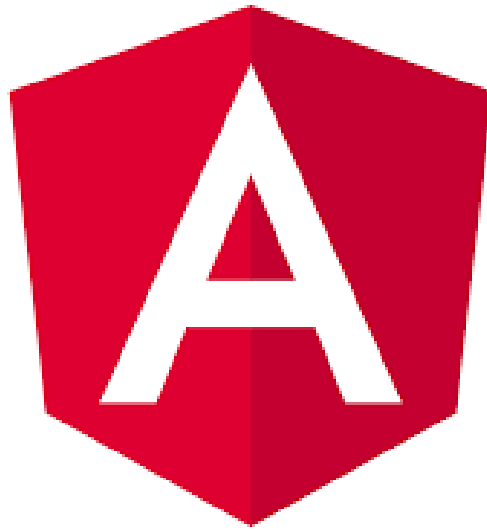


Angular Grundlagen



Was ist Angular?

- Ein JavaScript-Client-Framework für dynamische Webapplikationen
- Speziell für die Entwicklung von Single-Page-Apps gedacht
- Entwickelt von Google
- Ein Open-Source-Projekt
- Eine **MVC/MVVM-Framework**, das bidirektionales Databinding unterstützt
- Auf gute Testbarkeit ausgelegt

Warum Client-Frameworks

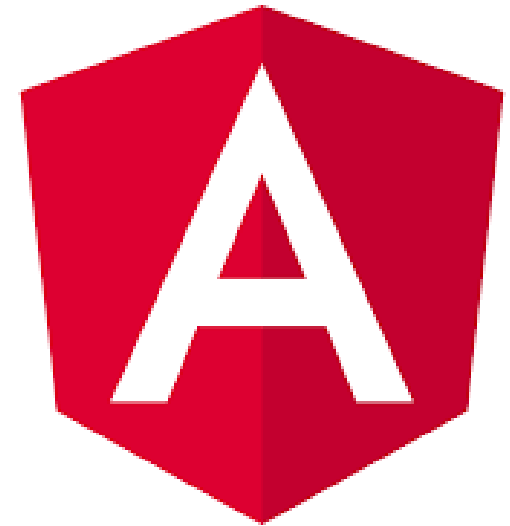
- Am Client steht viel Leistung zur Verfügung
- Der Server liefert nur mehr die Grundstruktur der Webseite an die Clients aus
- Client bezieht über REST-Services die benötigten Daten.
- Am Client steht Software zur Verfügung, die aufgrund der Templates den DOM aufbaut und mit den Daten aus dem REST-Service füllt.
- Code Reduktion (durch Automatisierung von Standardaufgaben und Data-Binding)

Warum Angular?

- Hinter Angular steht Google → Etablierung am Markt
- Angular ist äußerst leistungsfähig und stellt eine Struktur für die Organisation eines Projektes zur Verfügung.
- Trennung zwischen Design und Logik
- Die Serverseite ist vom Design nicht betroffen, dort liegen alle Files, Webservices, sowie Business-Logik. Auch der Datenbank-Zugriff wird am Server ausgeführt.
- Die Architektur entspricht der bisherigen Philosophie, aber ohne HTML-Generierung und Routing.
- JavaScript-Frameworks sind per Definition Open-Source, daher hat man einen genauen Einblick sowie eigene Weiterentwicklungsmöglichkeit.

BuildingBlocks

- Component
- Module
- Directive
- Service



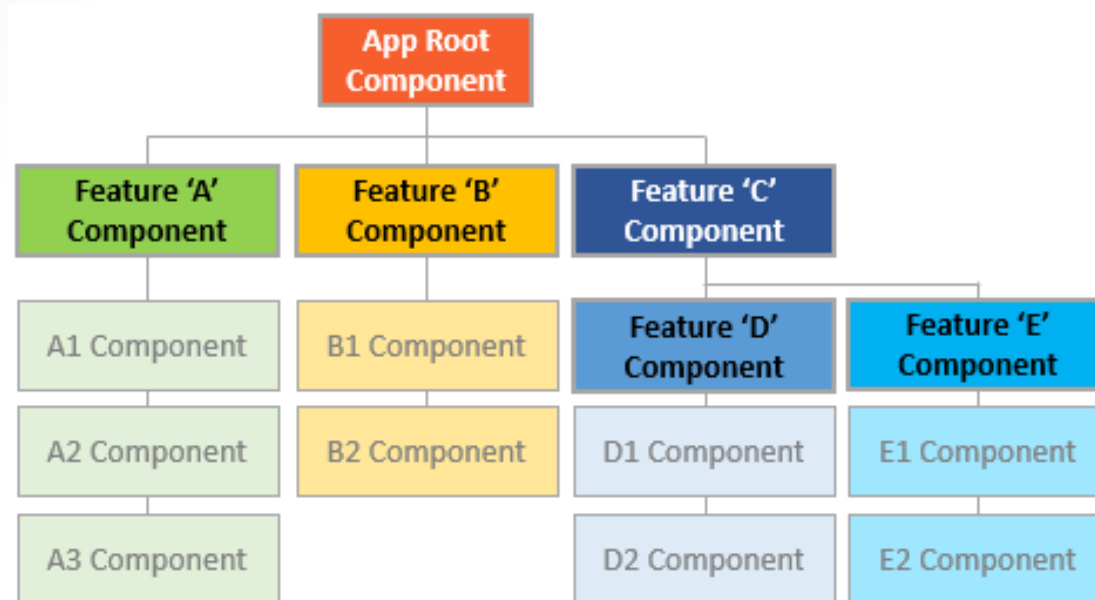
Component

- Verwalten die verschiedenen Bereiche der WebSite
 - Navbar, SideBar, Content
 - Sind baumartig strukturiert
- Speziell „dekorierte“ Klasse
- MVVM-Implementierung



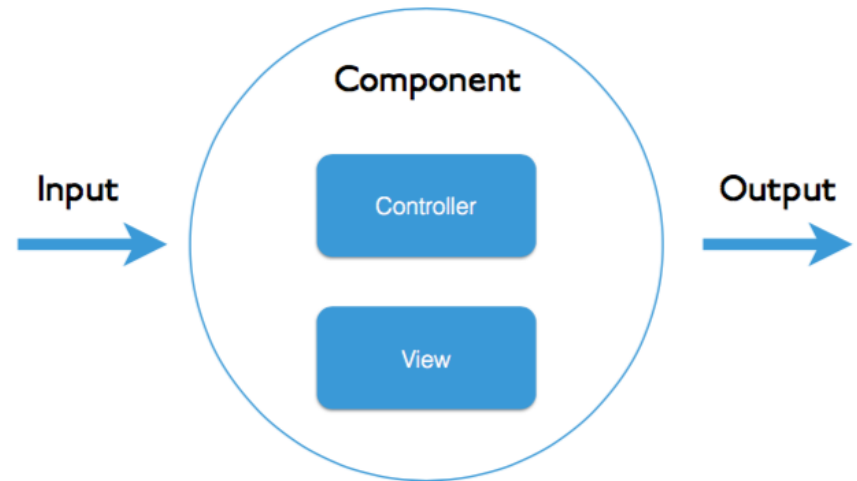
Angular Module

- Strukturierung der App in mehrere Areas
- Auch wieder baumartig aufgebaut
- Mindestens ein Module app-module



Komponente erzeugen

- Codierung
- Registrierung
- Einbettung in HTML

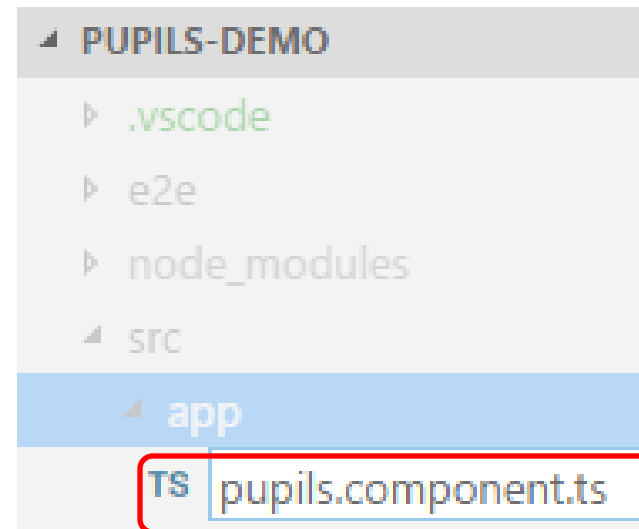


Komponente von Hand erzeugen

- Angular-Projekt anlegen „ng new pupils-demo“
- Projektverzeichnis in VsCode öffnen
- Datei pupils.component.ts erzeugen
- Klasse heißt PupilsComponent
 - Codierrichtlinien beachten!

TS pupils.component.ts ●

```
1
2 export class PupilsComponent {
3
4 }
```



Als Component dekorieren

- Annotation mit Decoratorfunction
 - Konfigurationsobjekt als Parameter

TS pupils.component.ts ✕

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-pupils',
5    template: '<h2>Pupils</h2>'
6  })
7  export class PupilsComponent {
8  }
```

In app.module.ts registrieren

- import wird automatisch hinzugefügt
 - VsCode-Extension AngularEssentials

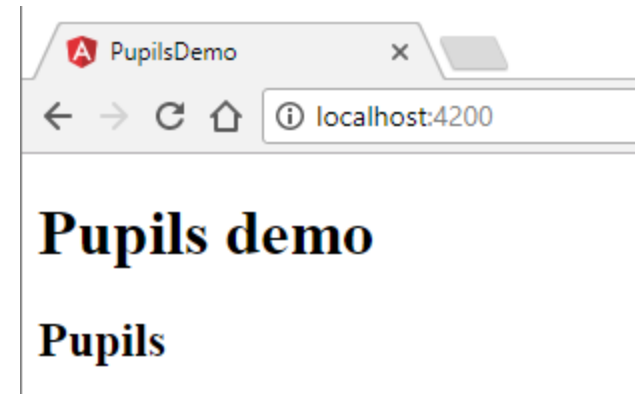
```
import { AppComponent } from './app.component';  
import { PupilComponent } from './pupil/pupil.component';  
import { PupilsComponent } from './pupils.component';
```

```
@NgModule({  
  declarations: [  
    AppComponent,  
    PupilComponent,  
    PupilsComponent  
  ],  
})
```

Neues Element in HTML eintragen

- ng serve läuft in CommandLine

```
<> app.component.html x
1 | <h1>Pupils demo</h1>
2 |
3 | <app-pupils></app-pupils>
4 |
```



Component über CLI erzeugen lassen

- In VsCode Terminal öffnen (Strg ö) `ng g c pupil`
- Erzeugt Dateien entsprechend geltender Codierrichtlinie

```
create src/app/pupil/pupil.component.html (24 bytes)
create src/app/pupil/pupil.component.spec.ts (621 bytes)
create src/app/pupil/pupil.component.ts (265 bytes)
create src/app/pupil/pupil.component.css (0 bytes)
update src/app/app.module.ts (394 bytes)
```

- HTML-Template als eigene Datei
- Vorlage für Unittests (spec)

Je Component eigener Ordner

```
└─ src
   └─ app
      └─ pupil
         # pupil.component.css
         <> pupil.component.html
         TS pupil.component.spec.ts
         TS pupil.component.ts
```

Html-Template ist überschaubar

<> pupil.component.html ✕

```
1  <p>  
2    pupil works!  
3  </p>
```

Angelegter Rahmen für Klasse

TS *pupil.component.ts* ✕

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-pupil',
5    templateUrl: './pupil.component.html',
6    styleUrls: ['./pupil.component.css']
7  })
8  export class PupilComponent implements OnInit {
9
10     constructor() { }
11
12     ngOnInit() {
13     }
14
15 }
```


Template bearbeiten - BackTick

```
@Component({  
  selector: 'app-pupils',  
  template: `  
    <h2>Pupils</h2>  
  `;  
})
```

- Erlaubt mehrzeilige Inlinetemplates

Erste Verbindung View ⇔ Class

TS pupils.component.ts x

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-pupils',
5    template: `
6      | | | | | <h2>{{title}}</h2>
7      |
8    })
9  export class PupilsComponent {
10    private _title = 'PupilsTitle';
11
12    get title(){
13      | return this._title;
14    }
15
16  }
```

Interpolation

- Bequeme Methode, um in der View Platzhalter durch Texte zu ersetzen
- Erlaubt auch Expressions oder Methodenaufrufe

```
selector: 'app-pupils',  
template: `  
  <h2>{{17*3+2}}</h2>  
`  
})  
export class PupilsComponent {
```

Pupils demo

53

- Manipulieren den DOM
- Strukturelle Änderungen → Präfix *
 - *ngFor, *ngIf

Directive *ngFor

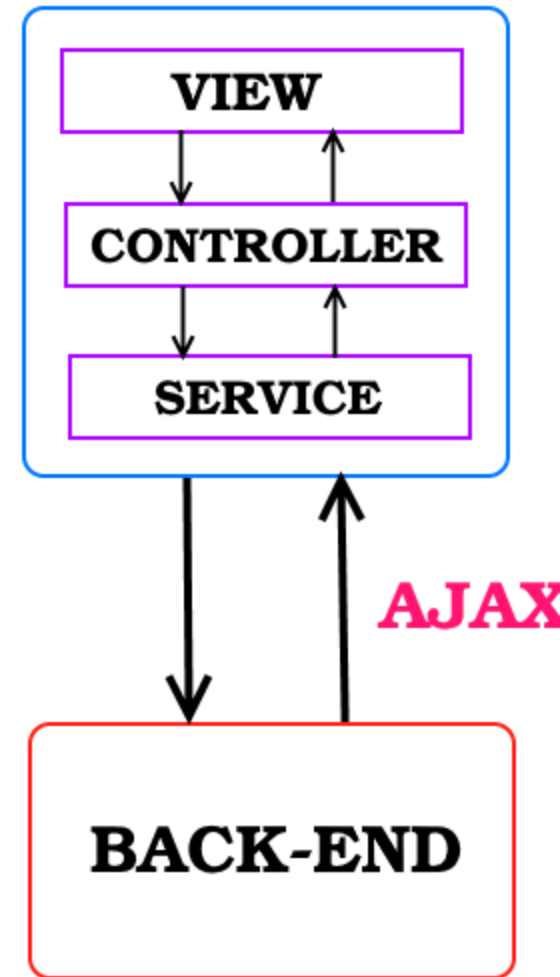
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-pupils',
  template: `
    <h2>{{title}}</h2>
    <ul>
      <li *ngFor="let pupilName of pupilNames">
        {{pupilName}}
      </li>
    </ul>
  `
})
export class PupilsComponent {
  private _title = 'Pupils from ZABTE/ZAKTE';
  pupilNames = ['Eder Michael', 'Bachinger Daniel', 'Krump Manuel', 'Dujmovic Richard' ];

  get title(){
    return this._title;
  }
}
```

Services

- Klassen, die keine eigene Oberfläche besitzen
- Für bestimmte Aufgaben verantwortlich sind
 - Datenbeschaffung über REST
 - Notifications
 - Berechnungen
 -
- Erhöhen Testbarkeit
 - Test des Service unabhängig von der UI
 - Test der UI unabhängig vom Service (FakeService)



- Angular enthält DI-Framework
 - Constructor-Injection in Components
- Normale Klasse
 - Injectable nur, falls Service anderen Service injiziert bekommt
 - Component ist von Haus aus Injectable

```
import { Injectable } from '@angular/core';

@Injectable()
export class PupilsService {

  constructor() { }

}
```

In app.module.ts registrieren

```
TS app.module.ts ●
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4
5  import { AppComponent } from './app.component';
6  import { PupilComponent } from './pupil/pupil.component';
7  import { PupilsComponent } from './pupils.component';
8  import { PupilsService } from './pupils.service';
9
10
11  @NgModule({
12    declarations: [
13      AppComponent,
14      PupilComponent,
15      PupilsComponent
16    ],
17    imports: [
18      BrowserModule
19    ],
20    providers: [
21      PupilsService
22    ],
23    bootstrap: [AppComponent]
24  })
25  export class AppModule { }
26
```


Service implementieren

- „Fake“-Service
 - Sonst Zugriff über REST-Service

TS pupils.service.ts x

```
1  import { Injectable } from '@angular/core';
2
3  @Injectable()
4  export class PupilsService {
5      _pupilNames = ['Eder Michael', 'Bachinger Daniel', 'Krump Manuel', 'Dujmovic Richard' ];
6
7      constructor() { }
8
9      getPupilNames() {
10         return this._pupilNames;
11     }
12
13 }
```

Verwendung in Component


TS pupils.component.ts x

```
1  import { Component } from '@angular/core';
2  import { PupilsService } from './pupils.service';
3
4  @Component({
5    selector: 'app-pupils',
6    template: `
7      <h2>{{title}}</h2>
8      <ul>
9        <li *ngFor='let pupilName of pupilNames'>
10          {{pupilName}}
11        </li>
12      </ul>
13    `
14  })
15  export class PupilsComponent {
16    private _title = 'Pupils from 7ABIF/7AKIF';
17    _pupilNames: string[];
18
19    constructor(pupilsService: PupilsService) {
20      this._pupilNames = pupilsService.getPupilNames();
21    }
22
23    get title() {
24      return this._title;
25    }
26
27    get pupilNames() {
28      return this._pupilNames;
29    }
30  }
```

CSS2 Selectors

*	All elements
div	<div>
div *	All elements within <div>
div span	 within <div>
div, span	<div> and
div > span	 with parent <div>
div + span	 preceded by <div>
.class	Elements of class "class"
div.class	<div> of class "class"
#itemid	Element with id "itemid"
div#itemid	<div> with id "itemid"
a[attr]	<a> with attribute "attr"
a[attr='x']	<a> when "attr" is "x"
a[class~='x']	<a> when class is a list containing 'x'
a[lang='en']	<a> when lang begins "en"

- Mittels css-File (cascading style sheets) je Component können die Styles gesetzt werden

```
h1 {  
  color:  #369;  
  font-family: Arial, Helvetica, sans-serif;  
  font-size: 250%;  
}
```

Model-Interface

```
ng g interface models/saveVehicle --type=model
```

```
export interface SaveVehicle {  
  id: number;  
  modelId: number;  
  makeId: number;  
  isRegistered: boolean;  
  features: number[];  
  contact: Contact;  
}
```

Instanz erstellen

```
vehicle: SaveVehicle = {  
  id: 0,  
  makeId: 0,  
  modelId: 0,  
  isRegistered: false,  
  features: [],  
  contact: {  
    name: '',  
    email: '',  
    phone: '',  
  },  
};
```