

Angular



Wozu Angular?

- Single-Page-Applications (SPA)
- Trennung Design und Logik
 - kein Zugriff auf DOM aus JS-Code heraus
 - keine Abhängigkeiten des JS-Codes vom DOM
- Baukastensystem mit Komponenten
- Wiederverwendbarkeit der Komponenten
- Seitenumschaltung mit URL-Navigation



Techniken

- Datenbindungen (Einweg, Zweiweg, String-Interpolation)
- ES2015-Modulsystem
- ES2015-Syntax (oder TypeScript)
- ES2016-Decorator (auch Annotations oder Attributes genannt)
- kontinuierlich Aktualisierungen mit neuen Javascript-/Typesrcipt-Versionen
- Dependency Injection



Bausteine

- Components
- Modules
- Services
- Pipes
- Routing
- u. a.



Dokumentation

https://angular.io/

neu: https://angular.dev/

Cheat-Sheet:

https://angular.io/docs/ts/latest/guide/cheatsheet.html

CLI-Commands:

https://angular.io/cli



Entwicklungswerkzeuge

- Beliebiger Editor
 - z. B. Visual Studio Code, WebStorm etc.
- Package-/Runtime-Management
 - npm, webpack usw.
- Compiler / Transpiler
 - TypeScript, Babel etc.
- Debugging
 - Browser-F12-Tools



Einrichtung



Vorbereitung

- Node.js installieren https://nodejs.org/en/
- Angular CLI installieren https://github.com/angular/angular-cli
- Verzeichnis anlegen
- Projekt anlegen mit ng new projektname
- Wechsel in Unterverzeichnis und Starten mit ng serve
- Automatische Prüfung auf Änderungen
 - Rebuild
 - Browser-Refresh



Editor, Angular CLI

- Visual Studio Code
 - Verzeichnis öffnen
- Dokumentation Angular CLI <u>https://github.com/angular/angular-cli/wiki</u>



Konzepte



Modules

- Angular benutzt ES 2015 Modules
- Vorabnutzung mit Typescript
 - oder Transpiler

Exports

continentList.component.ts

export class ContinentListComponent {...}

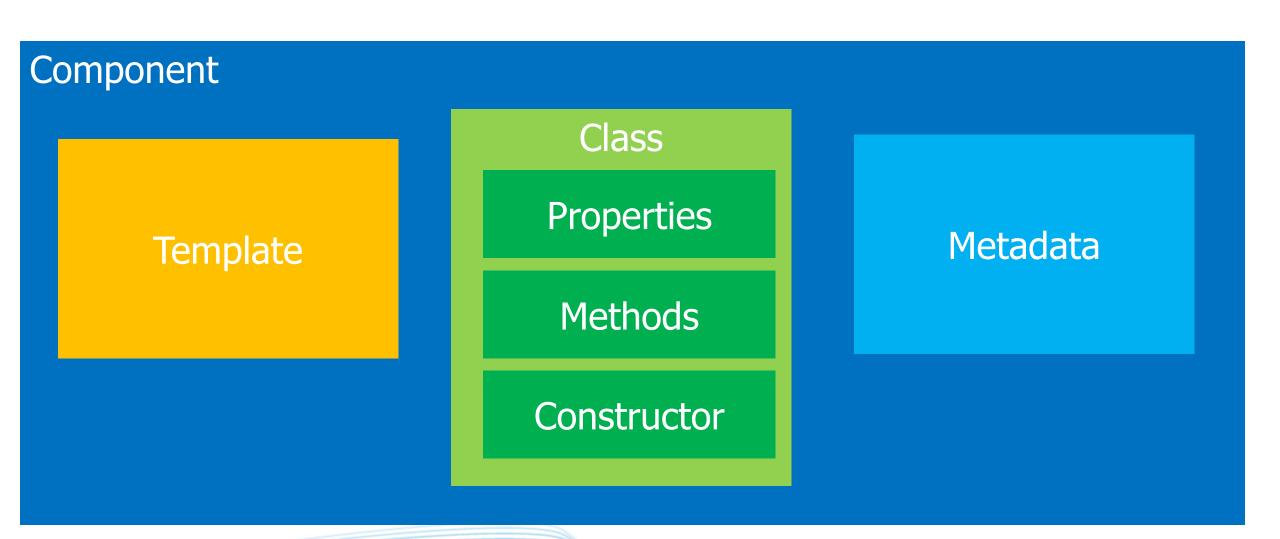
Imports

mondial.module.ts

import {ContinentListComponent} from
"./continentList.component";



Components





Component

continentList.component.ts

```
import { Component, OnChanges } from '@angular/core';
                                                                              Import
@Component({
 selector: 'mondial-continents',
 templateUrl: 'continentList.component.html',
 styleUrls: ['../app.component.css']
                                                                      Metadata,
                                                                       Template
export class ContinentListComponent {
 public selectedContinent: IContinent;
 . . .
                                                                            Class
```

Dr. Joachim Fuchs



Konventionen

- Component-Class:
 - Feature-Name + "Component"
 - Dateiname: Feature-Name.component.ts
- @Component()-Decorator
- Startup:
 - class AppComponent in app.component.ts



Templates

Inline Template

template:

"<h1>{{info}}</h1>"

Inline Template

template: `

<div>

<h1>{{info}}</h1>

• •

</div>

Linked Template

templateUrl:

"dingsbums.component.html"

ES2015 Backticks

Dr. Joachim Fuchs

*Directives

- *ngIf
- *ngFor

Bindings

- Interpolation
 - **-** {{...}}
- One-Way
 - [prop]="..."
 - [style.width.px]="..."
- Two-Way
 - [(ngModel)]="..."

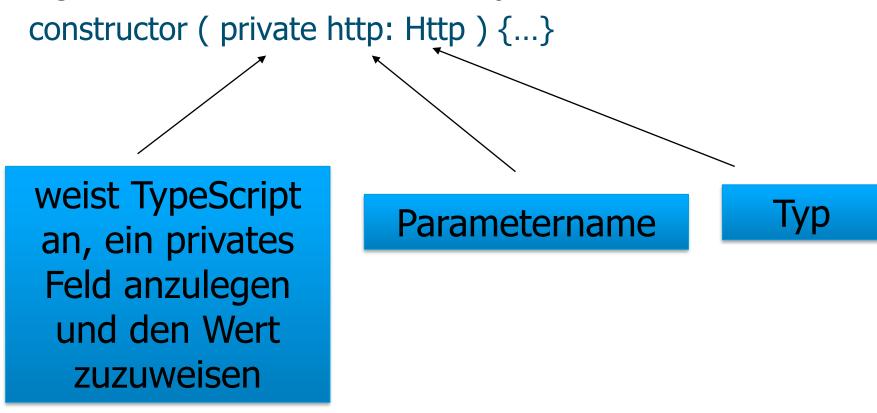
Employer: {{employer?.companyName}}

The safe navigation operator (?) means that the employer field is optional and if undefined, the rest of the expression should be ignored.



Dependency Injection

Angular verwendet Constructor-Injection:





Services



Service

- Ist eine Klasse annotiert mit @Injectable
- Singleton, wird über Dependency Injection bereit gestellt
- Muss in einem Module unter providers aufgelistet werden:

```
@NgModule({
    imports: [ ...],
    declarations: [...],
    providers:[MondialService]})
    export class MondialModule { }

oder über    @Injectable({
        providedIn: 'root',
     })
```



Aufbau von Anwendungen



Aufteilung in Module

- App-Module
- Shell- / Framework-Module
- Feature-Module 1
- Feature-Module 2
- ...
- Shared Module
- oder auch Library-Unterteilungen möglich



Components

Container Components

- als Host für andere Komponenten
- Logik
- Datenaustausch mit API / Services
- Datenweitergabe an Child-Components

Presentational Components

- keine externen Abhängigkeiten!
- nur Logik, die in dieser Komponente benötigt wird
- Kommunikation mit dem Container über @Input- und @Output-Properties
- Wiederverwendbar



Component vs. Service

Component

sollte nur die Logik für die Komponente selbst enthalten

Service

- übergeordnete Logik (Ablauf, clientseitige Geschäftslogik usw.)
- API-Aufrufe, Datenverwaltung
- Shell-Zugriffe (Header, Footer, Navigationsleiste etc.)
- sonstige Logik



Layout konzipieren

- Shell für gemeinsamen Rahmen
 - Menü, Statuszeile, Navigationsleiste, Content-Bereich(e) usw.
 - Service für allgemeinen Zugriff
 - Theme einbinden / definieren (durchgängige Gestaltung von Farben, Schriften, Größen usw.)
 - allgemeingültige Styles definieren



Vereinfachungen für Pfade

- tsconfig.json
 - über baseUrl und paths eindeutige, kurze Namen für lange Pfade definieren
 - (wird in VS-Code zum Teil nicht richtig angezeigt)
- index.ts-Dateien f
 ür Module
 - hier Exports zusammenfassen, um alle Exports unter einem einzigen Pfad bereit zu stellen