Motivation Angular

Alex Thalhammer

Outlook

- Overview & Motivation TypeScript

- Overview & Motivation Angular

Outline

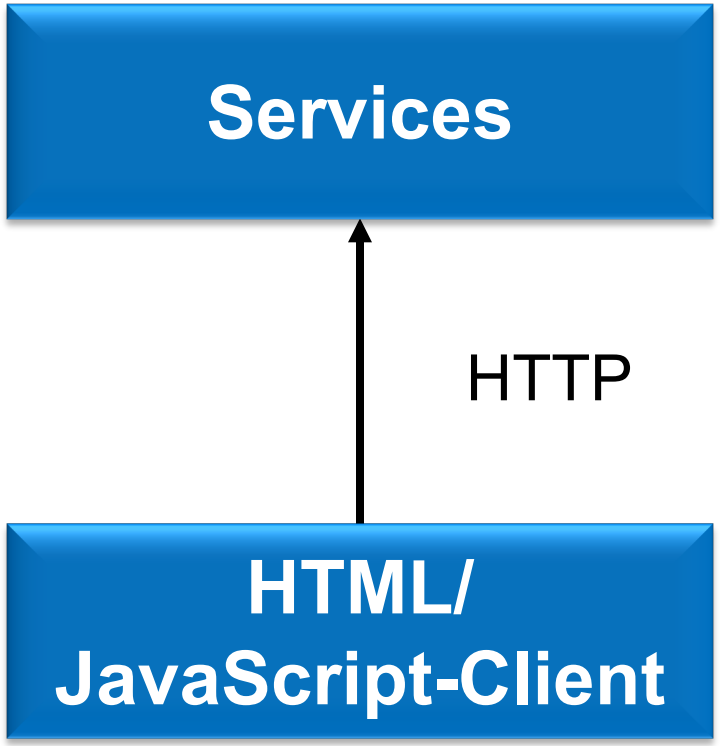
- Motivation
- Components
- Services
  - HTTP Client
- Directives
- Pipes

Motivation

Platforms & Usability

HTML + JavaScript

Single Page Application (SPA)




HTML + JavaScript =
Complexity

Frameworks make SPA manageable

Google

Community

Milions of Devs

Advantages (compared to other frameworks)

- Feature-rich, everything built-in, CLI, ...
- Clear separation of HTML, (S)CSS & TS
- Obligation for TypeScript
- Support for libraries, modules, schemantics & many more
- In our oppinion best choice for Enterprise Applications
- Comparison of frameworks: https://youtu.be/watch?v=lYWYWyX04JI

Angular Speedrun ;-)

Angular Components

AppComponent

```
@Component({
    selector: 'flug-app',
    templateUrl: './app.component.html'
})
export class AppComponent {
    title = 'Hallo Welt!';
}
```

AppComponent

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

@Component({
    selector: 'flug-app',
    templateUrl: './app.component.html'
})
export class AppComponent {
    title = 'Hallo Welt!';
}
```

Lib

E.g: @angular/core

Or own project

E.g.: ../entities/flight

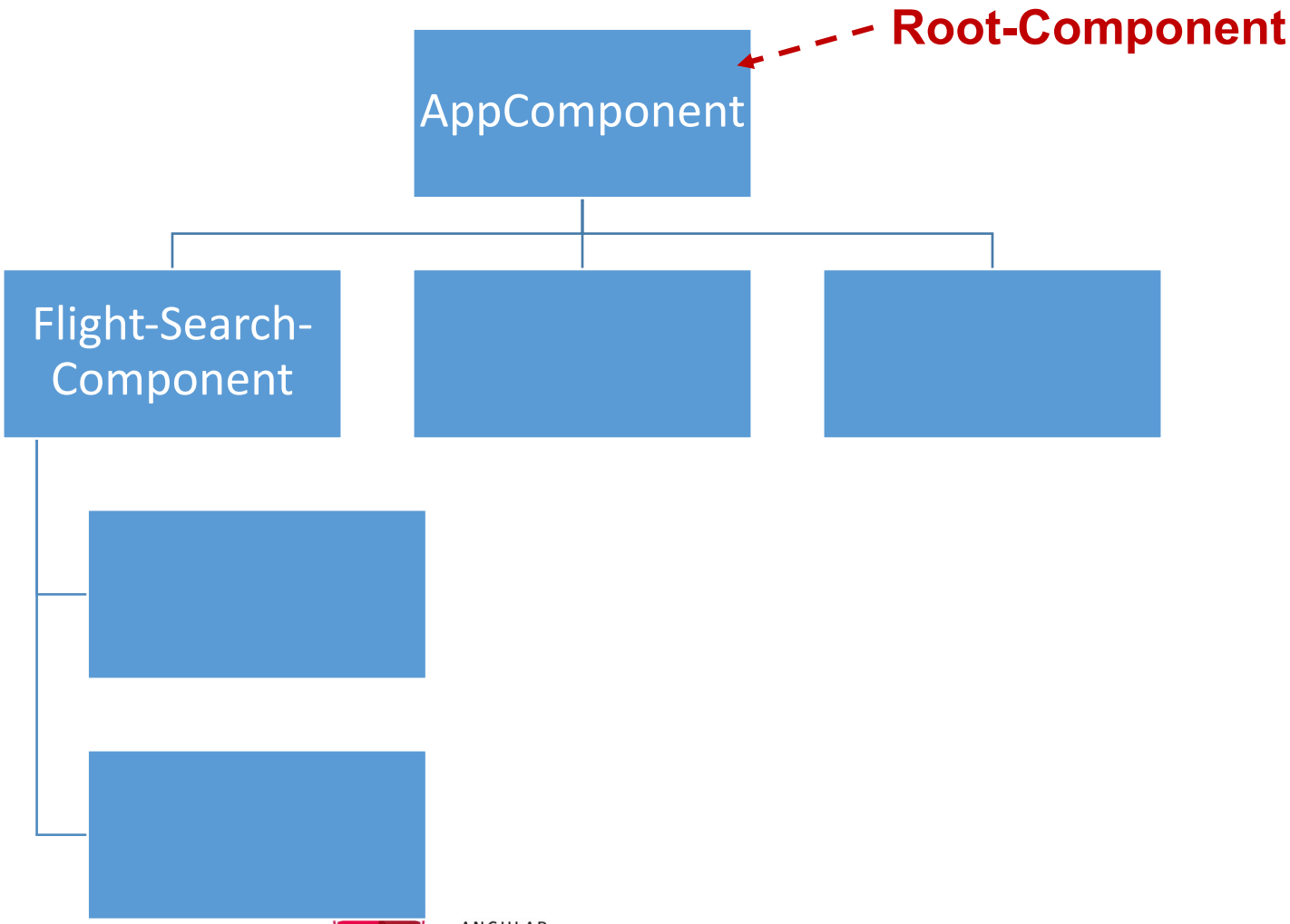
No ending .ts

AppComponent

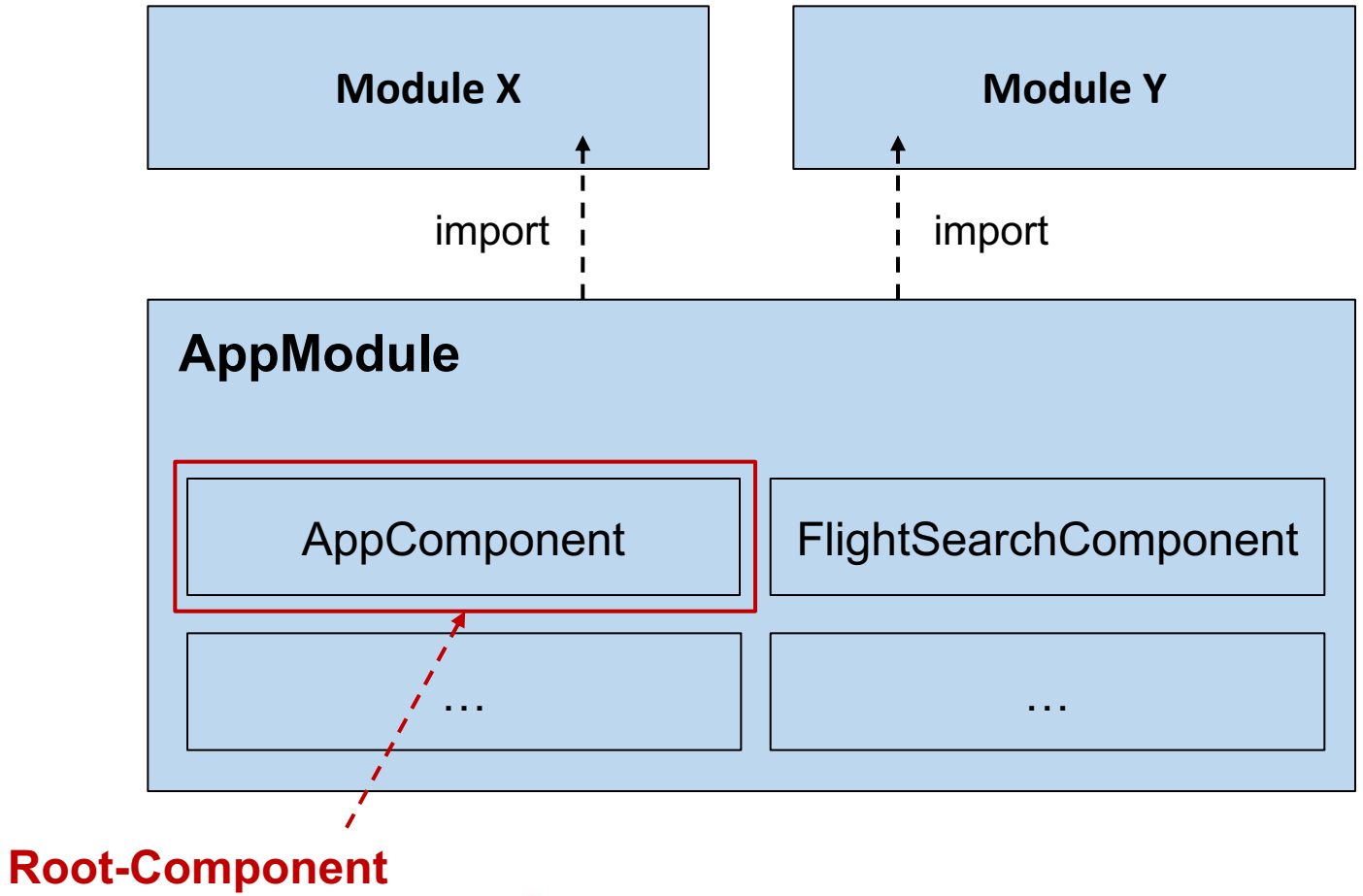
```
import { Component } from '@angular/core';

@Component({
    selector: 'flug-app',
    templateUrl: './app.component.html'
})
export class AppComponent {
    title = 'Hallo Welt!';
}
```

```
<h1>{{title}}</h1>
<div class="container">
    <flight-search></flight-search>
</div>
```

application == component tree



Module



AppModule

```
@NgModule({
    imports: [
       BrowserModule, HttpClientModule, FormsModule
    ],
    declarations: [
       AppComponent, FlightSearchComponent
    ],
    bootstrap: [
       AppComponent
    ]
})
export class AppModule {
}
```

Componente as TypeScript class

```
@Component({
    selector: 'app-flight-search',
    templateUrl: './flight-search.html'
})
export class FilghtSearchComponent {

    from: string;
    to: string;
    flights: Flight[];

    search(): void { [...] }
    select(flight: Flight): void { [...] }
}
```

HTML Template

Two-Way-Binding

```
<input [(ngModel)]="from">
<input [(ngModel)]="to">

<button [disabled]="!from || !to" (click)="search()">
    Search
</button>

<table>
    <tr *ngFor="let flight of flights">
        <td>{{flight.id}}</td>
        <td>{{flight.date}}</td>
        <td>{{flight.from}}</td>
        <td>{{flight.to}}</td>
    </tr>
</table>
```

Event-Binding

Property-Binding

Template

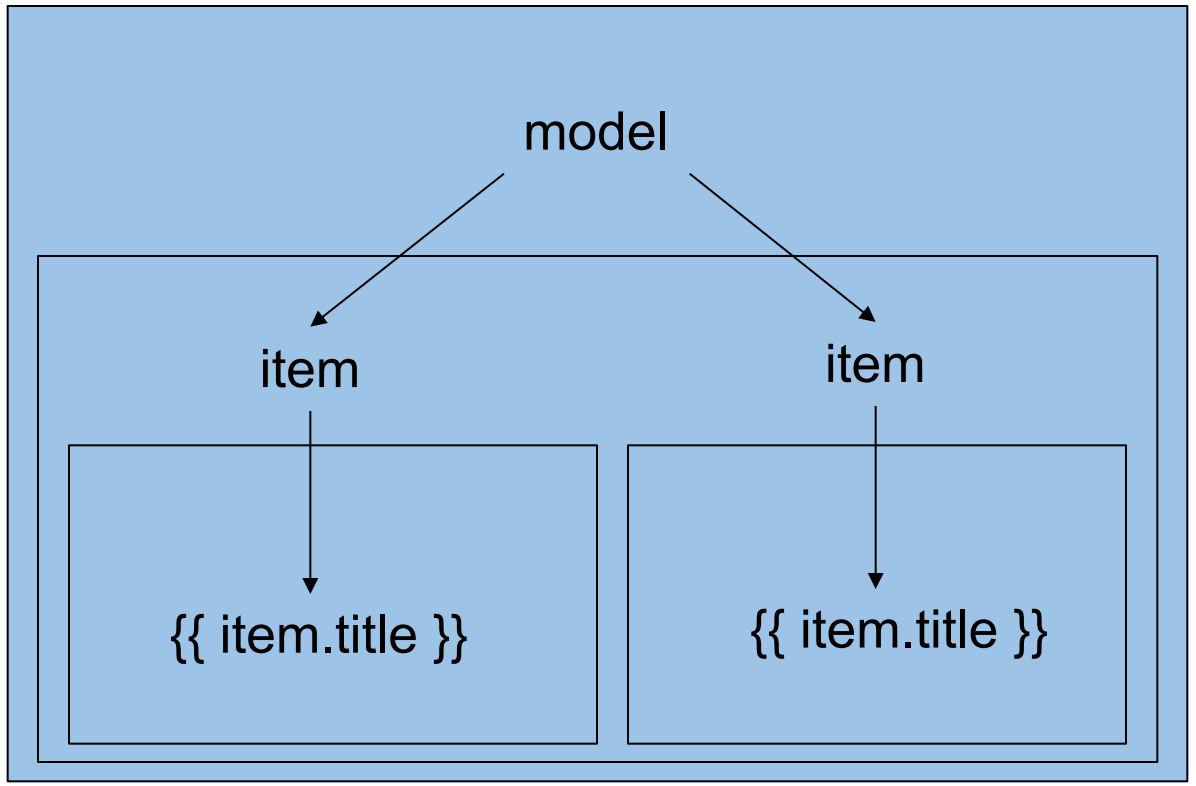
Angular Components – Data Binding

Property-Binding (one-way: top down)

- Data is passed from top to down
  - Parent sets input of children
  - Children cannot pass data to parent

- Component tree

Property-Binding



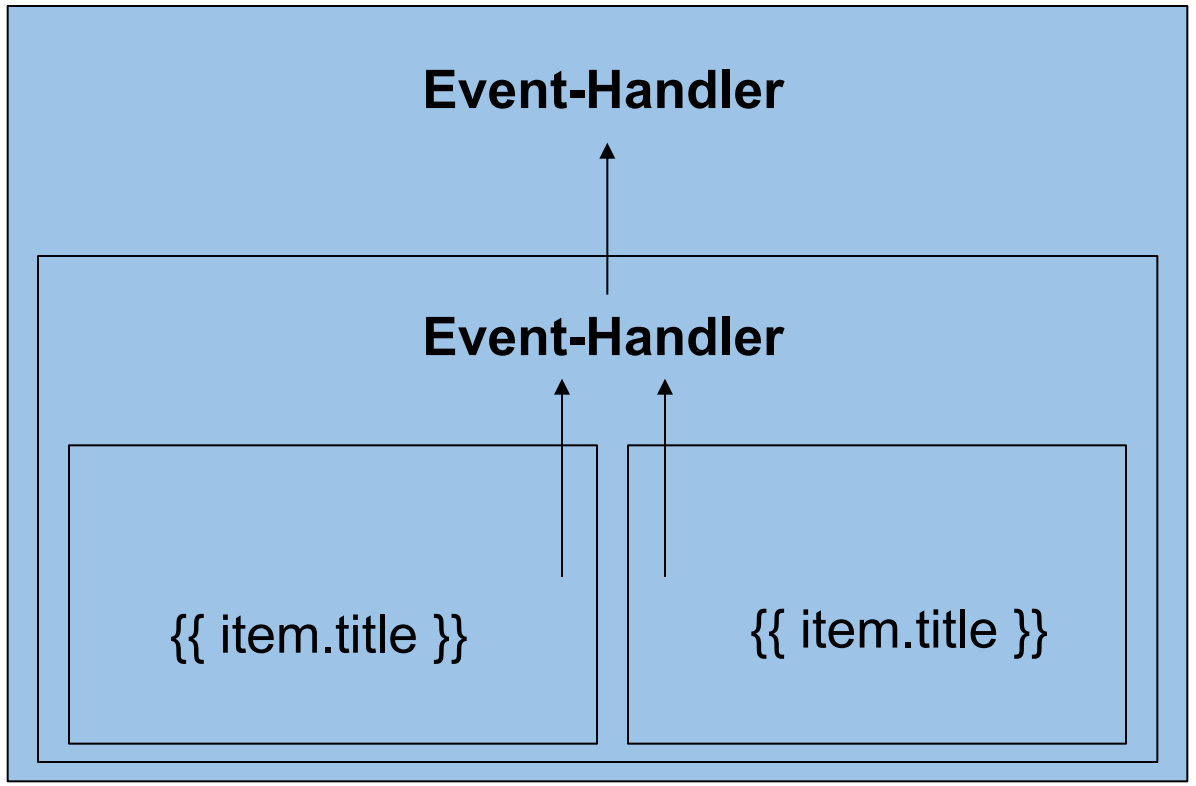
[http://victorsavkin.com/post/110170125256/change-detection-in-angular-2]

Event-Bindings (One-Way, Bottom/Up)

- Communication to parent is handled via event emitters

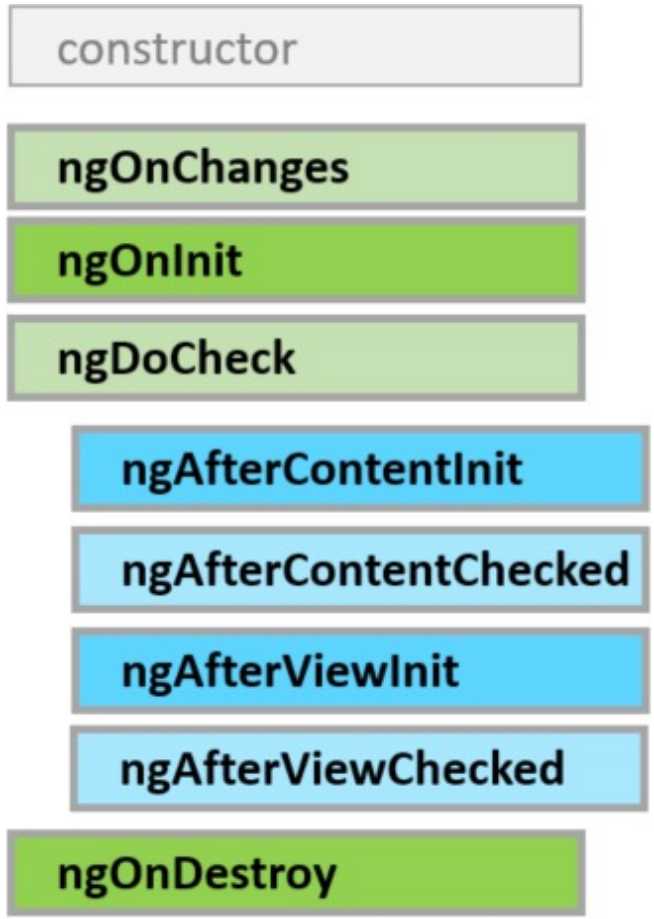
- Events can trigger data change → Property Binding

Event-Bindings (one-way, bottom up)



Angular Component Lifecycle Hooks

Angular Component Lifecycle Hooks



Angular Services

What are services?

Reusable functionality

Injectable

Testable

Classes

E.g.: HttpClient

HttpClient

- get<T>(url, options)
- post<T>(url, body, options)
- put<T>(url, body, options)
- delete<T>(url, options)
- …

HttpClient injection

```
@Component({
    selector: 'flug-suchen',
    templateUrl: './flug-suchen.html'
})
export class FlugSuchenComponent {

    von: string;
    nach: string;
    fluege: Array<Flug>;

    constructor(http: HttpClient) { […] }

    search(): void { [...] }
    select(flug: Flug): void { [...] }
}
```

HttpClient usage

```
let url = 'https://www.angular.at/api/flight';
```

HttpClient usage

```
let url = 'https://www.angular.at/api/flight';

let params = new HttpParams()
                        .set('from', this.from)
                        .set('to', this.to);
```

HttpClient usage

```
let url = 'https://www.angular.at/api/flight';

let params = new HttpParams()
                        .set('from', this.from)
                        .set('to', this.to);

this.http
    .get<Flight[]>(url, { params: params })
```

HttpClient usage

```
let url = 'https://www.angular.at/api/flight';

let params = new HttpParams()
                        .set('from', this.from)
                        .set('to', this.to);

this.http
    .get<Flight[]>(url, { params: params })
```

HttpClient usage

```
let url = 'https://www.angular.at/api/flight';

let params = new HttpParams()
                        .set('from', this.from)
                        .set('to', this.to);

this.http
    .get<Flight[]>(url, { params: params })
    .subscribe(
        function(flights) { […] }
    );
```

HttpClient usage

```
let url = 'https://www.angular.at/api/flight';

let params = new HttpParams()
                        .set('from', this.from)
                        .set('to', this.to);

let that = this;
this.http
    .get<Flight[]>(url, { params, headers })
    .subscribe(
        function(flights) {
              that.flights = flights;
        }
);
```

HttpClient usage

```
let url = 'https://www.angular.at/api/flight';

let params = new HttpParams()
                        .set('from', this.from)
                        .set('to', this.to);

this.http
    .get<Flight[]>(url, { params, headers })
    .subscribe(
        flights => {
                this.flights = flights;
        }
);
```

HttpClient usage

```
let url = 'https://www.angular.at/api/flight';

let params = new HttpParams()
                        .set('from', this.from)
                        .set('to', this.to);

this.http
    .get<Flight[]>(url, { params, headers })
    .subscribe(
        flights => { this.flights = flights; },
        err => { console.error('Loading error', err); } // ToDo proper handling
    );
```

HttpClient usage

```
let url = 'https://www.angular.at/api/flight';

let params = new HttpParams()
                        .set('from', this.from)
                        .set('to', this.to);

this.http
    .get<Flight[]>(url, { params, headers })
    .subscribe(
        flights => { this.flights = flights; },
        err => { console.error('Fehler beim Laden', err); }
    );
```

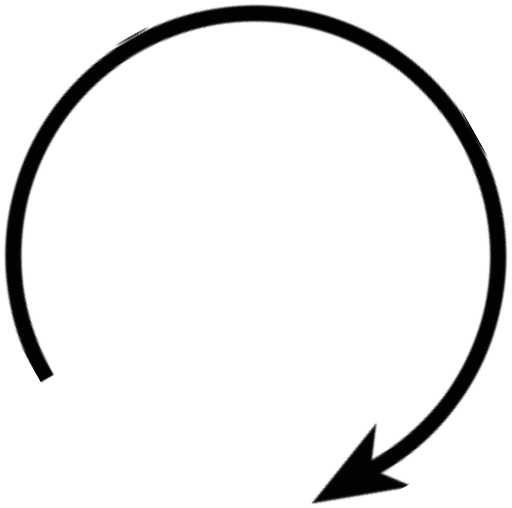
<- - - - - - - - Observable

Observable
„Source“

Operator
(z. B. map)

Observer
„Destination“

Observable



Observable

```
.subscribe(
    (result) => { … },
    (error) => { … },
    () => { … }
);
```

Observer

Angular Directives

What are directives?

- Add special behaviour to HTML Template
- Attribute directive:
  - <input **[(ngModel)]**="from">

- Structural directive (*ngIf, *ngFor, *ngSwitch)
  - <div ***ngFor**="let flight of flights; let index = index; let first = first; let last = last">…

Examples

```
<tr *ngFor="let flight of flights">
    <td>{{flight.id}}</td>
</tr>
```

```
<table *ngIf="flights.length > 0">
…
</table>
```

```
<tr [ngClass]="{ 'active': flight === selectedFlight }">
…
</tr>
```

```
<tr [class.active]="flight === selectedFlight">
</tr>
```

```
<tr [ngStyle]="{ 'background-color':
     (flight === selectedFlight) ?
     'orange' : 'blue' }">
</tr>
```

Custom Attribute Directives

aInteger

aNumber

integer.directive.ts

```ts
import { Directive, ElementRef, HostListener, Input } from '@angular/core';

@Directive({
  // tslint:disable-next-line:directive-selector
  selector: '[aInteger]'
})
export class IntegerDirective {
  constructor(protected el: ElementRef) {}

  @Input() knappInteger: boolean;

  @HostListener('keydown', ['$event']) onKeydown(event: KeyboardEvent): void {
    if (this.knappInteger && !this.isValidIntegerInputKeydown(event)) {
      event.preventDefault();
    }
  }

  protected isValidIntegerInputKeydown(event: KeyboardEvent): boolean {
    return (
      // Allow: Backspace, Tab, Enter, Escape, Delete
      event.key === 'Backspace' ||
      event.key === 'Tab' ||
      event.key === 'Enter' ||
      event.which === 13 || // .which needed for IE11
      event.key === 'Escape' ||
      event.which === 27 ||
      event.key === 'Delete' ||
      event.which === 46 ||
      // Allow: Ctrl+A, Ctrl+X, Ctrl+C, Ctrl+V & Ctrl+Z
      ((event.ctrlKey || event.metaKey) &&
        (event.code === 'KeyA' ||
          event.which === 65 ||
          event.code === 'KeyC' ||
          event.which === 67 ||
          event.code === 'KeyV' ||
          event.which === 86 ||
          event.code === 'KeyX' ||
          event.which === 88 ||
          event.code === 'KeyZ' ||
          event.which === 90)) ||
      // Allow: Home, End, ArrowLeft, ArrowRight
      event.key === 'Home' ||
      event.which === 36 ||
      event.key === 'End' ||
      event.which === 35 ||
      event.key === 'ArrowLeft' ||
      event.which === 37 ||
      event.key === 'ArrowRight' ||
      event.which === 39 ||
      // Allow 1-9
      (!event.shiftKey && +event.key >= 1 && +event.key <= 9) ||
      // Allow 0 if not at first pos
      (!event.shiftKey && event.key === '0' && this.el.nativeElement.selectionStart !== 0)
    );
  }
}
```

number.directive.ts

```ts
import { Directive, ElementRef, HostListener, Input } from '@angular/core';

import { IntegerDirective } from './integer.directive';
import { NumberService } from '@knapp/shared/domain';

@Directive({
  // tslint:disable-next-line:directive-selector
  selector: '[aNumber]'
})
export class NumberDirective extends IntegerDirective {
  constructor(protected el: ElementRef, private numberService: NumberService) {
    super(el);
  }

  @Input() knappNumber: boolean;

  @HostListener('keydown', ['$event']) onKeydown(event: KeyboardEvent): void {
    if (this.knappNumber && !this.isValidNumberInputKeydown(event)) {
      event.preventDefault();
    }
  }
```

```ts
private isValidNumberInputKeydown(event: KeyboardEvent): boolean {
  const natEl = this.el.nativeElement;
  const value = natEl.value;
  const decSep = this.numberService.getDecimalSeparator();
  const decSepRegEx = decSep === '.' ? /\./ : new RegExp(decSep);

  // If has minus disallow numbers at first pos
  if (
    value.substring(0, 1) === '-' &&
    natEl.selectionStart === 0 &&
    natEl.selectionEnd === 0 &&
    (event.key === '0' || (+event.key >= 1 && +event.key <= 9))
  ) {
    return false;
  }

  return (
    // Allow: one Minus only at first pos
    ((event.key === '-' || event.code === 'Minus') &&
      natEl.selectionStart === 0 &&
      (value.substring(natEl.selectionEnd).match( matcher: /-/g) || []).length === 0) ||
    // Allow: one Decimal Separator
    (event.key === decSep &&
      (value.substring(0, natEl.selectionStart).match(decSepRegEx) || []).length +
        (value.substring(natEl.selectionEnd).match(decSepRegEx) || []).length ===
        0) ||
    // Allow: 0 even at first pos
    (!event.shiftKey && event.key === '0') ||
    // Allow: All other integer inputs (inherited)
    this.isValidIntegerInputKeydown(event)
  );
}
```

number.directive.ts – usage

```
<input type="text" class="form-control has-clear-span"
       knappNumber
```

Angular Pipes

Build-in Pipes

DatePipe

UpperCasePipe

LowerCasePipe

DecimalPipe

CurrencyPipe

PercentPipe

JsonPipe

Custom Pipes (examples)

formattedValue

formattedNumber

formatted-number.pipe.ts

```ts
@Pipe({ name: 'formattedNumber' })
export class FormattedNumberPipe implements PipeTransform {
  constructor(private numberService: NumberService) {}

  transform(value: number | string): string {
    if (value == null) {
      return null;
    }

    if (this.numberService.isValidNumberValue(value)) {
      return this.roundAndReplaceDecimalSeparator(Number(value));
    } else if (this.numberService.isValidPercentageValue(value)) {
      // remove percentage sign, convert to number and add sign again
      const valueString = value.toString();
      const percentageString = valueString.substring(0, valueString.length - 1);
      return this.roundAndReplaceDecimalSeparator(Number(percentageString)) + '%';
    }

    return value.toString();
  }
```

formatted-number.pipe.ts – usage

```
<span>
  {{ value|formattedNumber }}
</span>
```

Angular 3rd party packages

Angular 3rd party packages – HowTo

- Think about prefering Angular Material / CDK
- Use the big ones (downloads, stars, updates, ...)
- Typically increase their versions with NG

Angular 3rd party packages – My Favs

- https://ngrx.io/
- https://github.com/ngx-translate/core
- **https://github.com/swimlane/ngx-datatable**
- https://github.com/ng-select/ng-select
- https://github.com/KoderLabs/ngx-device-detector

- Many component libs

Angular Best Practices

Recommendations on Angular

- Evergreen policy! (Update on a regular basis)
- Wait for packages to do their updates
- Follow the Angular Style Guide

  https://angular.io/guide/styleguide
- Add an own styleguide for html and scss

HowTo Learn This Thing?

- The fast: Our workshops
  https://www.angulararchitects.io/angular-schulung/

- The cheap: Max Schwarzmüller on Udemy (that's how I did it in '16)
  https://www.udemy.com/course/the-complete-guide-to-angular-2/

- Other
  - Official Docs https://angular.io/docs
  - Manfred's books: https://www.angulararchitects.io/book/

Now you know Angular

- Use it and you’re gonna love it
  - If you are a Web Developer

- Any questions left over? ☺