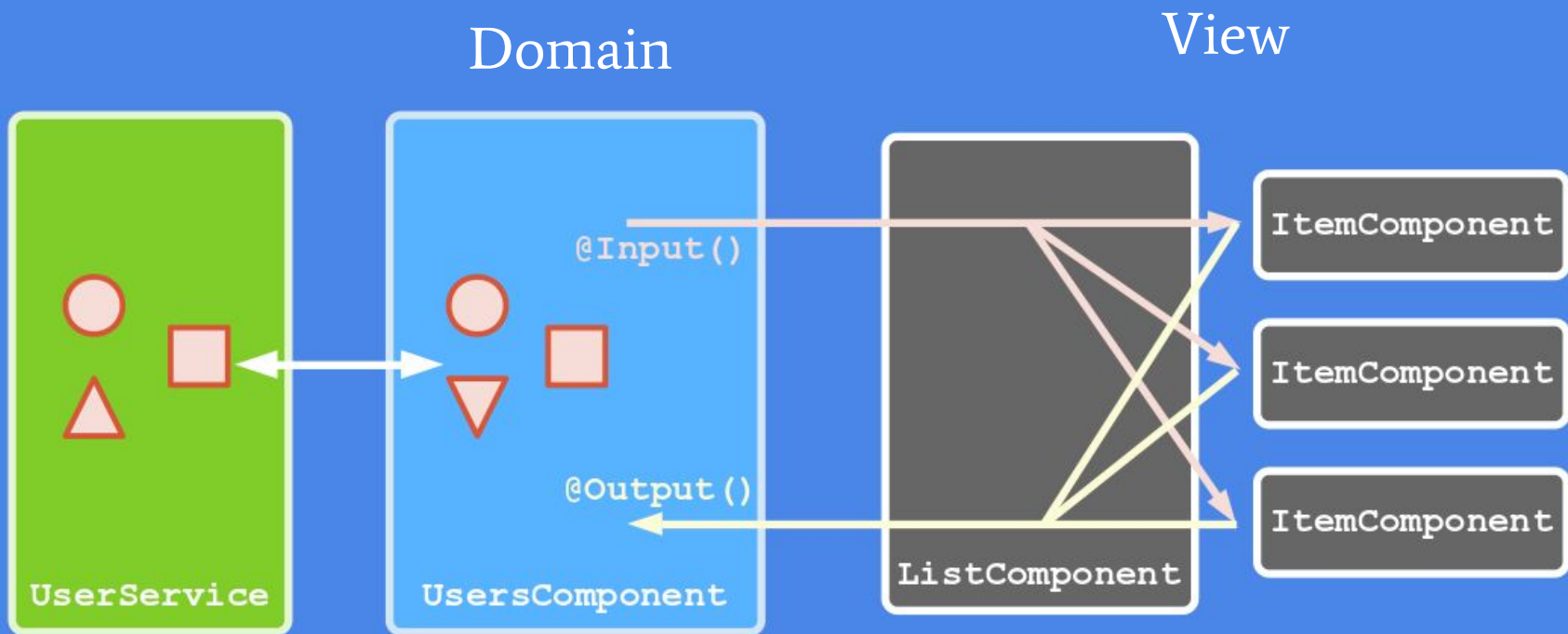
# Domain vs. View Components

- Domain Komponenten
  - Komplexere und spezifischere (fachliche) Logik
  - Orchestrieren View-Komponenten
  - Mappen Datenmodell für View-Komponenten
  - Anbindung an Store, fachliche Services etc.
  - Möglicherweise zustandsbehaftet





# Domain vs. View Components

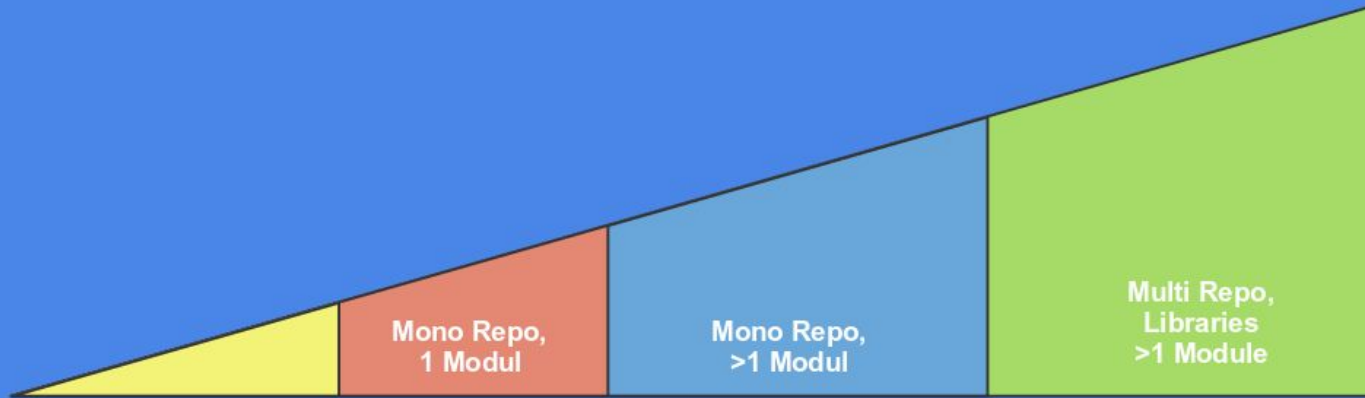


# Komponenten-Frameworks

- Einsatz fertiger Komponenten-Frameworks
  - Material, Bootstrap, ...
  - Kopplung an Framework bewerten
- Styling durch CSS
  - Nur View-Components stylen!
  - CSS Architektur ist Thema für sich ...

# Anwendungsschnitt

Systemschnitt



# Anwendungsschnitt

- Anwendung nach Features aufbauen
  - Sowohl *technische* als auch *fachliche* Features
- Kapselung der Features in Module (`@NgModule()`)
  - Lazy Loading und Code-Splitting
- Option: Separierung in mehrere Anwendungen
  - “Self Contained Systems”
- Bereitstellung von Modulen via NPM Artefakt-Repository

# Wiederverwendbare Module

- Vorgehen
  - Schnitt nach Einsatzzweck
  - Angular Package Format (APF) verwenden (momentan v6)
  - Wird für volle Unterstützung (etwa AoT) benötigt
- Tools, die APF-konforme Packages generieren
  - ng-packagr
  - Angular CLI ab Version 6

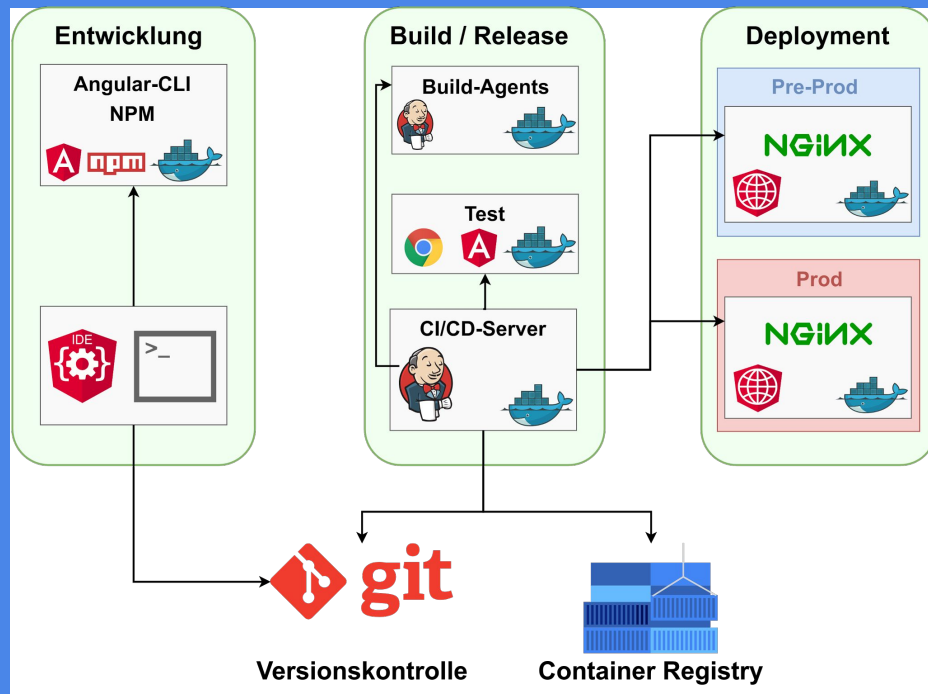
**Build/Deployment**

# Docker

- Für Build
  - Umgebung benötigt keine speziellen Werkzeuge
- Für Dev
  - Bereitstellung von Backend/Frontend Services
- Für Tests
  - Isolierter Test, kleinerer Test-Scope
  - Mit Docker eigener Datenbestand simpel
  - Reproduzierbare, stabile Tests
- Für Deployment
  - Atomar, rollbackfähig, schrittweiser Rollout

# Docker

- [hub.docker.com/r/trion/](https://hub.docker.com/r/trion/)
- trion/ng-cli
  - Angular-CLI, node, npm, yarn
- trion/ng-cli-karma
  - Chrome Browser, xvfb
- trion/ng-cli-e2e
  - Java, webdriver
- Doku
  - [Englisch](#), [Deutsch](#)







Testing

# Tests automatisieren



- Unit- und E2E-Tests
- Sollten automatisiert in der CI/CD Pipeline laufen
  - Inkl. Code-Coverage
  - `ng test -cc` bzw. `ng test --code-coverage`

 passed	#405 x86_64 docker	lint	🕒 01:21 📅 5 minutes ago	
 passed	#406 x86_64 docker	test:karma	🕒 01:27 📅 5 minutes ago	92.86%

# Code Qualität

- Code Coverage
- Linting (tslint/codelyzer)
  - Ebenfalls Teil der CI
  - **ng lint**

- Quellcode-Dokumentation
  - *@compodoc*
  - Doc-Coverage

@compodoc

TodoMVC Angular documentation

Type to search

- Getting started
- README
- Overview
- Additional documentation
- Modules
- Components
- Directives
- Classes
- Injectables
- Pipes
- Routes
- Miscellaneous
- Documentation coverage

Overview

Legend

- Declarations
- Module
- Bootstrap
- Providers
- Exports

Zoom in Reset Zoom out

10 modules

9 components

1 directive

2 injectables

1 pipe

2 classes

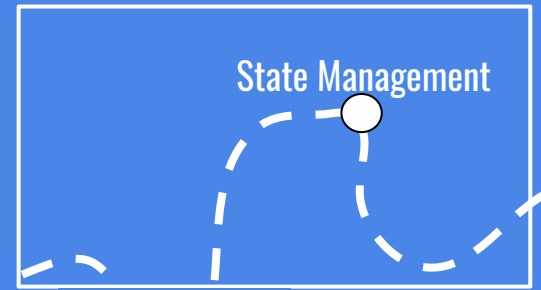
12 routes

Documentation generated using compodoc

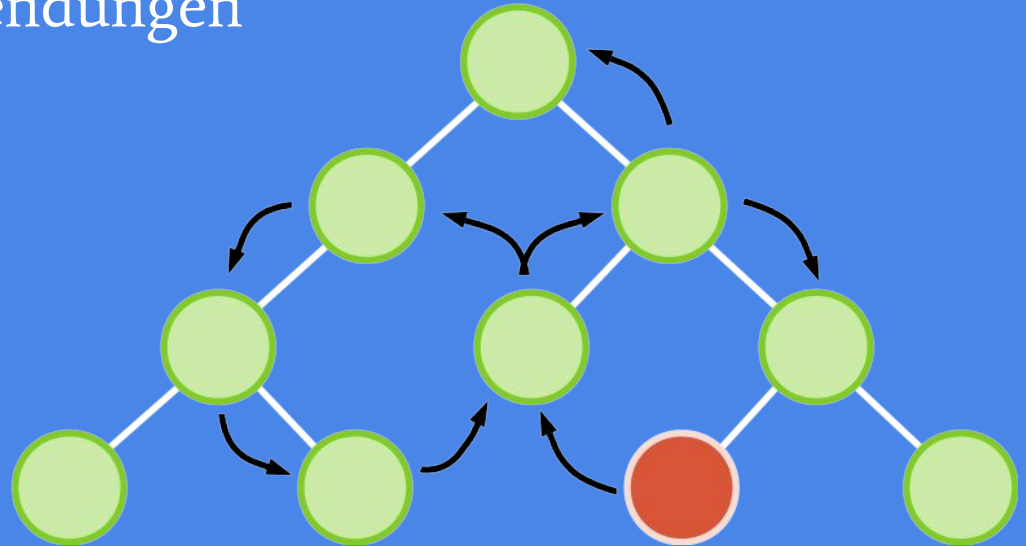
# State Management

# State Management

- Wo wird Zustand der Anwendung gehalten?
  - Komponenten, Services, Template, ...?
- Große/Komplexe Anwendungen



Komponente, von der die Änderung ausgeht

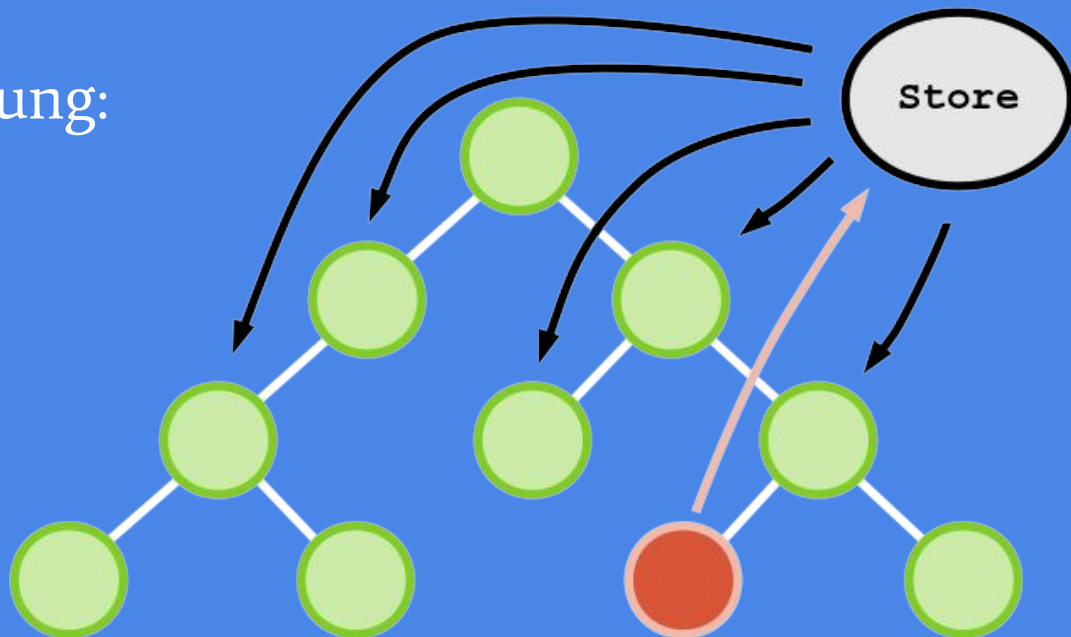


# Flux/Redux

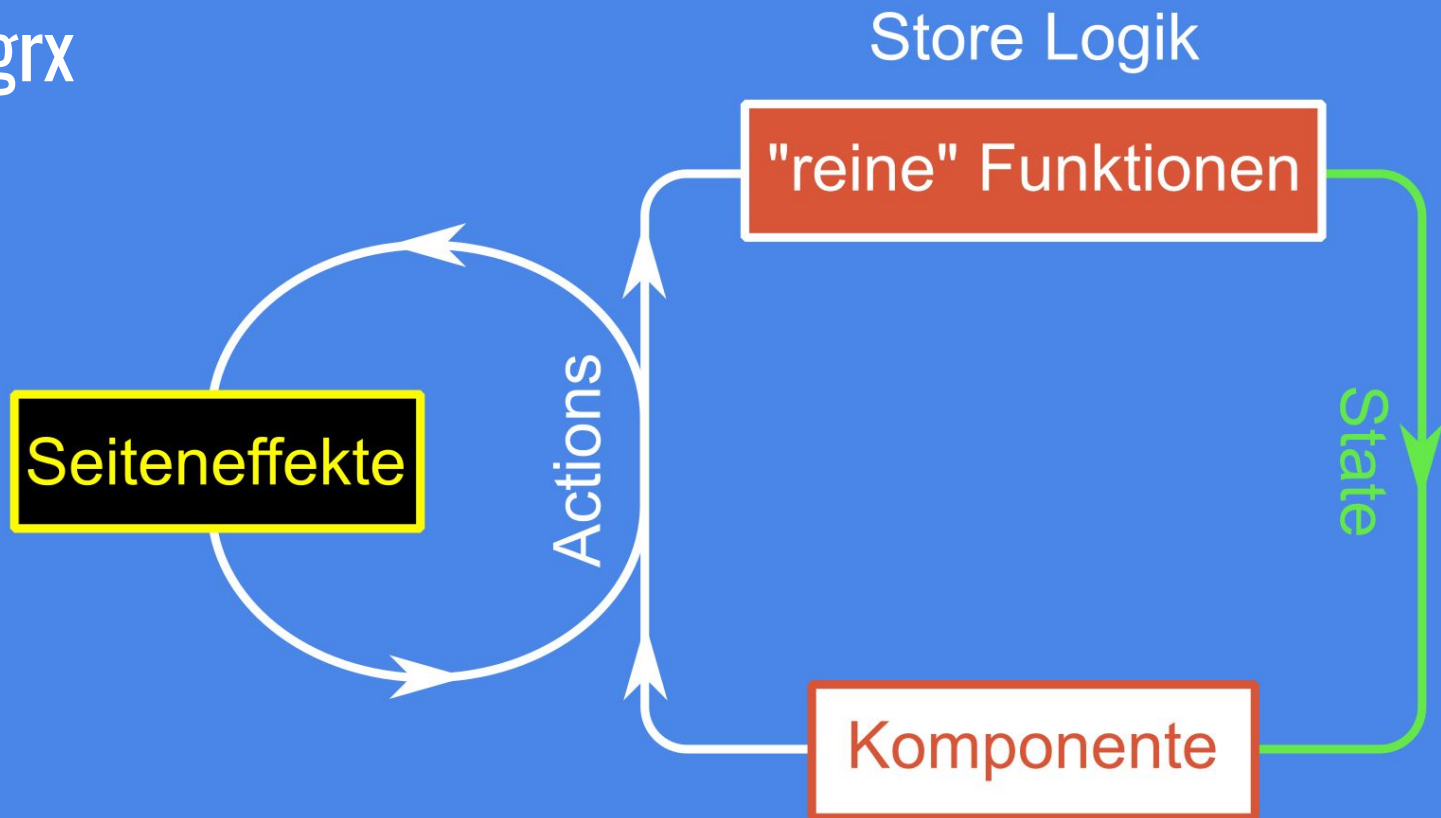
- Wo wird Zustand der Anwendung gehalten?
  - Zentraler Store
- Eine mögliche Ausprägung:  
@ngrx



Komponente, von der die Änderung ausgeht



@ngrx



# Update Policy



***“We (Google) also have one version policy, which means we are not allowed to keep existing applications on old versions of Angular”***

Ng-conf Day 1 Keynote 2018

# Fazit

- Angular stellt eine Option mit vielen Vorteilen dar
- Das Umfeld entwickelt sich kontinuierlich weiter
- Adäquate Pflege der Anwendungen und Entwickler unerlässlich
  - Kontinuierliche Fortbildung
  - Code-Review (Team-Intern/Extern)

[https://sitterberg.com/  
karsten@sitterberg.com](https://sitterberg.com/karsten@sitterberg.com)  
@kakulty