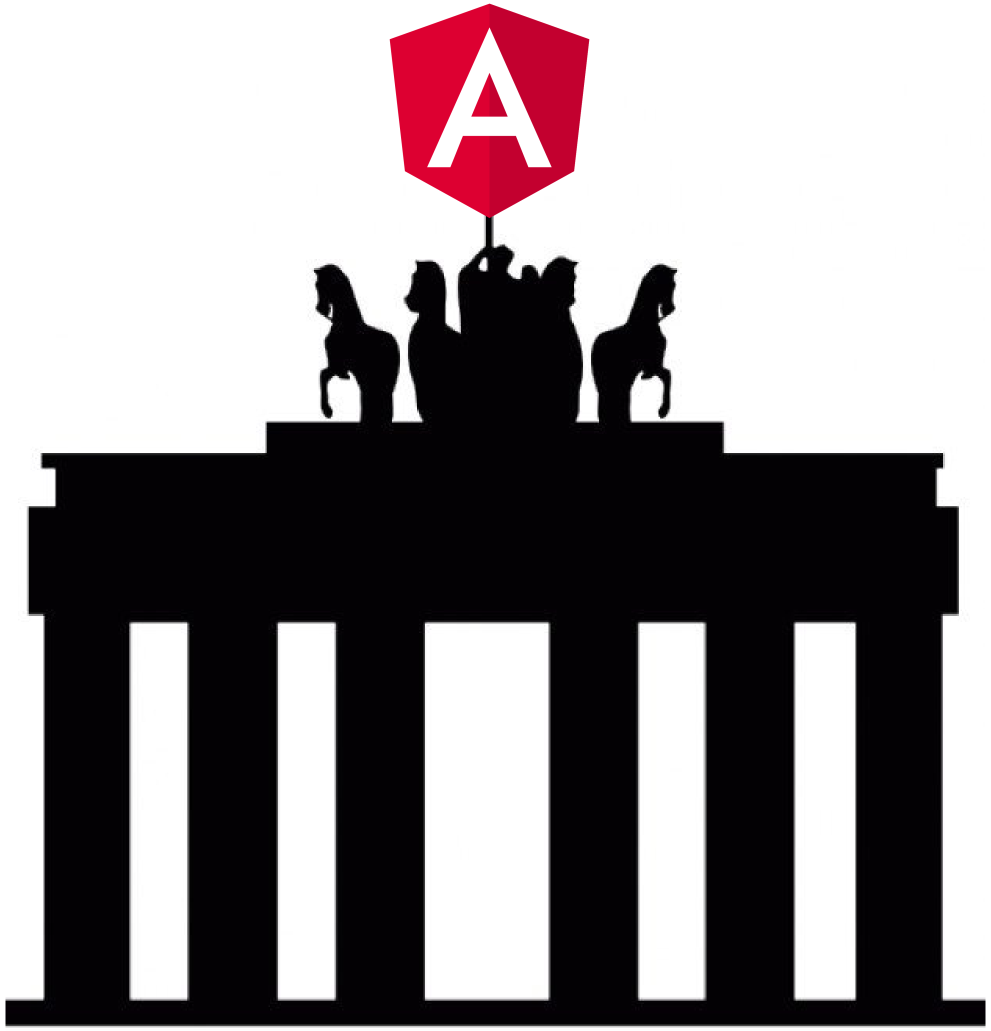
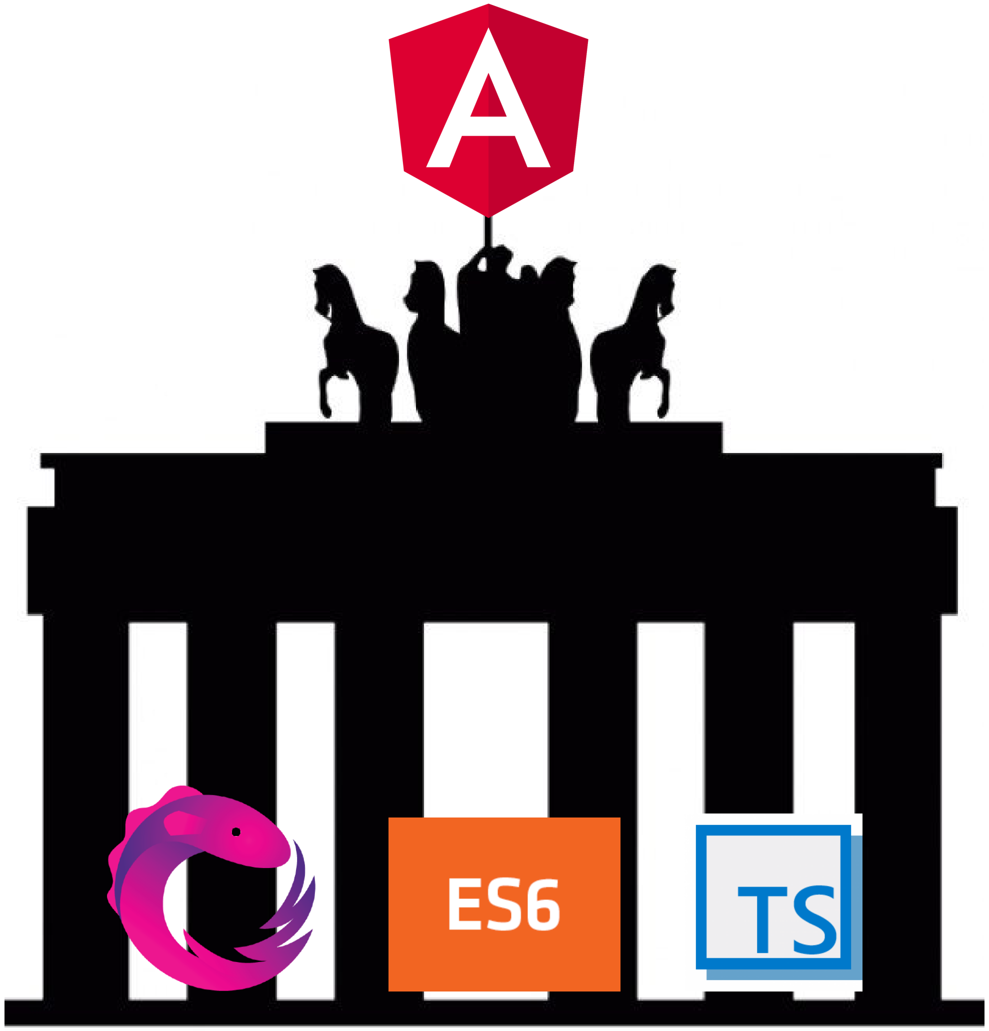
Angular

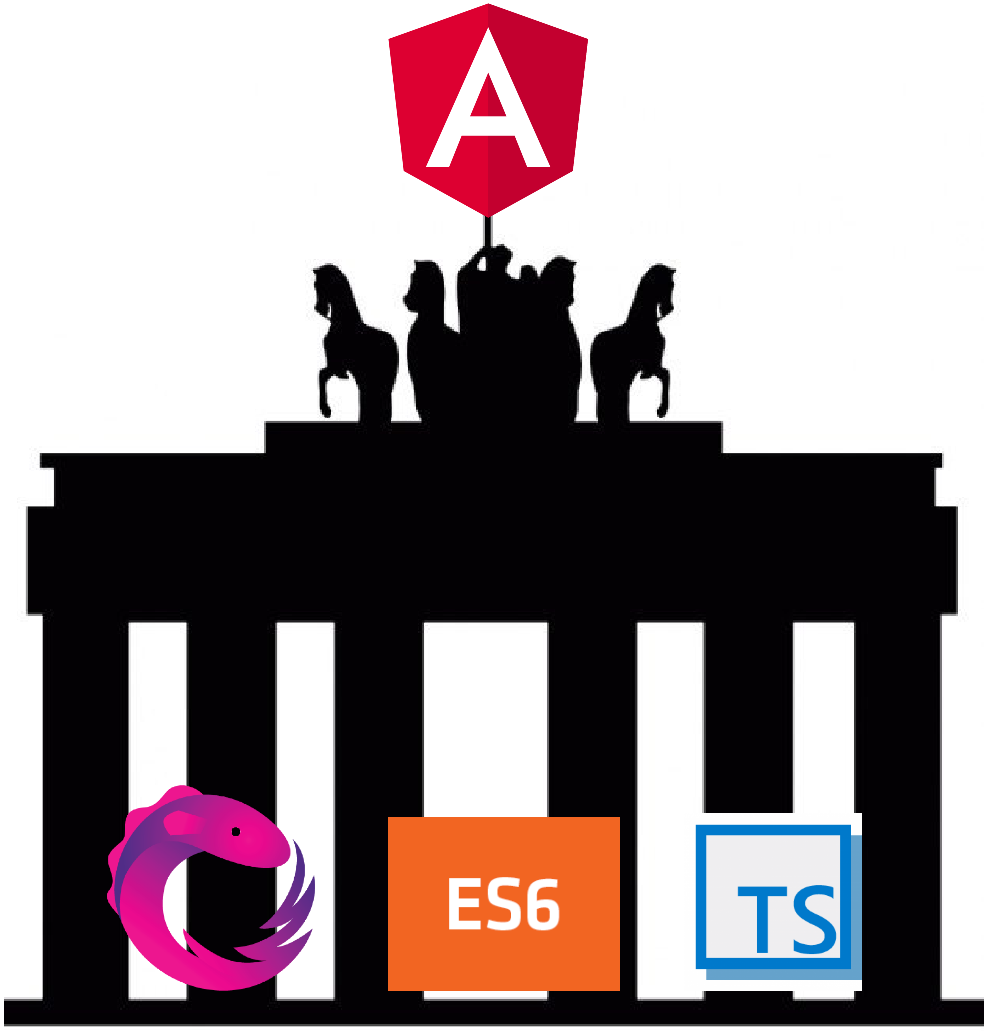
Tomasz Ducin
13-15 February 2019
Wrocław

agenda

- Dependencies
  - zone.js
  - core-js
  - RxJS
  - TypeScript*
- Architecture
  - Component Arch.
  - Reactivity
- Building Blocks
  - Components
  - Services
  - NgModules
  - Pipes
- Usage
- Implementation exercises
- Design exercises
- Real life examples




ES6
TS



zone.js

Dependency Injection

Component Architecture

Change Detection

SETUP

angular CLI

quick guide: cli.angular.io

global install (desync!)

```
npm install -g @angular/cli
ng new <APP>
cd <APP>
// using global `ng`
ng serve
ng generate component <COMP>
```

local install (locked in package.json)

```
// one time install
npx -p @angular/cli ng new <APP>
// local install under `node_modules`
// downside: ng not available in path
// benefit: no desync
cd <APP>
npm start // same as `ng serve`
npm run ng -- generate component <COMP>
```

angular CLI

example

```
ducin@macbook-pro-tomasz angular $ npx -p @angular/cli ng new ng-yo-dawg
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE ng-yo-dawg/README.md (1025 bytes)
CREATE ng-yo-dawg/angular.json (3895 bytes)
CREATE ng-yo-dawg/package.json (1309 bytes)
CREATE ng-yo-dawg/tsconfig.json (435 bytes)
CREATE ng-yo-dawg/tslint.json (2824 bytes)
CREATE ng-yo-dawg/.editorconfig (246 bytes)
CREATE ng-yo-dawg/.gitignore (576 bytes)
CREATE ng-yo-dawg/src/favicon.ico (5430 bytes)
CREATE ng-yo-dawg/src/index.html (295 bytes)
CREATE ng-yo-dawg/src/main.ts (372 bytes)
...
added 1200 packages from 1190 contributors and audited 40178 packages in 18.523s
found 1 high severity vulnerability
  run `npm audit fix` to fix them, or `npm audit` for details
    Successfully initialized git.
```

angular CLI

docs: angular.io/cli

```
npm run ng -- generate module employees --routing
npm run ng -- g m employees --routing
// or without --routing
```

```
npm run ng -- generate component employee-list
npm run ng -- g c employee-list
```

```
npm run ng -- generate service employees/EmployeeModel
npm run ng -- g s employees/EmployeeModel
  --module=employees/employees.module
```

CONVENTIONS

app-* prefix

- **angular.json**: projects/<name>/prefix
- restricted during compilation

```
// declaration
@Component({
  selector: 'app-my-component'
})
export class MyComponent {
  ...
}


// usage
<app-my-component></app-my-component>
```

ANGULAR & TYPESCRIPT

TS decorators

```
@Input()
@Output()
@NgModule()
@Component()
@Directive()
@Injectable()
@Inject()
@ViewChild()
@ViewChildren()
@HostListener()
```

ANGULAR
& RxJS

event emitters

(RxJS Subjects)

```
export class MyComponent {
  @Output()
  someEvent: EventEmitter<MyType> = new EventEmitter();

  imperativeMethod() {
    this.someEvent.emit(item) // ng-way
    // or
    this.someEvent.next(item) // rx-way
  }
}
```

ngrx effects

(RxJS: stream-in, stream-out)

```
@Injectable()
export class AuthEffects {
  constructor(
    private http: Http,
    private actions$: Actions
  ) { }

  @Effect() login$ = this.actions$
      // Listen for the 'LOGIN' action
      .ofType('LOGIN')
      // Map the payload into JSON to use as the request body
      .map(action => JSON.stringify(action.payload))
      .switchMap(payload => this.http.post('/auth', payload)
        // If successful, dispatch success action with result
        .map(res => ({ type: 'LOGIN_SUCCESS', payload: res.json() }))
        // If request fails, dispatch failed action
        .catch(() => Observable.of({ type: 'LOGIN_FAILED' }))
      );
}
```

common problem

HOT vs COLD observables

e.g. HttpClient returns cols observables

common mistake

multiple subscriptions with Async Pipe

```
@Component({
  template: `<div>data: {{ data$ | async }}</div>
             <div>data: {{ data$ | async }}</div>`
})
export class MyComponent {
  data$: Observable<number> = ...
}
```

ANGULAR CORE BB

components

```
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs';

import { Employee } from '../typings';
import { EmployeeModel } from 'src/app/employees/employee-model.service';

@Component({
  selector: 'app-employees-list',
  templateUrl: './employees-list.component.html',
  styleUrls: ['./employees-list.component.css']
})
export class EmployeesListComponent implements OnInit {
  employees$: Observable<Employee[]>

  constructor(
    private employeeModel: EmployeeModel
  ) {}

  ngOnInit() {
    this.employees$ = this.employeeModel.getEmployees()
  }
}
```

components lifecycle hooks

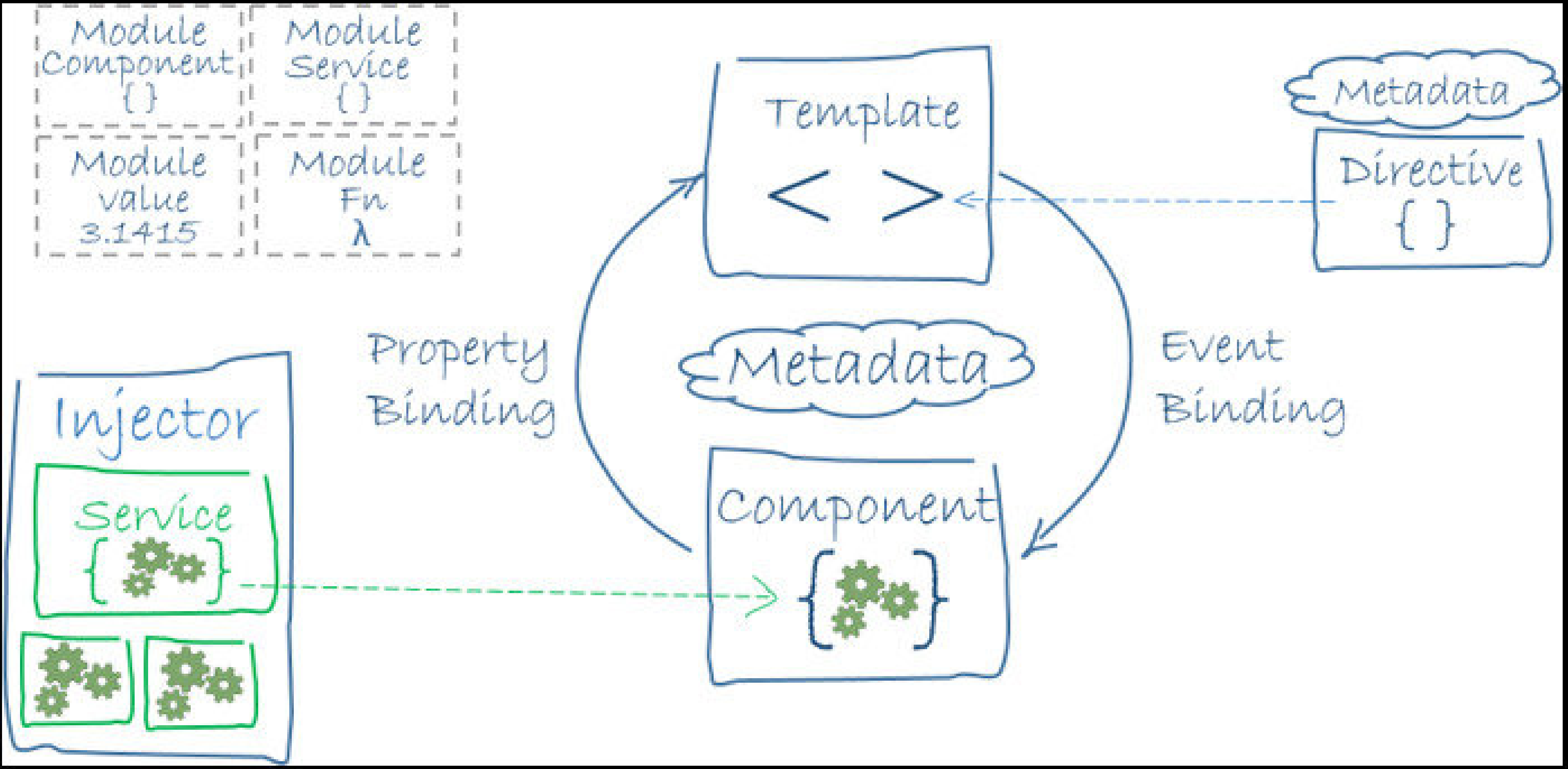
- ngOnChanges()
- ngOnInit()
- ngOnDestroy()
- *ngDoCheck()*
- *ngAfterContentInit()*
- *ngAfterContentChecked()*
- *ngAfterViewInit()*
- *ngAfterViewChecked()*

services

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

import { Employee } from './typings';

@Injectable({
  providedIn: 'root'
})
export class EmployeeModel {

  getEmployee(id: Employee["id"]){
    return this.http.get<Employee>(`http://api.com/employees/${id}`)
  }

  constructor(
    private http: HttpClient
  ) {}
}
```

(COMPONENT) TEMPLATES

data bindings



template syntax

template syntax

```
{{ expr }}
```

- interpolation

template syntax

`{{ expr }}`

- interpolation

`(click)="myFn()"`

- event binding (up)

template syntax

`{{ expr }}`

- interpolation

`(click)="myFn()"`

- event binding (up)

`#name`

- DOM node as variable
- component as variable

template syntax

`{{ expr }}`

- interpolation

`(click)="myFn()"`

- event binding (up)

`#name`

- DOM node as variable
- component as variable

`[attr]="value"`

- property binding (down)

template syntax

`{{ expr }}`

- interpolation

`#name`

- DOM node as variable
- component as variable

`*ngSomething`

- structural directives

`(click)="myFn()"`

- event binding (up)

`[attr]="value"`

- property binding (down)

template syntax

`{{ expr }}`

- interpolation

`#name`

- DOM node as variable
- component as variable

`*ngSomething`

- structural directives

`(click)="myFn()"`

- event binding (up)

`[attr]="value"`

- property binding (down)

`[(ngModel)]="name"`

- two-way property binding (both)

template syntax

`{{ expr }}`

- interpolation

`#name`

- DOM node as variable
- component as variable

`*ngSomething`

- structural directives

`expr | myFormatter`

- pipe (mapper)

`(click)="myFn()"`

- event binding (up)

`[attr]="value"`

- property binding (down)

`[(ngModel)]="name"`

- two-way property binding (both)

pipes

- pure, impure
- stateful, stateless
- built-in pipes:
  - DatePipe
  - UpperCasePipe
  - LowerCasePipe
  - CurrencyPipe
  - PercentPipe
  - async

the async pipe

- **most important one (!)**
- impure
- stateful
- manages subscriptions

```
@Component({
  template: `<div>{{ promise | async }}</div></div>
    <div>{{ stream$ | async }}</div></div>`
})
```

directives

Attribute directives

- NgStyle
- NgClass
- NgModel

Structural directives

- *ngIf / NgIf
- *ngFor / NgForOf
- *ngSwitchCase / NgSwitch

MODULARITY

Dependency Injection

NgModules

```
// empty root module
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
  ],
  exports: [],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

NgModules

Module types

- Root Module - this one is bootstrapped
- Feature Modules - either pre-loaded or lazy loaded
- Routing Modules - separate file with routing only
- Shared Modules

Commonly used built-in modules:

- BrowserModule - bootstrapping + Common
- CommonModule - ngIf
- FormsModule - NgModel, ...
- ReactiveFormsModule - reactive forms
- RouterModule - routing
- HttpClientModule - http