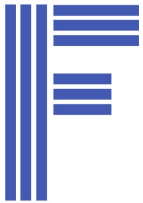
Angular Architektur

Mit Angular Elements

Thomas Kruse

- Entwickler, Trainer, Berater
  - www.trion.de
  - @everflux
- Java User Group Münster
- Frontend Freunde Münster



Karsten Sitterberg

- Entwickler, Trainer, Berater
  - sitterberg.com
  - @kakulty
- Java User Group Münster
- Frontend Freunde Münster

JAVA USER GROUP MÜNSTER

Model

- Datenzugriff / Datenänderungen (Backend)
- Validierung von Daten
- Geschäftslogik (prüfbar durch Unit Tests)

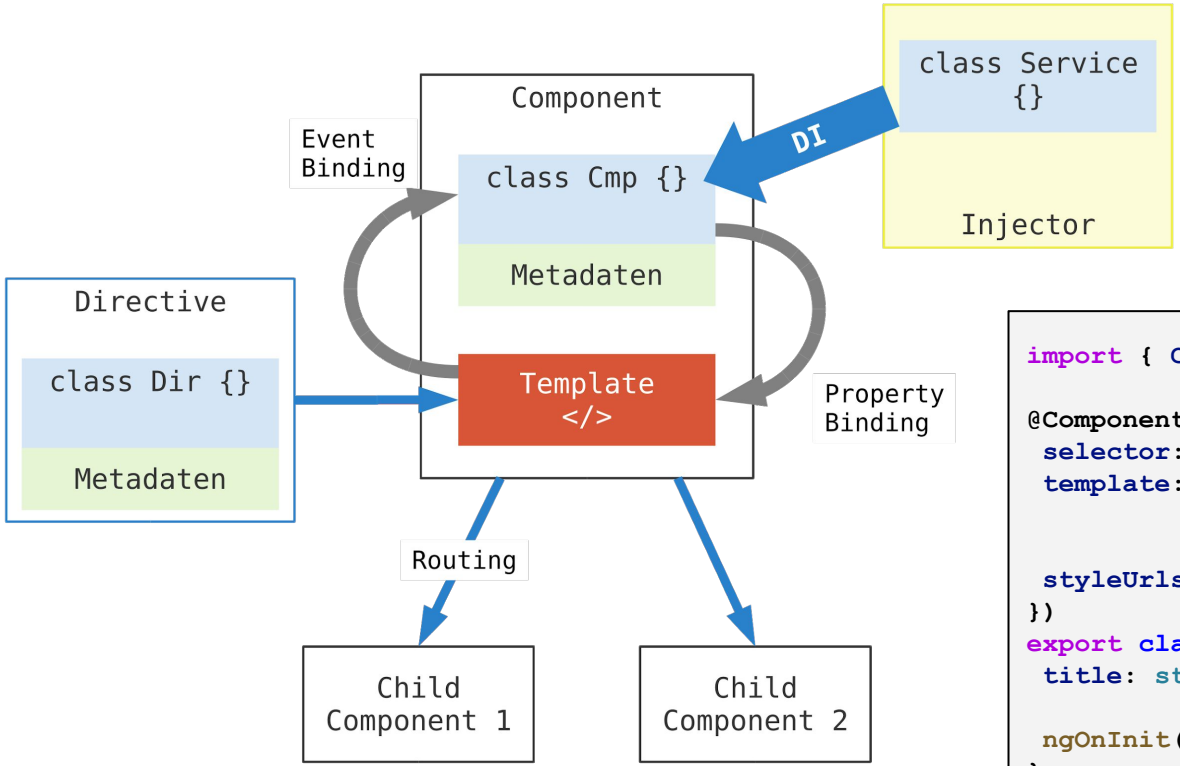
View

- Inhalte darstellen / Benutzereingaben weiterleiten
- Bindet an Eigenschaften des ViewModel

ViewModel

- UI-Logik (Model der View)
- Aufruf Methoden oder Dienste auf Model
- Bereitstellung Eigenschaften und Befehle für View
- Kein Wissen über View

Angular Komponenten




```
import { Component, OnInit } from '@angular/core';

@Component({
 selector: 'app-demo',
 template: `<h1 class="head">
              Hello {{title}}!
            </h1>`,
 styleUrls: ['./demo.component.css']
})
export class DemoComponent implements OnInit {
 title: string;

 ngOnInit() {this.title = 'World';}
}
```

CSS Isolation

- Motivation
  - Kapselung von Komponenten-Style gemeinsam mit der Komponente
  - Wiederverwendbarkeit der Komponente verbessern
  - Seiteneffekte vermeiden
- Angular
  - **Emulated** (Attribut-Selektoren)
  - **Native** (ShadowDOM)
  - **None**

Komponenten Typen

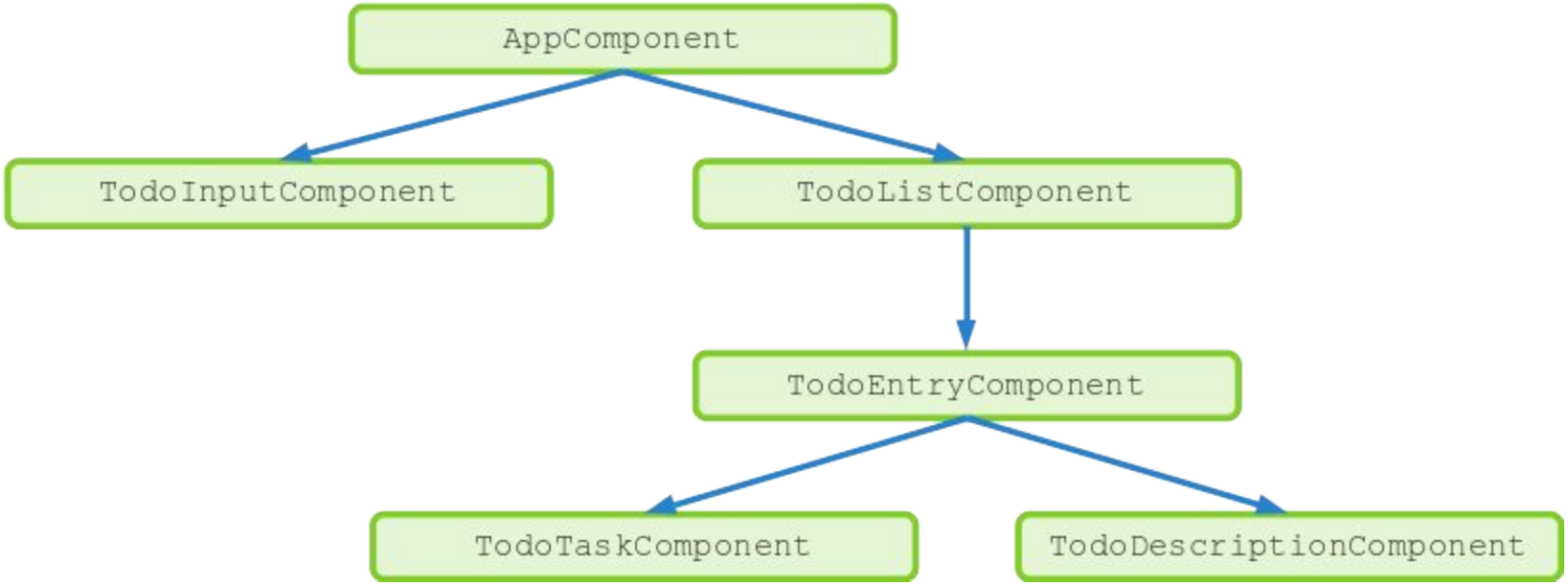
Container

- Struktureller Aufbau
- Fachlicher Kontext
- Datentransformation
- Eventhandling

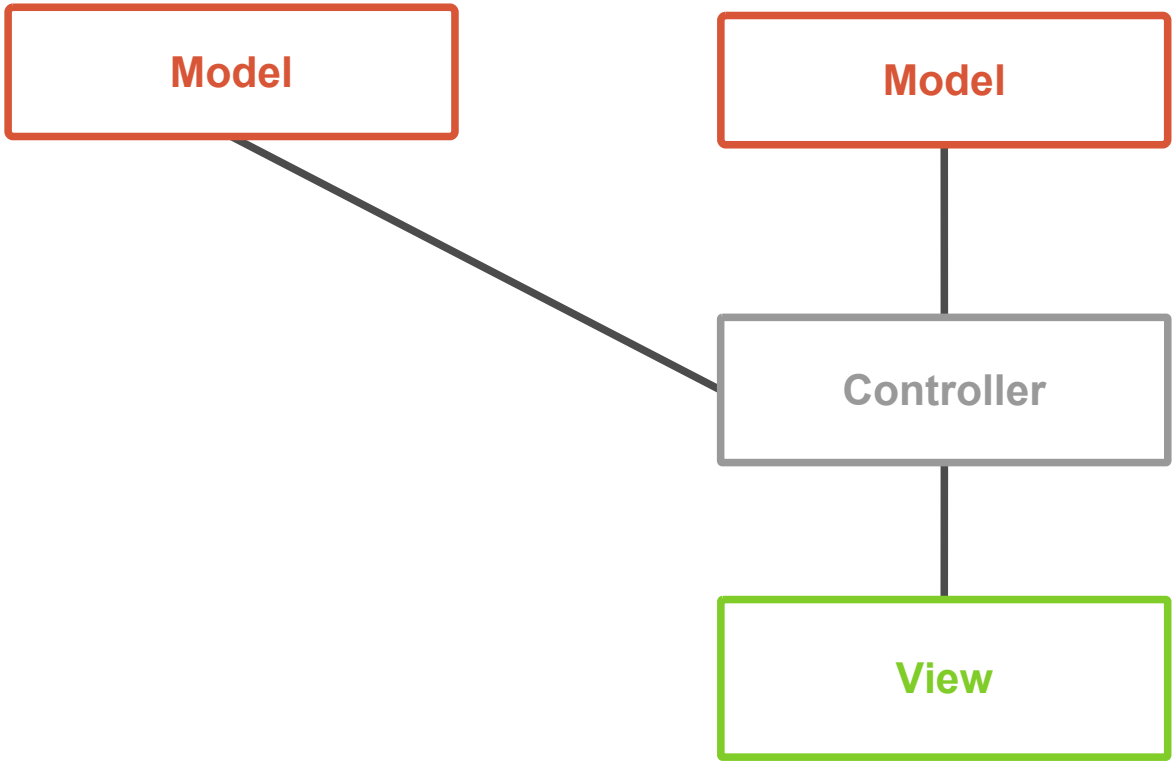
Präsentation (View)

- Rendering der View
- Behandlung Input / Nutzerinteraktion
- Primitiv / Domänenagnostisch
- Meist hohe Wiederverwendbarkeit

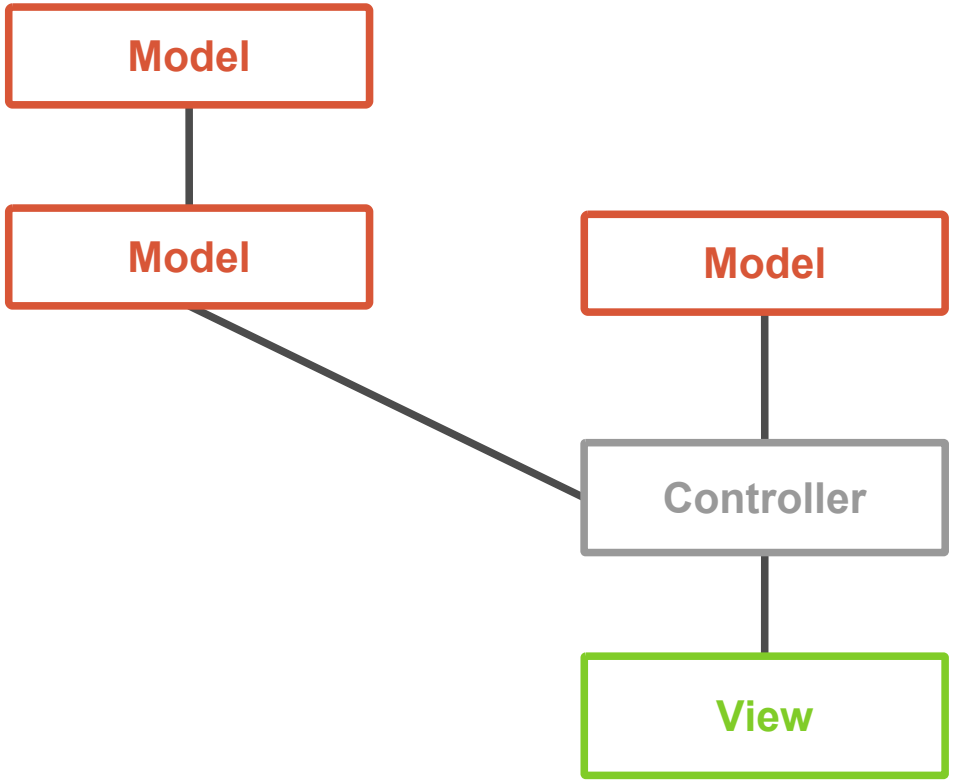
Struktureller Aufbau Angular Anwendung



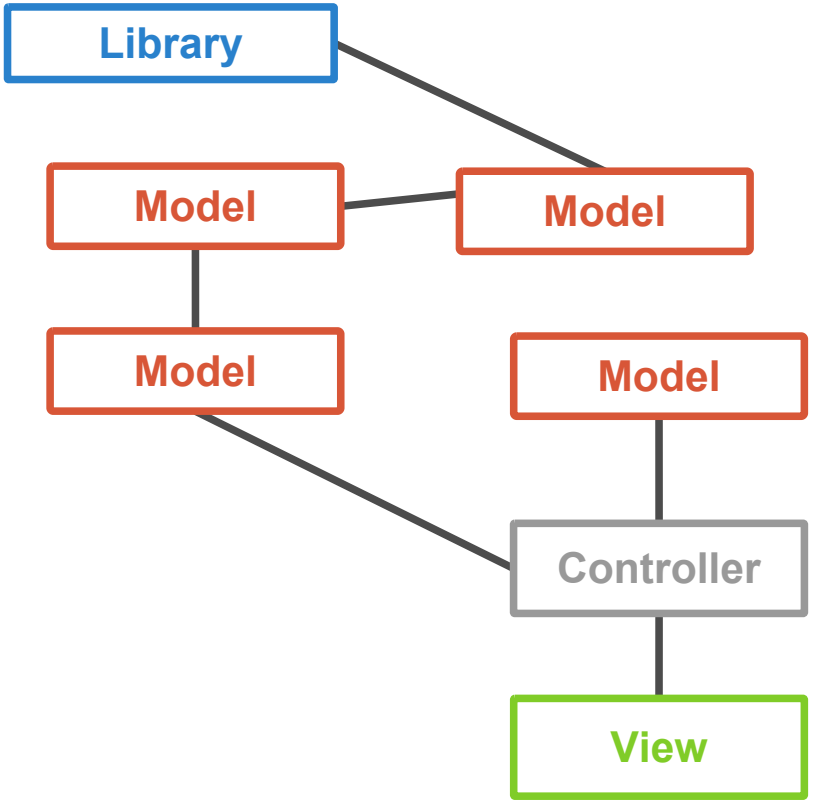
Komplexität wachsender Anwendungen



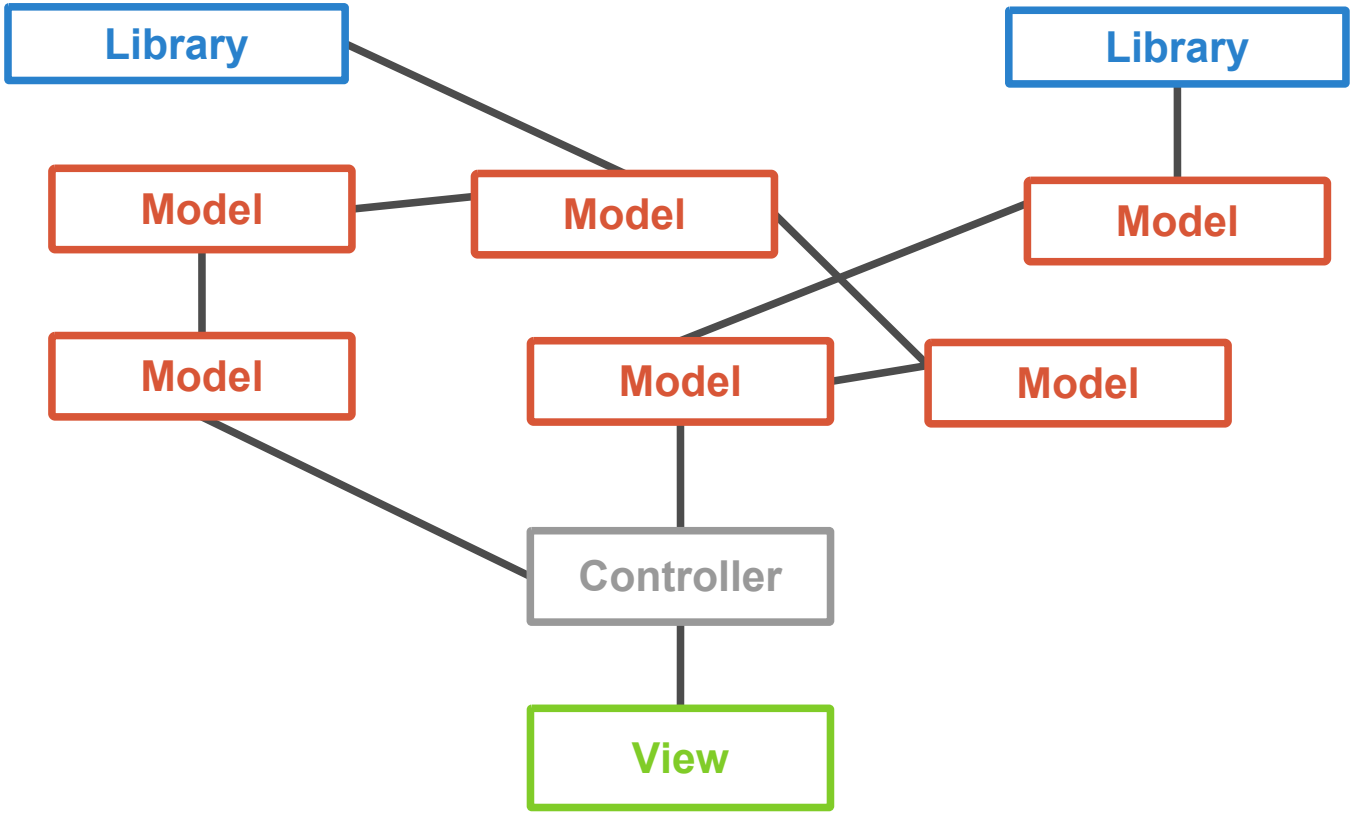
Komplexität wachsender Anwendungen



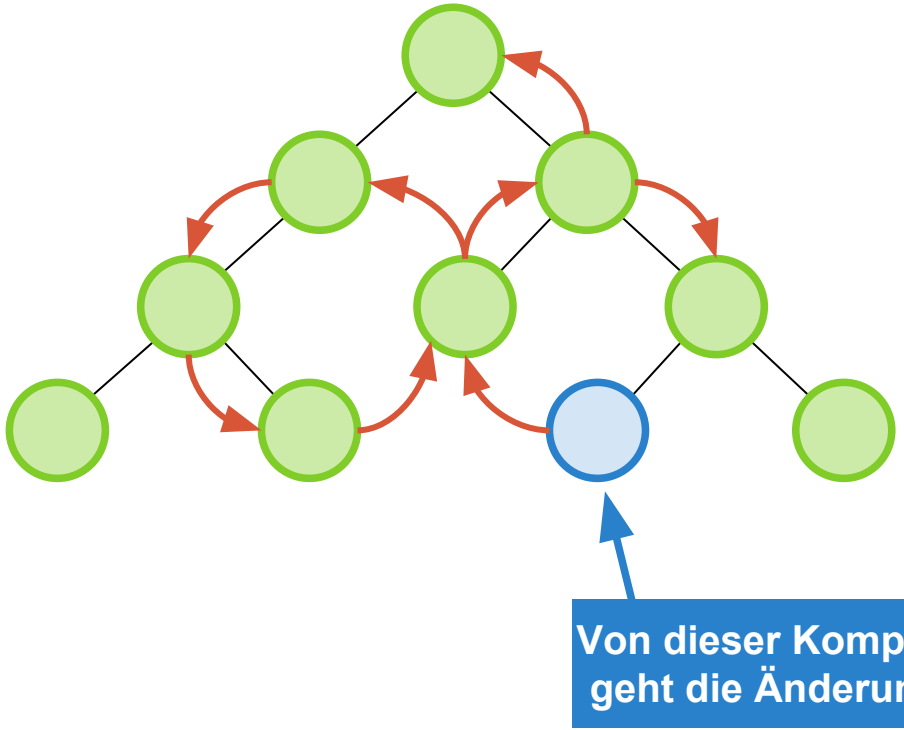
Komplexität wachsender Anwendungen



Komplexität wachsender Anwendungen



Change Propagation in Angular

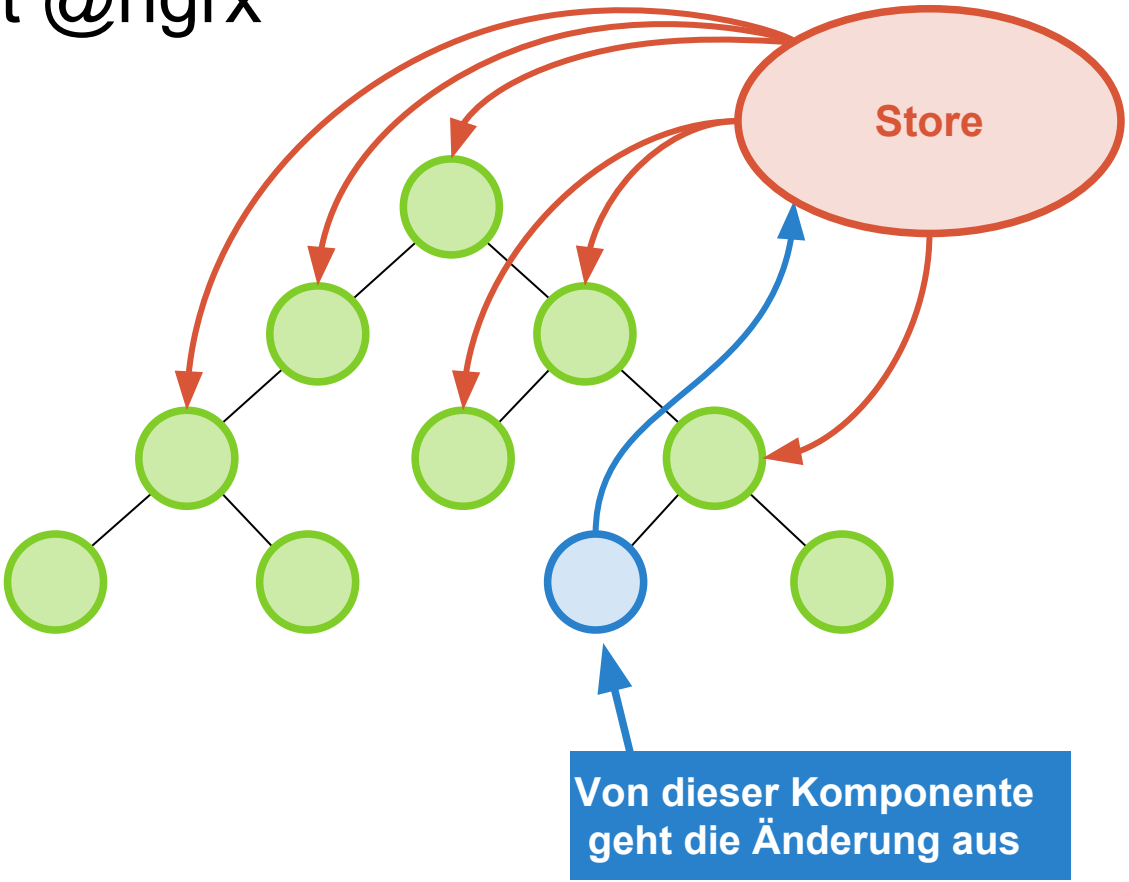


- Direkte Kommunikation:
  - Input/Property-Binding
  - Output/Event-Binding
- Indirekte Kommunikation:
  - Services
  - PubSub
  - EventBus

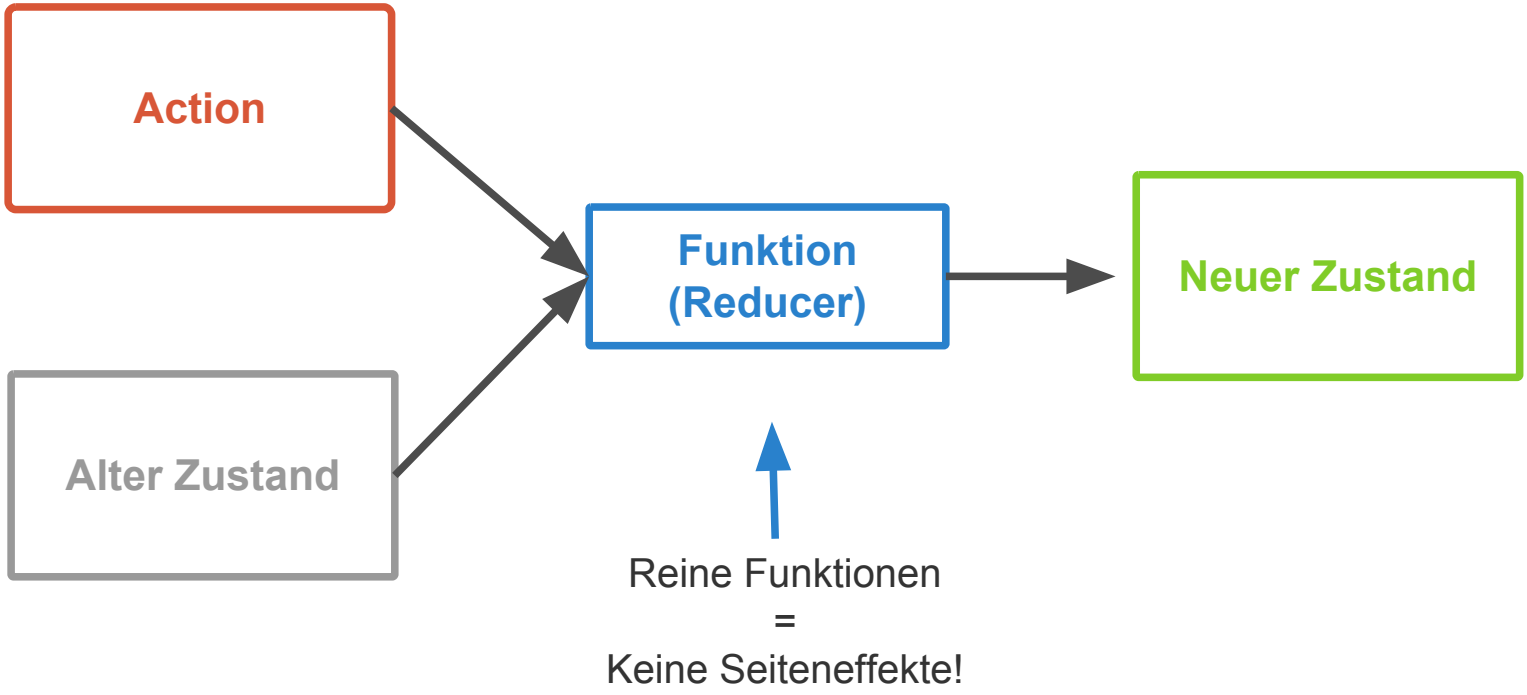
Herausforderungen bei Zustand

- Unklar, welcher Zustand wo zu halten ist
  - In Komponente
  - In Services
  - In Templates
  - Zentraler Zustandsservice
- Verteilt über Anwendung
  - Auswirkungen von Änderungen schwer einzuschätzen
  - Performance bei Change Detection
  - Schwer zu testen

Redux mit @ngrx



Aktualisierung des Zustands

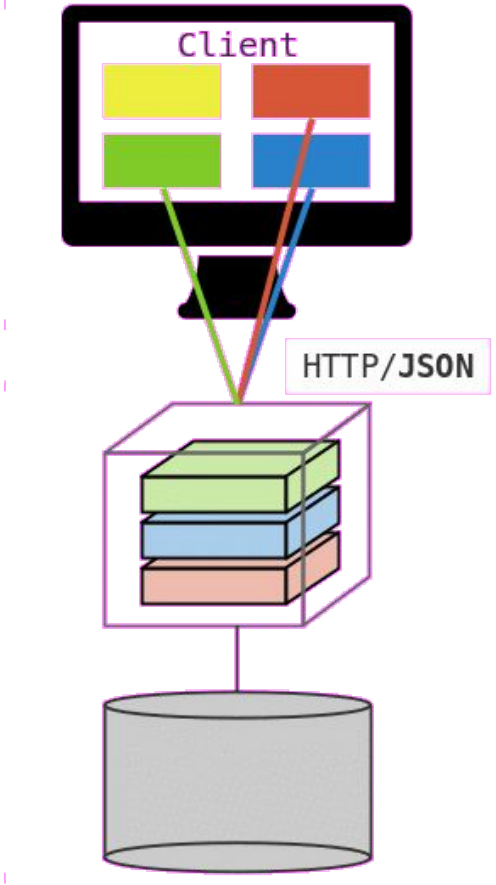


Modulsystem

- Motivation
  - Separation der Anwendung in logische Teile
  - “Features” (Separation of Concern)
  - Wiederverwendung
  - Skalierung der Entwicklung durch Teams
- Angular
  - @NgModule
  - Untergliederung der Anwendung in Features
  - Lazy Loading möglich

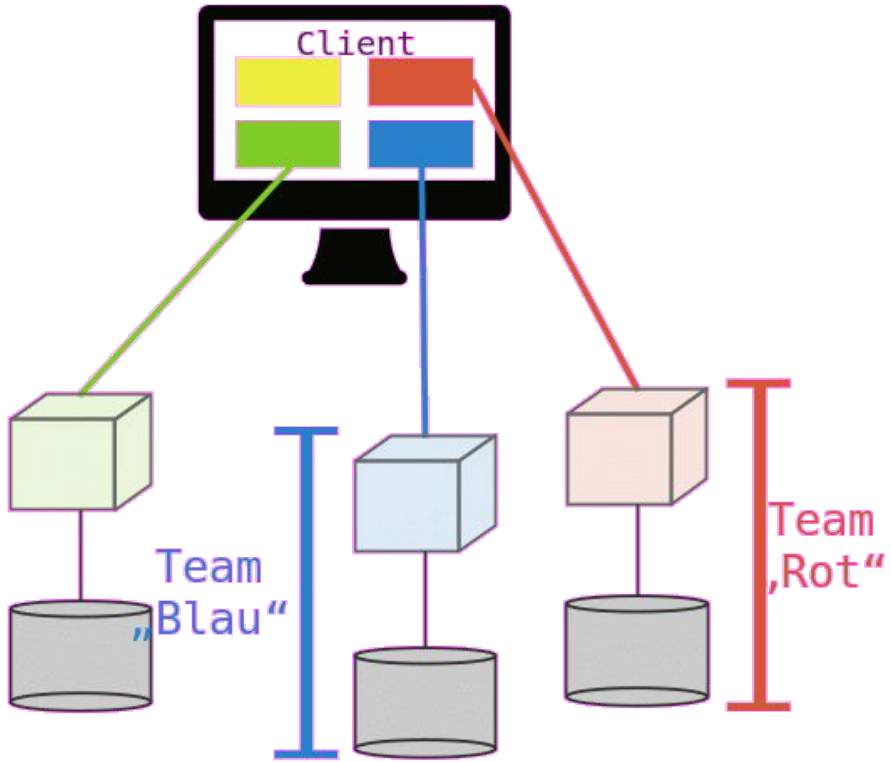
Modularer Frontendmonolith

- Mit gezeigten Mitteln lassen sich große Anwendungen entwickeln und warten
- Typischerweise wird Backend im selben Team entwickelt
- Der Trend geht zum Zweitbackend

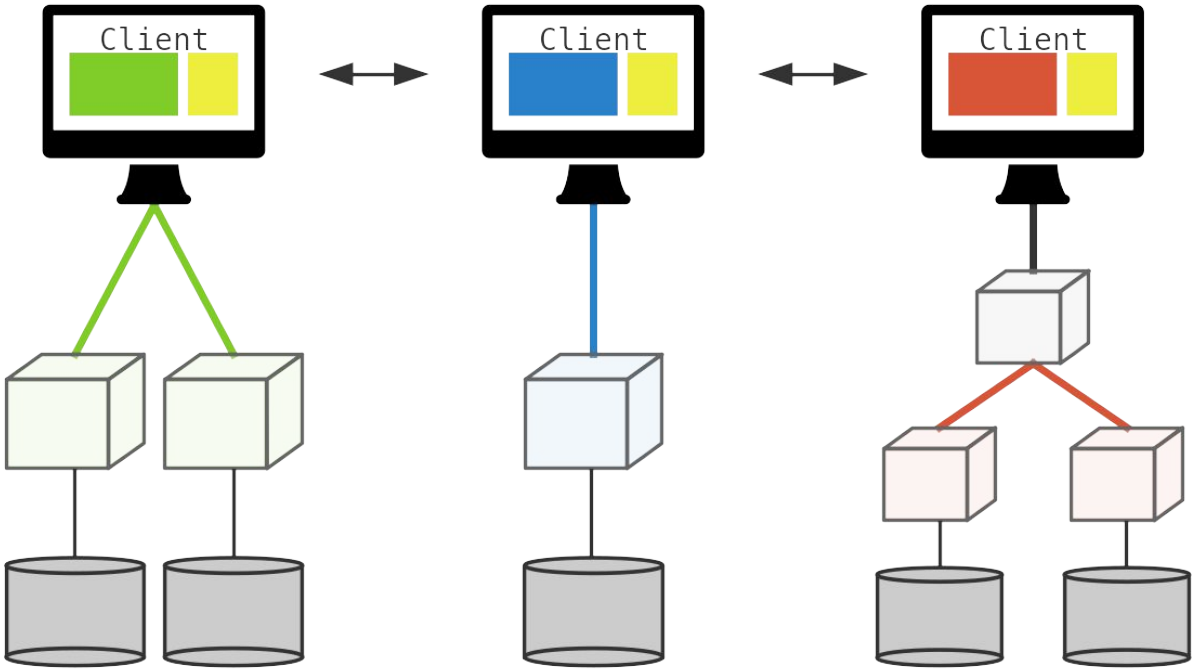


Microservices im Backend

- Besseren Skalierung der Teams
- Polyglotte Umgebung
- Was ist mit dem Frontend?

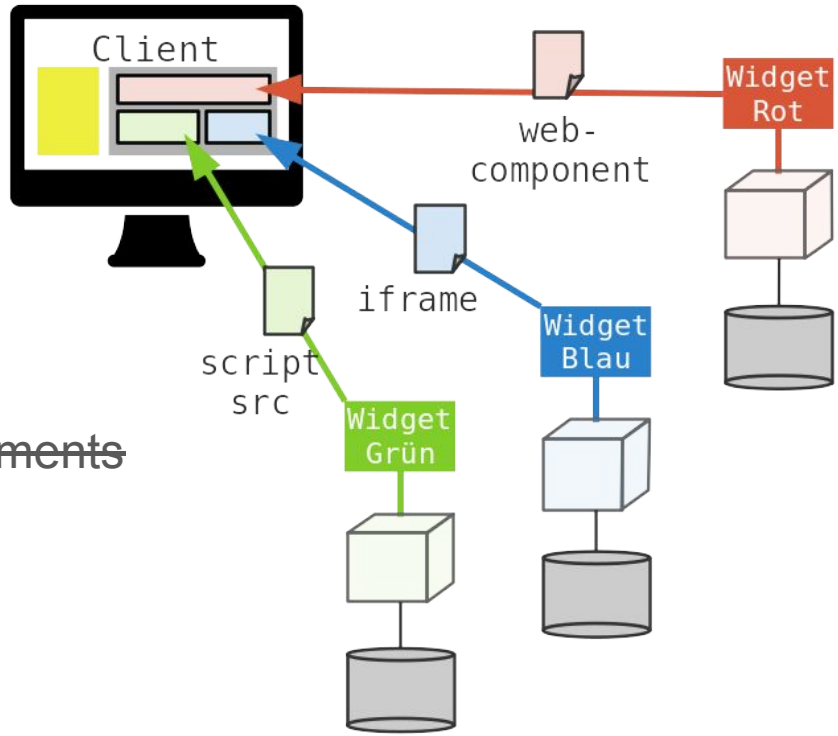


Variante: Self Contained Systems



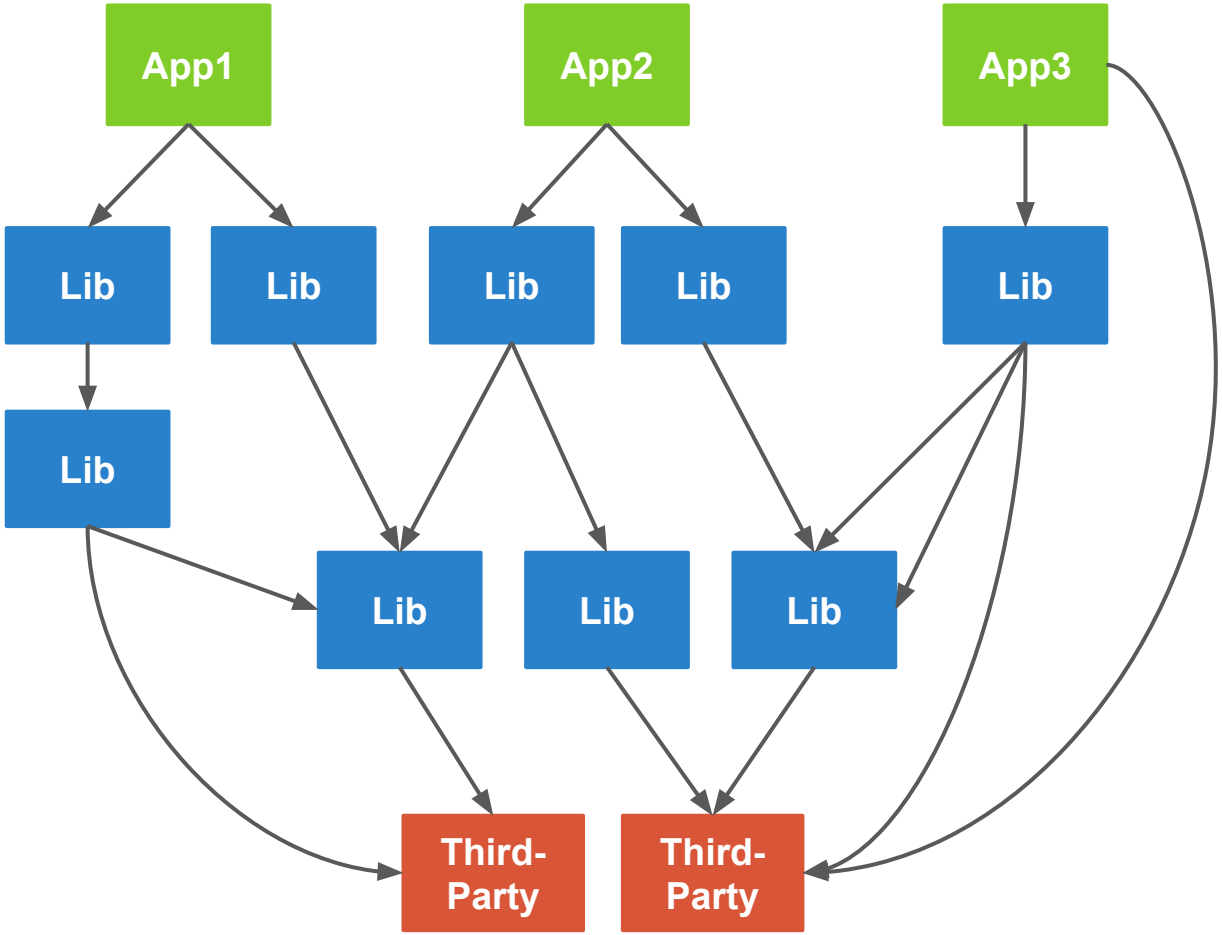
Variante: Wiederverwendung

- Libraries
- Angular Elements
- WebComponents
- ~~IFrame, ESI/SSI, Ajax Fragments~~

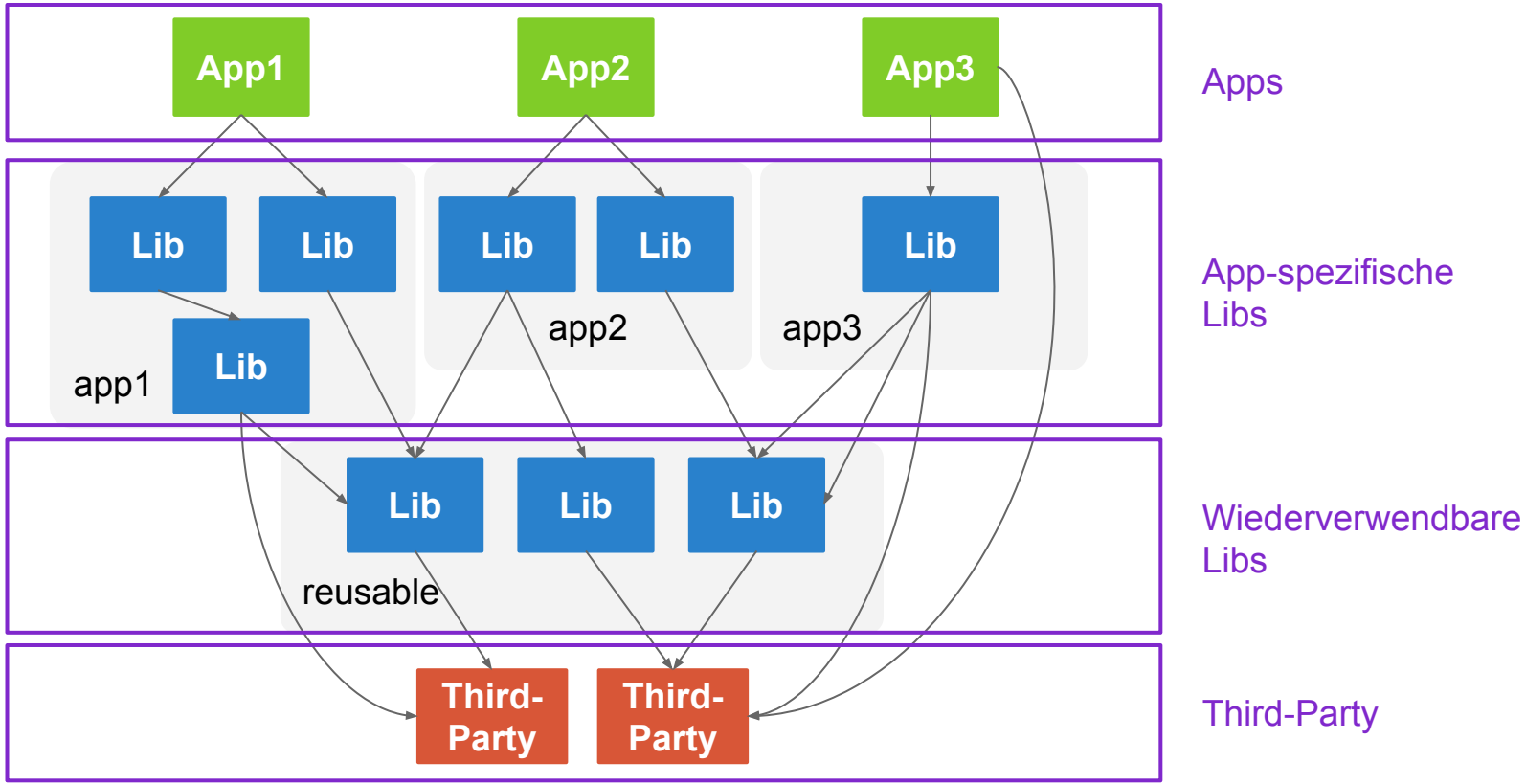


Libraries

- Umsetzung mit NPM
- Artefakt Repository
- Build-Time



Library Strukturierung



Custom Elements in Angular verwenden

```
<my-angular-app>
 <fancy-input
    [attr.foo]="someString"
    [someProp]="someProp"
    (someEvent)="doStuff()">
 </fancy-input>
</my-angular-app>
```

- Wie normale DOM Elemente im Template
- Property- und Event-Binding

Angular Elements

- Ermöglicht Entwicklung von Web Components (Custom Elements)
- Self-bootstrapping: In Seite einfügen und fertig
- Innerhalb von Custom Element ist Angular Component
- Bridge zwischen DOM APIs und Angular Components:
  - @Inputs -> Properties
  - @Outputs -> Events
  - @HostBinding/Listener -> CE Attributes / ObservedAttributes
- Zones sind optional

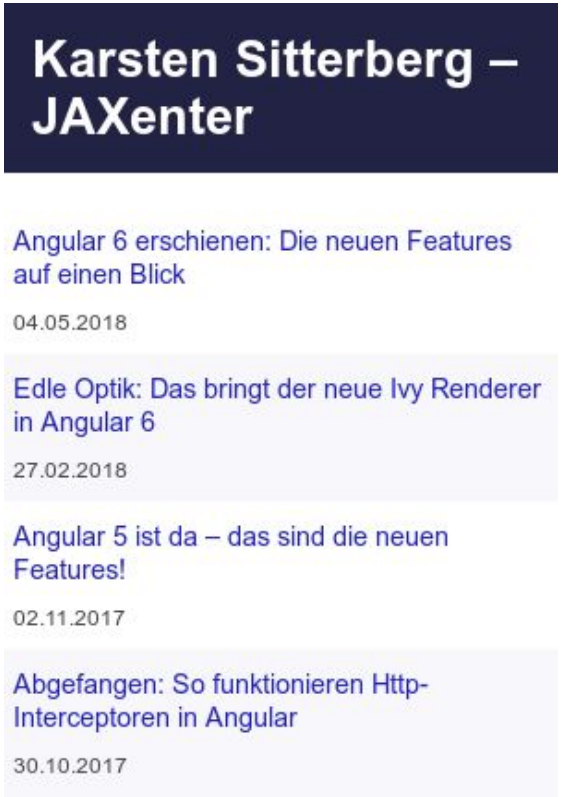
Eigenschaften Angular Elements

- Einsatz mehrerer Angular Versionen
- Können beliebig mit anderen Technologien (React, Vue, …) gemischt werden
- Angular Know-How für noch breiteres Einsatzspektrum anwendbar
- Angular Element kann simples UI Widget sein bis zu ganzer Anwendung
  - Zustand gekapselt und nur über Element API verfügbar
- Kombinierbar mit @ngrx Store

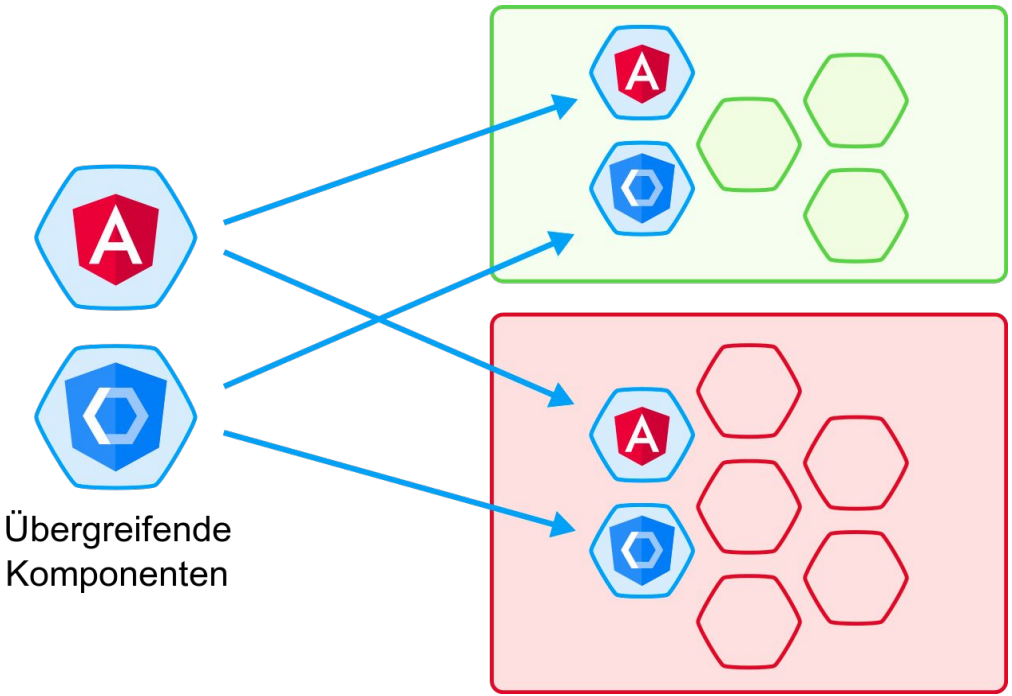
Demo

Live: https://nezrack.gitlab.io/angular-elements-demo

Code: https://github.com/codecoster/angular-elements



Einsatz von Angular und Web Components



Fazit

- Bereits heute für Architektur berücksichtigen
- Mit Angular 7 / Ivy Renderer optimal nutzbar
- Keine One-Size-Fits-All Lösung - Varianten betrachten

Danke.

Fragen?

@kakulty sitterberg.com

@everflux trion.de