



# Angular: First steps

Hosted by Alex Thalhammer

# Outline

- Motivation
- First steps
- Your first component
- HTTP client



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# Motivation



# Platforms and Usability



HTML + JavaScript

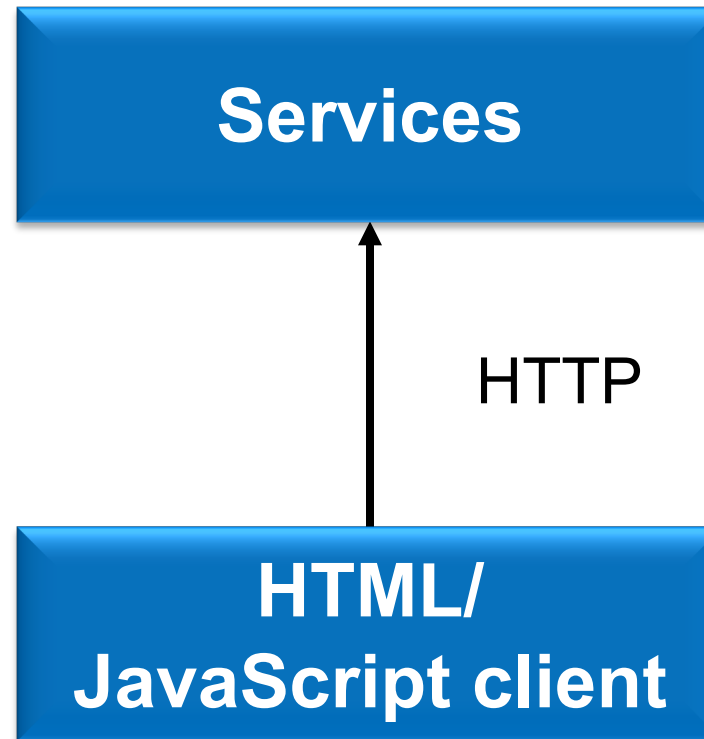



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# Single Page Application (SPA)





HTML + JavaScript =  
Complexity





Frameworks make SPA manageable



Google

Community

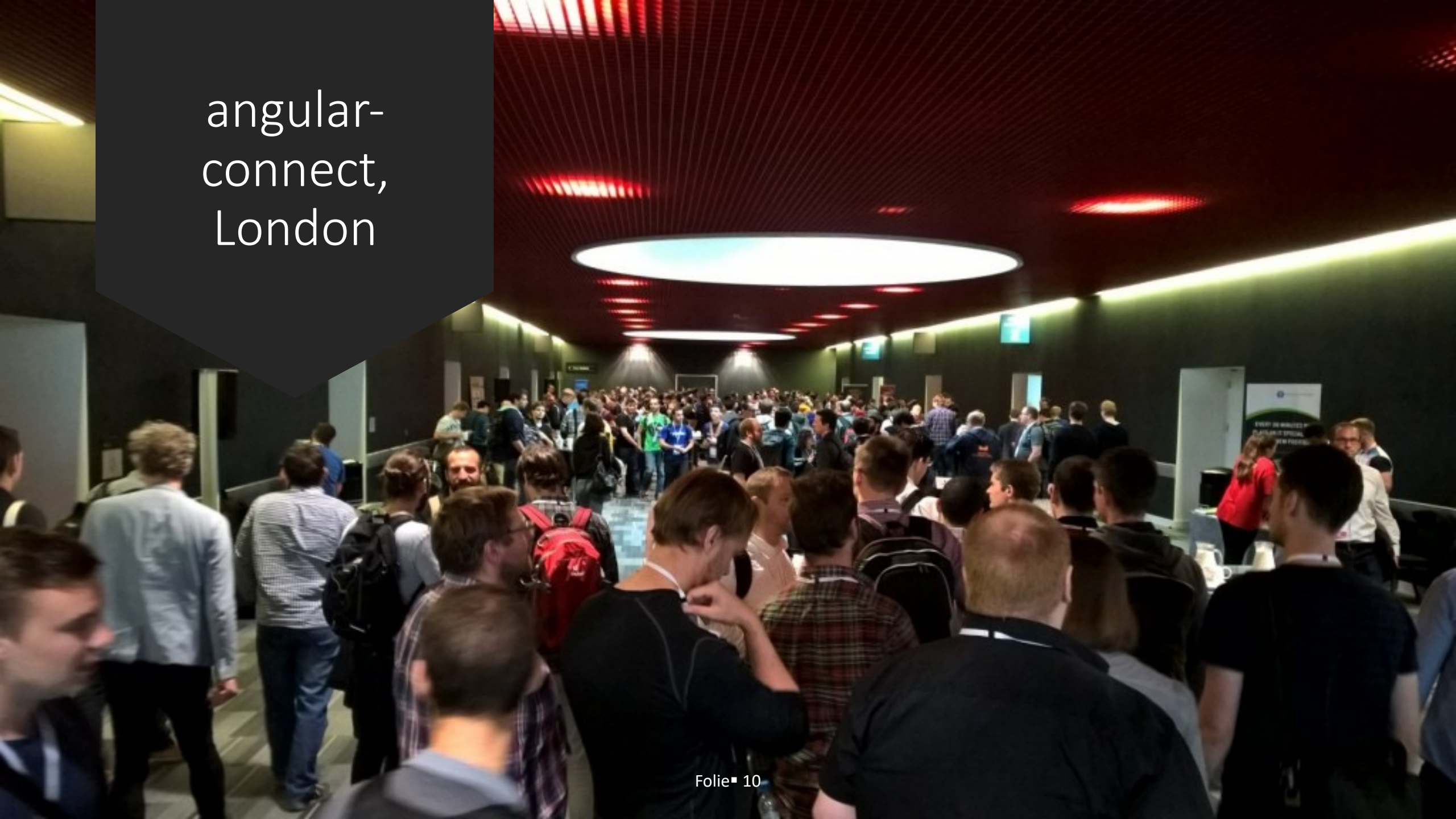
Angular  
>2M Devs



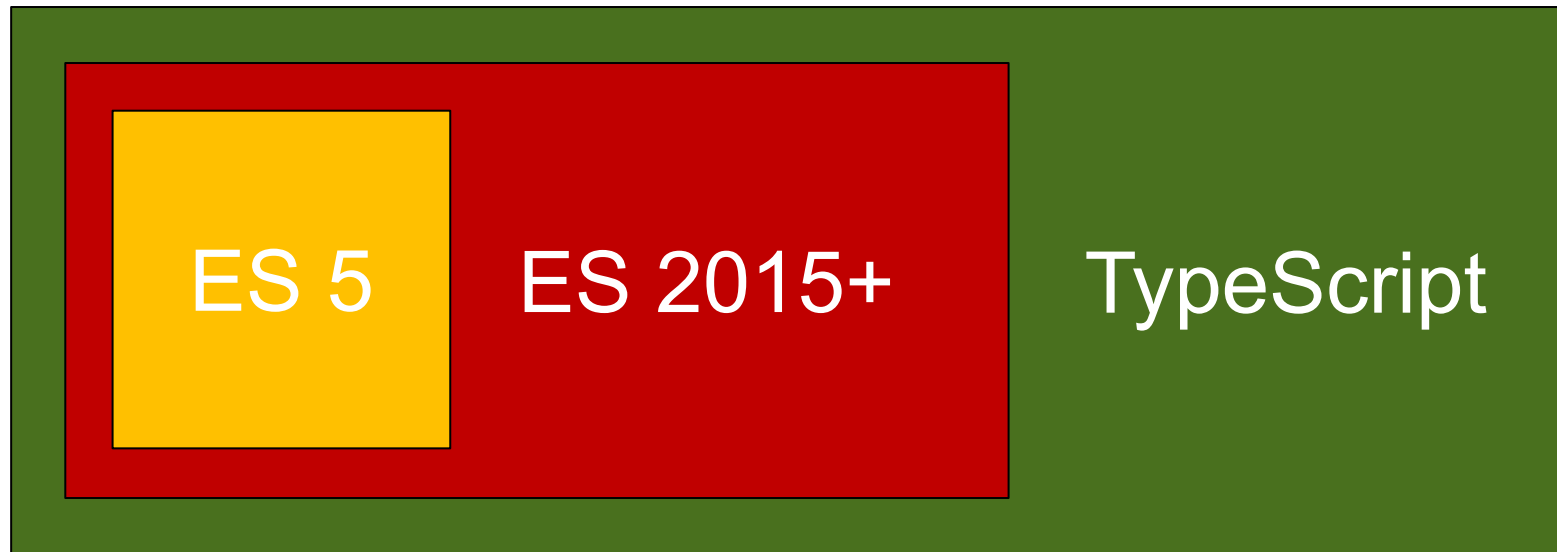
# angular- connect, London



# angular- connect, London



# JavaScript vs TypeScript



←  
compilation





# First steps with Angular

# AppComponent

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



# AppComponent

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

```
@Component({  
  selector: 'flight-app',  
  templateUrl: './app.component.html'  
})
```

```
export class AppComponent {  
  title = 'Hello World!';  
}
```

## Library

E.g.: @angular/core

## Own project

E.g.: ../entities/flight  
No ending ".ts"



# AppComponent

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

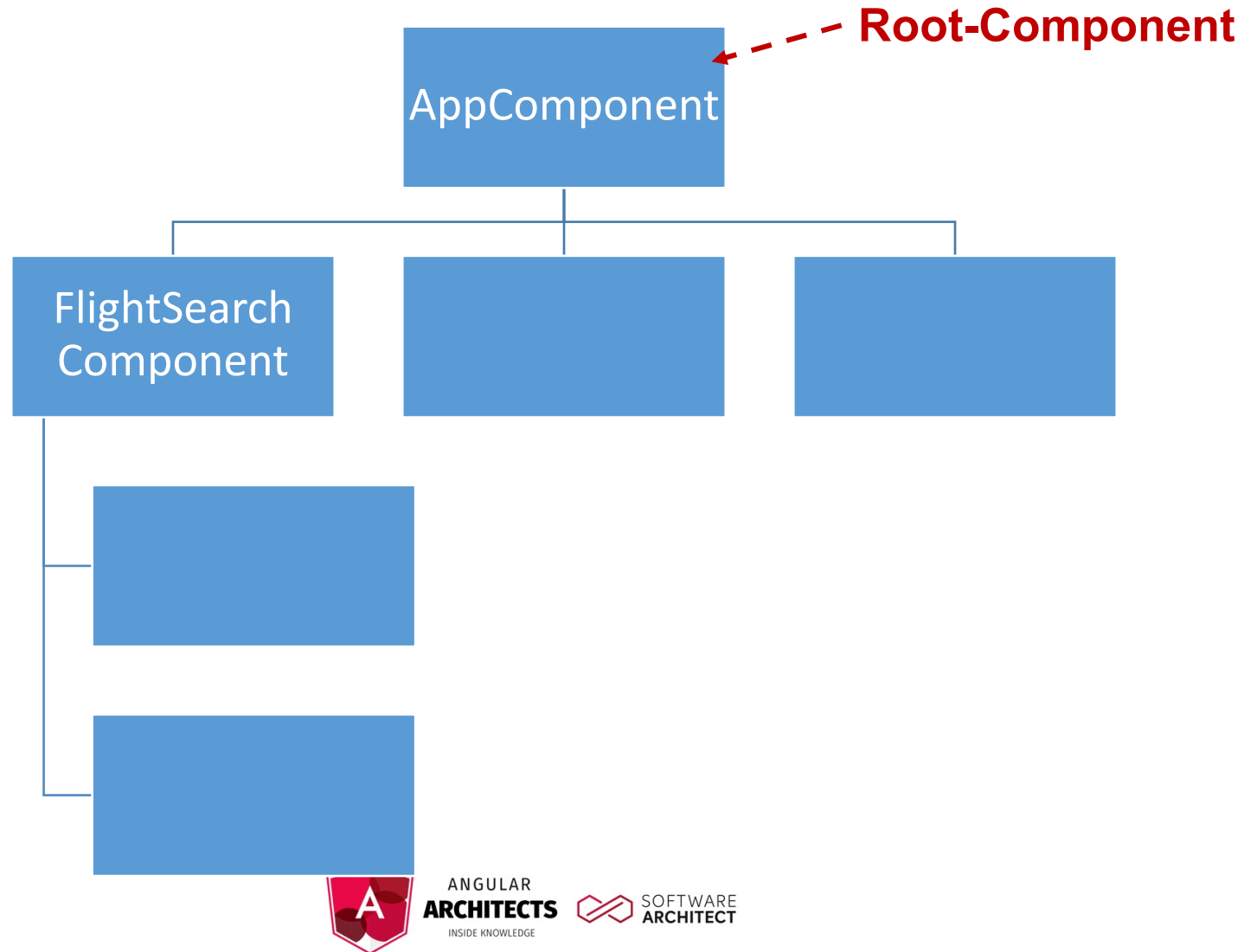
@Component({
  selector: 'flight-app',
  templateUrl: './app.component.html'
})
export class AppComponent {
  title = 'Hello World!';
}
```

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

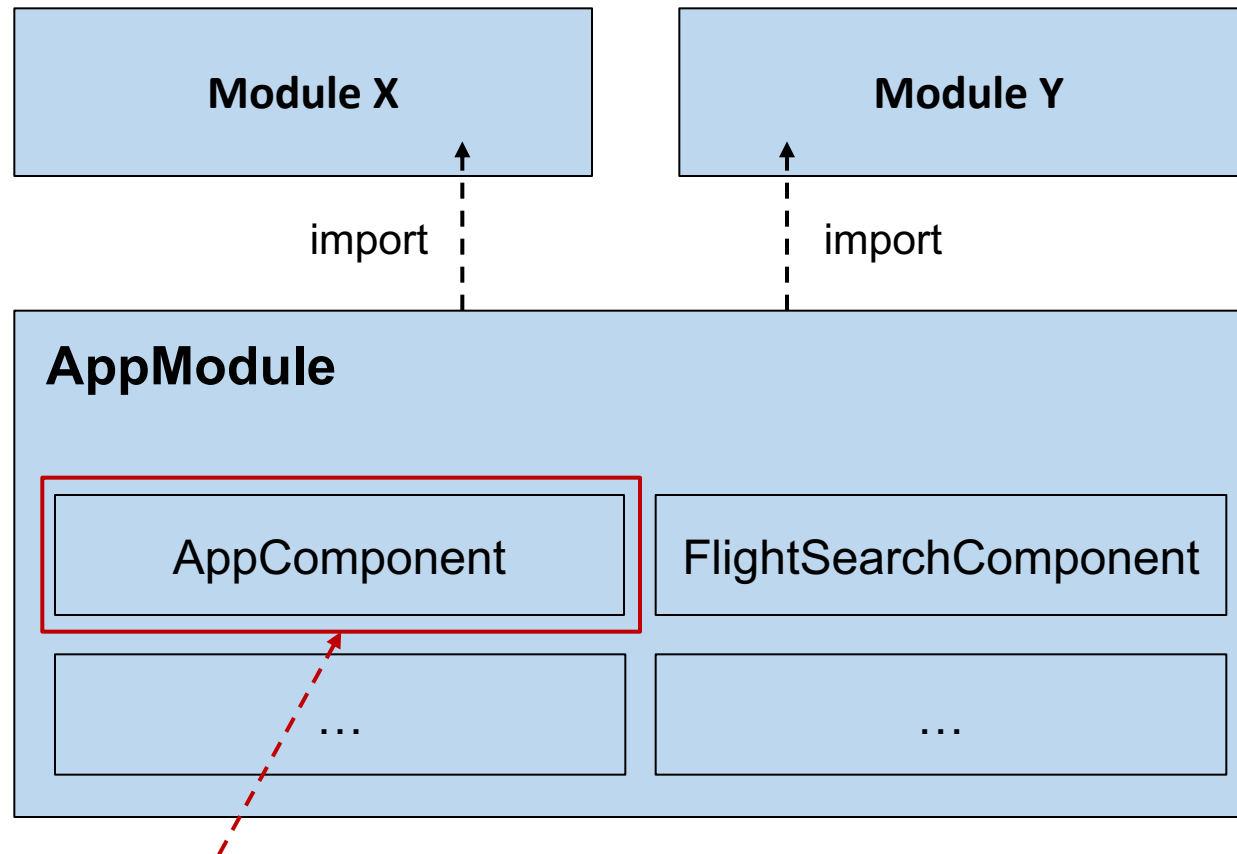




# App == component tree



# Module



**Root-Component**

# AppModule

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



# Bootstrapping

- Start Angular
- Load RootModule/AppModule with RootComponent/AppComponent



# Bootstrapping

```
platformBrowserDynamic().bootstrapModule(AppModule);
```



# index.html

[...]

<body>

<flight-app></flight-app>

<script src="..."></script>

</body>

[...]



# Start new project

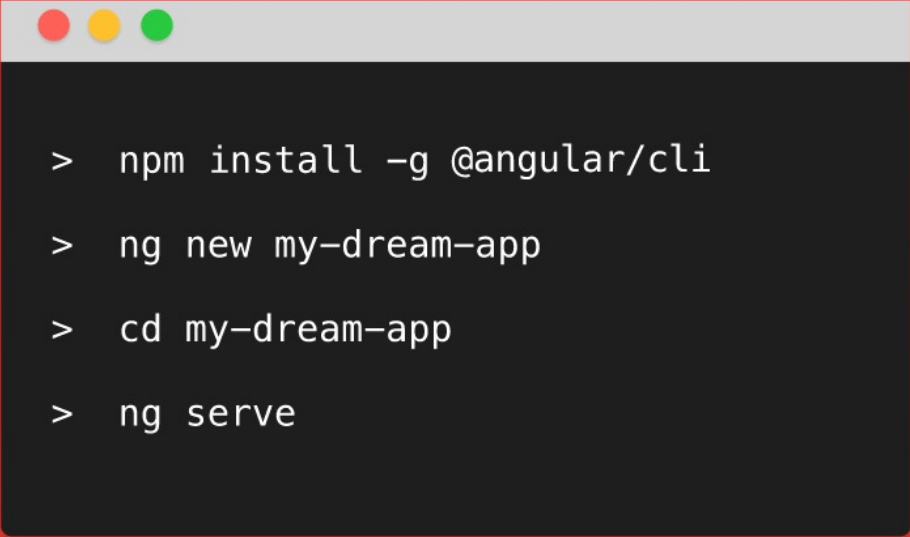


ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**





```
> npm install -g @angular/cli  
> ng new my-dream-app  
> cd my-dream-app  
> ng serve
```

# Angular CLI

A command line interface for Angular

GET STARTED

## Angular CLI

# Our Starterkit

- `ng new starter`
- `cd starter`
- `npm i bootstrap --save`



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# DEMO



# LAB



# Let's get it on

- Pull the repo <https://github.com/L-X-T/uni-muenster>
- Please get started with lab 00\_getting\_started
- Yarn (or npm i) and then Yarn start (or npm start)
- Take a closer look at the starter kit
- Optional: You may add the plugins mentioned



# Your first component

# Component as TypeScript class

```
@Component({  
  selector: 'flight-search',  
  templateUrl: './flight-search.html'  
})  
export class FlightSearchComponent {  
  
  from: string;  
  to: string;  
  flights: Flight[];  
  
  search(): void { [...] }  
  select(flight: Flight): void { [...] }  
}
```





# Template

Two Way Binding

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

Event (/Output) Binding

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

Property (/Input) Binding

```
<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>
```

Template



# DEMO



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**



Access HTTP resources

# HttpClient

- `get(url, options)`
- `post (url, body, options)`
- `put(url, body, options)`
- `delete(url, options)`
- ...



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# HttpClient

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



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# Inject HttpClient

```
@Component({  
    selector: 'flight-search',  
    templateUrl: './flight-search.html'  
})  
export class FlightSearchComponent {  
  
    from: string;  
    to: string;  
    flights: Flight[];  
  
    constructor(http: HttpClient) { [...] }  
  
    search(): void { [...] }  
    select(flight: Flight): void { [...] }  
}
```



# Use HttpClient (I)

```
const url = 'http://www.angular.at/api/flight';
```



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**



# Use HttpClient (II)

```
const url = 'http://www.angular.at/api/flight';  
  
const params = new HttpParams().set('from', this.from).set('to', this.to);
```



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# Use HttpClient (III)

```
const url = 'http://www.angular.at/api/flight';  
  
const params = new HttpParams().set('from', this.from).set('to', this.to);  
  
const headers = new HttpHeaders().set('Accept', 'application/json');
```



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# Use HttpClient (IV)

```
const url = 'http://www.angular.at/api/flight';

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

const headers = new HttpHeaders().set('Accept', 'application/json');

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



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE



SOFTWARE  
ARCHITECT

# Use HttpClient (V)

```
const url = 'http://www.angular.at/api/flight';  
  
const params = new HttpParams().set('from', this.from).set('to', this.to);  
  
const headers = new HttpHeaders().set('Accept', 'application/json');  
  
this.http.get<Flight[]>(url, { params, headers })
```

**Short hand**



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE



SOFTWARE  
ARCHITECT

# Use HttpClient (VI)

```
const url = 'http://www.angular.at/api/flight';

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

const headers = new HttpHeaders().set('Accept', 'application/json');

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



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# Use HttpClient (VII)

```
const url = 'http://www.angular.at/api/flight';

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

const headers = new HttpHeaders().set('Accept', 'application/json');

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



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE



SOFTWARE  
ARCHITECT

# Use HttpClient (VIII)

```
const url = 'http://www.angular.at/api/flight';

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

const headers = new HttpHeaders().set('Accept', 'application/json');

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



# Use HttpClient (IX)

```
const url = 'http://www.angular.at/api/flight';

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

const headers = new HttpHeaders().set('Accept', 'application/json');

this.http.get<Flight[]>(url, { params, headers })
  .subscribe({
    next: (flights) => { this.flights = flights; },
    error: (err) => { console.error('Error loading', err); }
  });
```



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**



# DEMO



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# LAB



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# Use HttpClient (X)

```
const url = 'http://www.angular.at/api/flight';

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

const headers = new HttpHeaders().set('Accept', 'application/json');

this.http.get<Flight[]>(url, { params, headers })
  .subscribe({
    next: (flights) => { this.flights = flights; },
    error: (err) => { console.error('Error loading', err); }
  });
```

←----- **Observable**



Observable  
„Quelle“



Operator  
(z. B. map)

Observer  
„Senke“

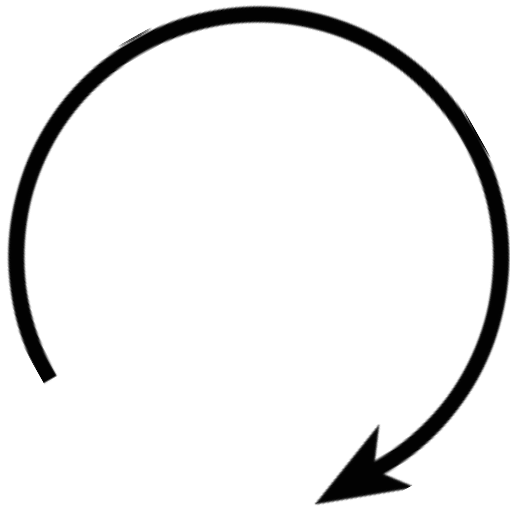


ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# Observable



Observable

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

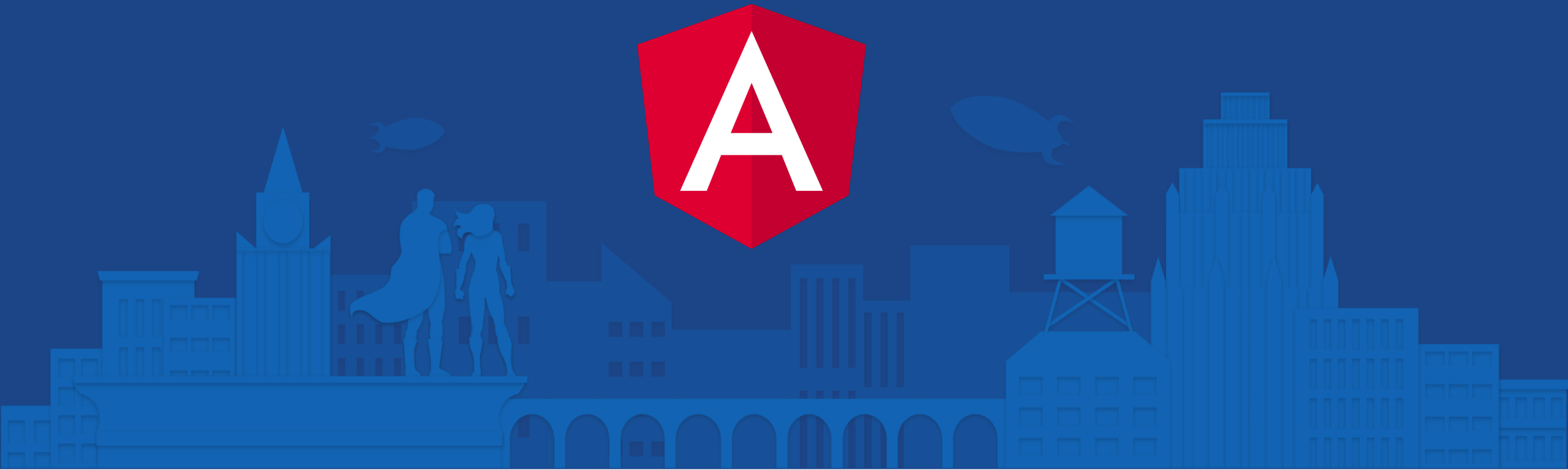
Observer



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**



Use Angular Directives

# What are Directives?

- Add behaviour to html elements
- Are used as html attributes
- Examples:
  - `<input [(ngModel)]="from">`
  - `<div *ngFor="let flight of flights">...</div>`

# Types of Directives?

- Structural Directives
  - \*ngIf="statement"
  - \*ngFor="let element of array"
  - \*ngSwitch="something"
- Attribute directives
  - Built-ins
    - [(ngModel)]
    - [ngClass] or [ngStyle]
  - Custom ones



# Examples (I)

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

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

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

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



# Examples (II)

```
<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>
```

