The ultimate Angular Workshop

TechEvent 2017

Thomas Bandixen (@tbandixen)
Thomas Gassmann (@gassmannT)

@gassmannT @tbandixen

Agenda

1. Introduction
2. Data Binding
3. Directives
4. Components
5. Routing
6. Exercise

Introduction

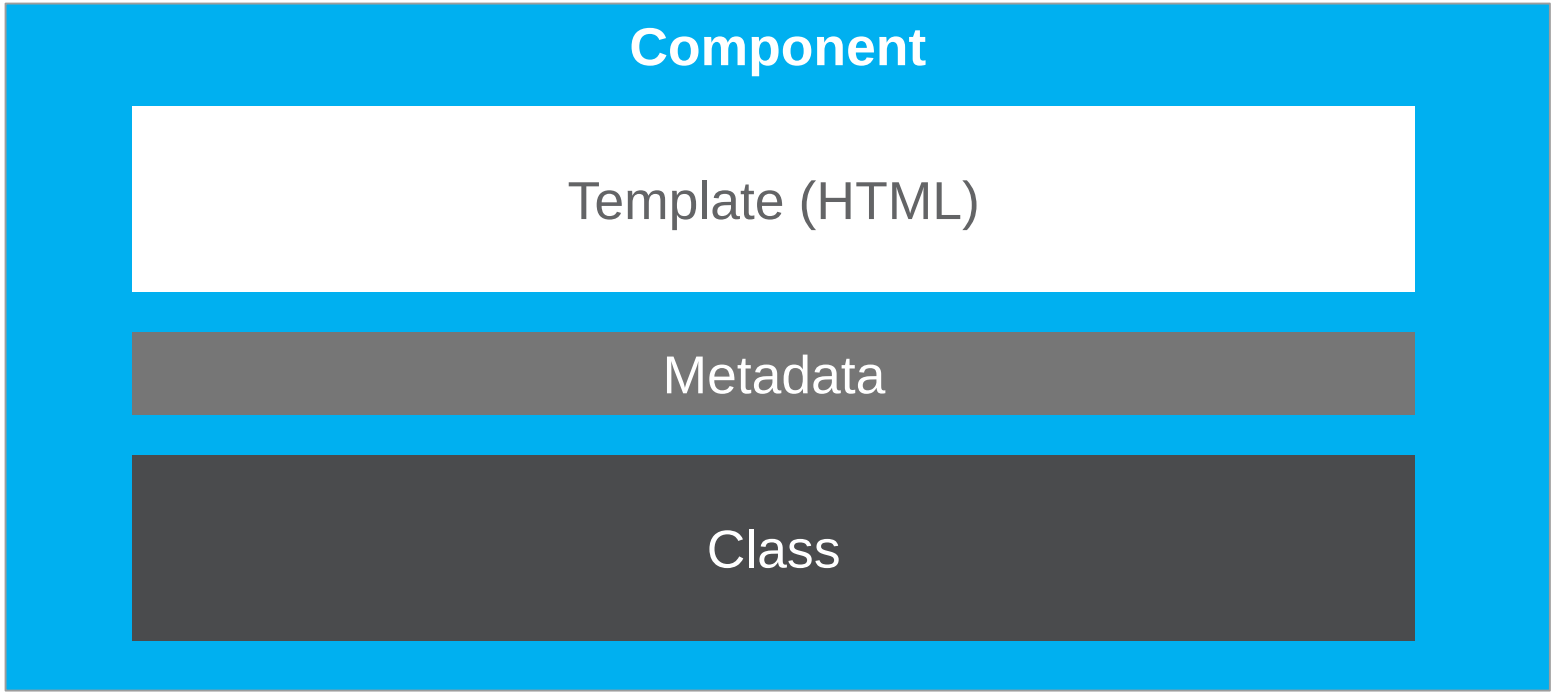
Angular

- Single-Page-Application (SPA) Framework by Google
  - is different to Angular 1.x: No controllers, no $scope
  - Angular is more than a Framework, it’s a plattform
- Completely rebuilt with TypeScript
- Use your HTML, CSS and TypeScript skills to build apps
  - Instead of TypeScript you could also use plain JavaScript or Dart
- Current Version Angular 4.0.0-rc.3

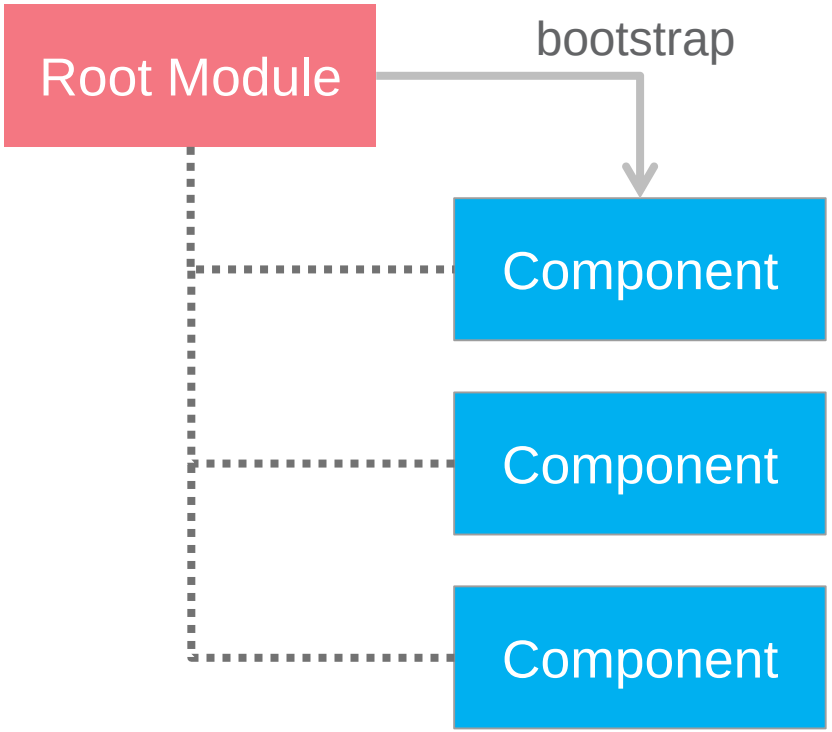
Why should you choose Angular 2 (or >)?

- It’s fast!
- It’s clean!
- It has a powerful data-binding engine
- It is component-based and allows great structuring of apps

Angular Apps are component-based



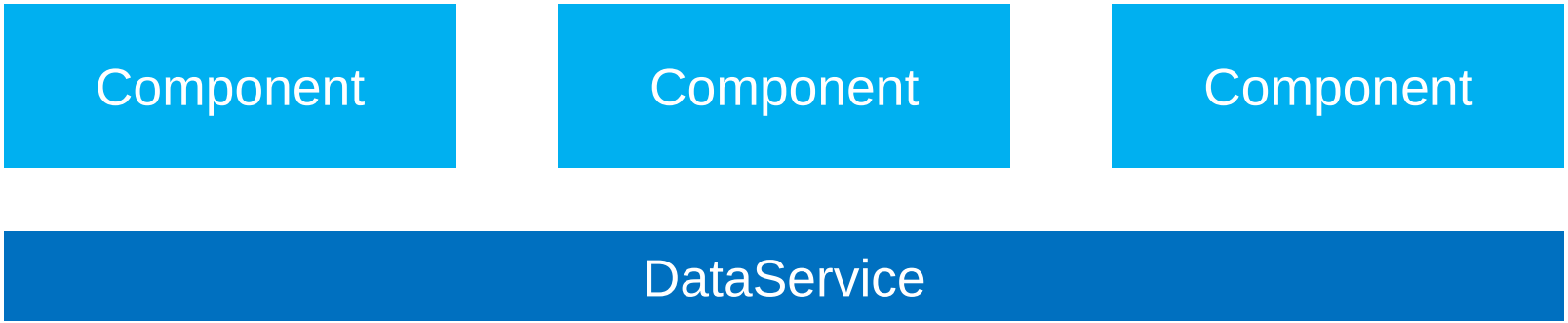
Components are structured into Angular-Modules



- Every Angular App has at least one module, the Root Module
- The module can bundle multiple components
- One Component must be bootstrapped for startup
  - only true for the root module

Beside Components an app uses Services

- A service is a class that gets injected into components
  - for example a class that encapsulates access to a REST-Api



Setting up an Angular application: The Options

- Manually walk through the quickstart and create the needed files:
  - https://angular.io/docs/ts/latest/quickstart.html
- Grab the finished quickstart-app from GitHub
  - https://github.com/angular/quickstart/blob/master/README.md

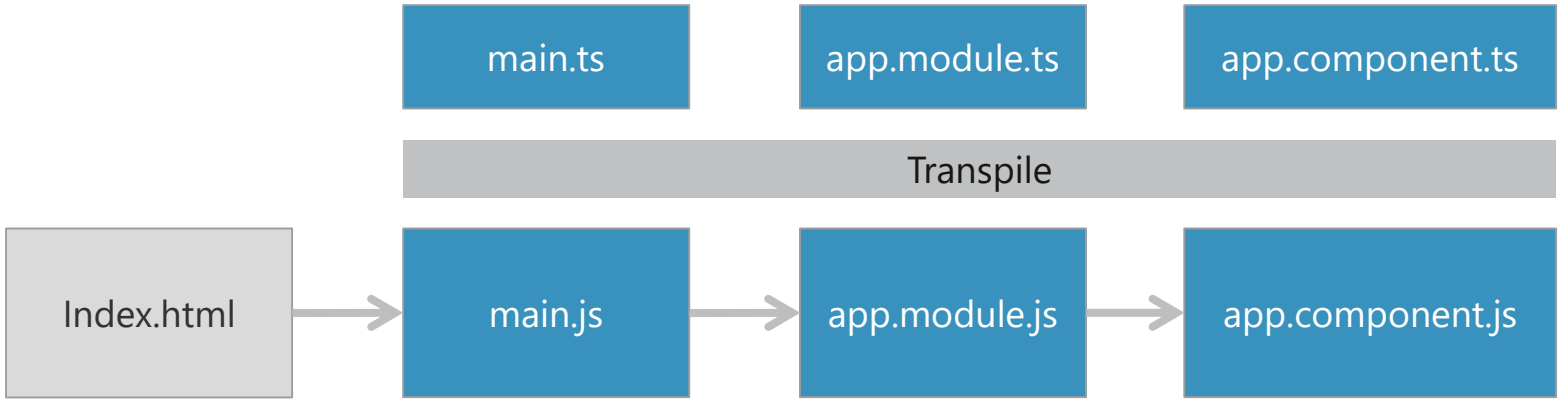
```
git clone  https://github.com/angular/quickstart  my-proj
```

- Use the command line interface: angular-cli
  - https://github.com/angular/angular-cli

```
npm install -g angular-cli
```

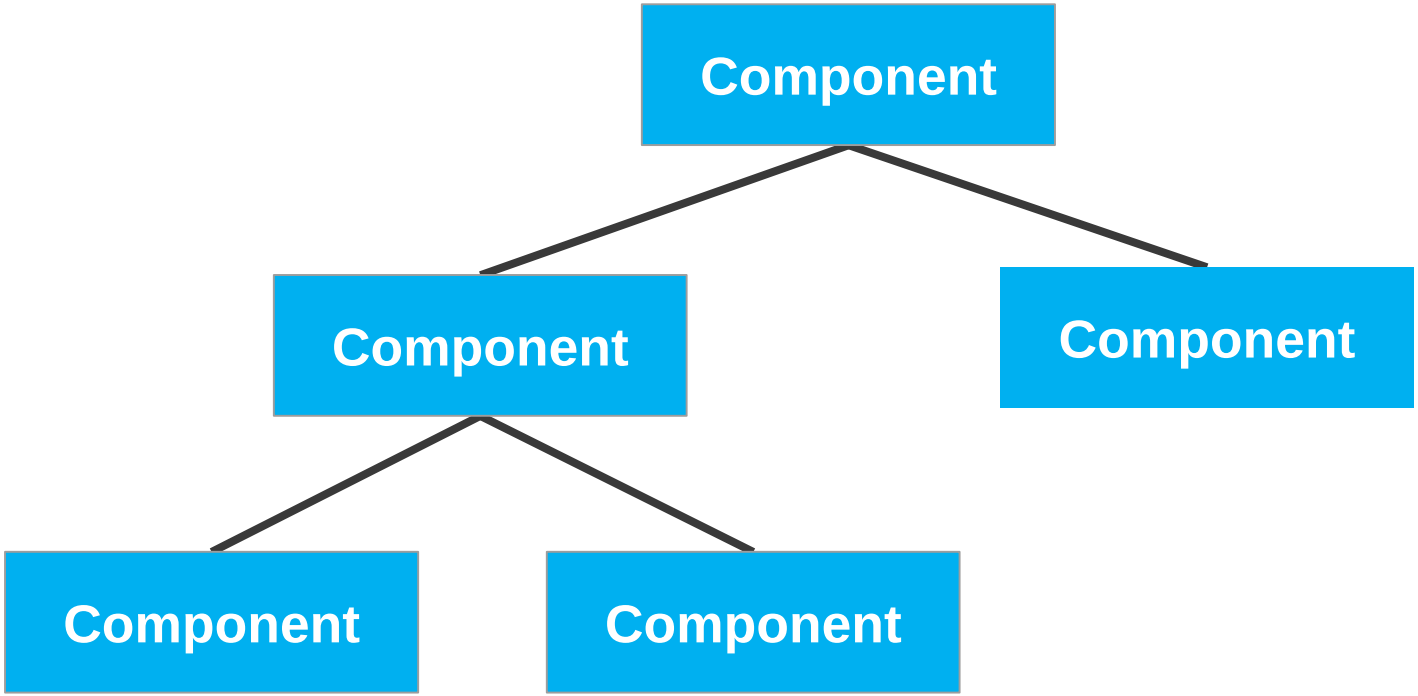
```
ng new my-proj
```

Architecture of the initial app

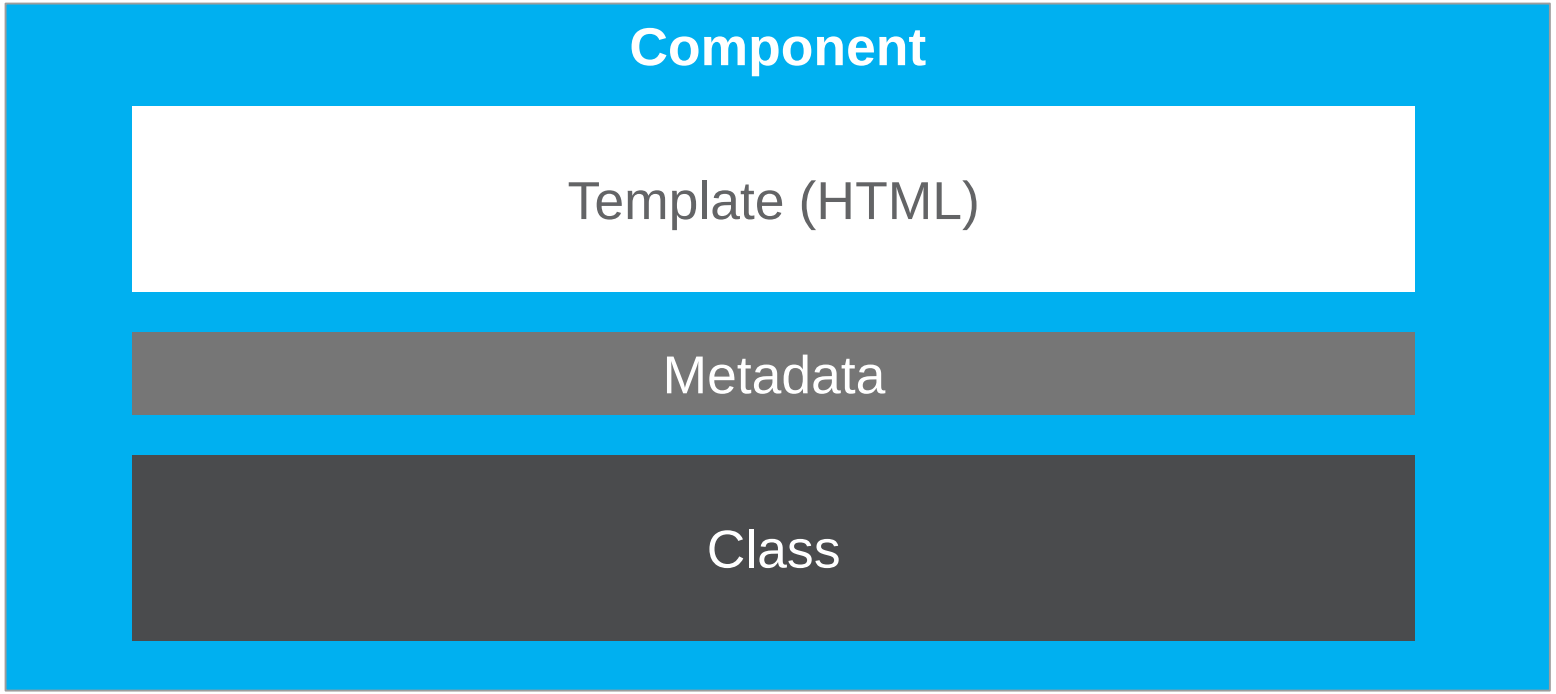


Data Binding

An Angular app is a tree of components



Angular Apps are component-based



The component and its template

A component is a class decorated with the Component-decorator

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

@Component({
  selector: 'my-app',
  template: `<h1>Details of {{fullname}}</h1>`
})
export class AppComponent {
  fullname: string = "Lara Croft";
}
```

Data Binding overview

- **{{}}** => Display the value of a property in the UI
- **[]** => Bind an element-property to a property of your class
- **()** => Bind an element-event to a method of your class
- **[()]** => Bind a property two-way
  - that you don't have to think whether it's **([])** or [()], just use the "banana in a box"-mnemonic to get the syntax right

Simple Rendering with Interpolation

- Use an expression in {{ }}
  - for example {{1 + 1}} will write out 2
- use the pipe iterator to pass the expression to a Pipe
  - you can write your own pipes or use existing pipes like *uppercase*

```
@Component({
  selector: 'my-app',
  template: `<h1>Details of {{fullname | uppercase}}</h1>`
})
export class AppComponent {
  fullname: string = "Lara Croft";
}
```

Binding properties

- use the brackets **[]** to bind to a property
- This creates a OneWay-Data Binding

```
@Component({
  selector: 'my-app',
  template: `<h1>Details of {{fullname | uppercase}}</h1>
             <input type="text" [value]="fullname">`
})
export class AppComponent {
  fullname: string = "Lara Croft";
}
```

Binding events

- Use parentheses **()** to bind events to methods

```
@Component({...,
  template: `...
            <input type="text" [value]="fullname">
            <button (click)="onUpdate()">Update</button>`
})
export class AppComponent {
  fullname: string = "Lara Croft";
  onUpdate()
  {
    this.fullname = "Duke Nukem";
  }
}
```

TwoWay-Data Bindings: Binding to an object

- Just use the dot-syntax to access properties

```
@Component({
  selector: 'my-app',
  template: `<h1>Details of {{person.fullname | uppercase}}</h1>
            <input type="text" [(ngModel)]="person.fullname">`
})
export class AppComponent {
  person: Person = { fullname: "Lara Croft" };
}

interface Person { fullname:string }
```

Add the FormsModule to your AppModule

- This allows you to use ngModel for TwoWay-Data Bindings

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import {FormsModule} from '@angular/forms';
import { AppComponent }   from './app.component';

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

Directives

There are three directives in angular

- Components
  - are referenced by their selector (custom HTML tag)
- Structural Directives
  - are changing the DOM layout by adding/removing DOM elements
- Attribute Directives
  - change the appearance or behavior of an element
  - e.g. ngModel

Structural Directives

- **ngFor** – to generate elements while looping over a collection
- **ngIf** – to show or hide an element with an if-statement
- **ngSwitch** – to show/hide elements based on a condition

ngFor

```
@Component({...
  template: `...
            <table>
              <tr *ngFor="let person of persons">
                <td>{{person.firstname}}</td>
                <td>{{person.lastname}}</td>
                <td>{{person.githubaccount}}</td>
              </tr>
            </table>
             `
})
export class AppComponent {
   persons:Person[]=PERSONS;
}
```

ngIf

- div is only displayed if selectedPerson is not undefined/null

```
@Component({...
  template: `...
            <div *ngIf="selectedPerson">
              ...
            </div>
          `
})
export class AppComponent {
   persons:Person[]=PERSONS;
   selectedPerson:Person; ...
}
```

ngSwitch

- Based on the firstname a div is displayed

```
<div [ngSwitch]="selectedPerson?.firstname">
    <div *ngSwitchCase="'Thomas'">I'm teaching Angular</div>
    <div *ngSwitchCase="'Lara'">I'm playing games</div>
    <div *ngSwitchDefault>I'm just a fallback</div>
</div>
```

Routing

What is Routing?

- Angular is a Single Page Application (SPA) framework
- Somehow users need to navigate
- Navigation means Routing the user to different components

Setting up Routing: Adding the base tag

- **Step 1:** Add the base tag to your index.html-file

```
<head>
  <base href="/">
  ...
</head
```

- Behind the scenes, Angular’s router uses the browser's **history.pushState** for navigation
  - This allows you to make the URL-path look like you want

Setting up Routing: The Service Provider

- **Step 2:** Setup the Router service provider
  - call the RouterModule's forRoot-method to create a provider

```
import { RouterModule } from '@angular/router';
@NgModule({
  imports: [BrowserModule, FormsModule,
   RouterModule.forRoot([
   { path: 'sessions', component: SessionListComponent},
   { path: 'session/edit/:id', component: SessionEditComponent },
   { path: '', redirectTo: '/sessions', pathMatch: 'full'},
   { path: '**', redirectTo: '/', pathMatch: 'full' }
    ])],
  ...
})
export class AppModule { }
```

Setting up Routing: Add the router-outlet component

- **Step 3:** Add the router-outlet component
  - It comes with the router library @angular/router
- This is where the linked components will be displayed

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

@Component({
  selector: 'my-app',
  template: `<h1>Simple Routing</h1>
            <router-outlet></router-outlet>
            `
})
export class AppComponent { }
```

Setting up Routing: Adding links

- **Step 4:** Adding router links in template

```
<a routerLink="/sessions">Sessions</a>

<a [routerLink]="'/sessions'">Sessions</a>

<a [routerLink]="['/session/edit', item.id]">Edit</a>
```

Angular 4

Angular 2 vs. Angular 4

- Performance boost
- Angular Universal
- TypeScript's StrictNullChecks compliancy
- Stand alone animation module
- ngIf supports now else
- SEO optimizations
- …

Angular 2 vs. Angular 4


```
<ng-template #hidden>
  <p>You are not allowed to see our secret</p>
</ng-template>
<p *ngIf="shown; else hidden">
  Our secret is being happy
</p>
```

Angular 2 vs. Angular 4


```
//new in Angular 4 for SEO
this.title.setTitle(`TechEvent - ${this.pageTitle}`);
this.meta.addTag({
    name: "Description",
    content: `TechEvent - ${this.pageTitle}`
});
```

Exercise

Your machine

- Visual Studio Code
  - Microsoft’s powerful cross-platform code editor

  https://code.visualstudio.com/Download
- Node.js and NPM
  - powerful server and package manager

  https://nodejs.org/
- Clone git repository:
  - https://github.com/gassmannT/AngularWorkshop

What to do?

github.com/gassmannT/AngularWorkshop/

- navigate to starter

Fragen?

Thomas Bandixen
thomas.bandixen@trivadis.com

Tel. +41 58 459 52 78

Thomas Gassmann
thomas.gassmann@trivadis.com

Tel. +41 58 459 52 81