ANGULAR 10

Christian Ulbrich (Zalari)

ANGULAR 2

The king is dead, long live the king!

Christian Ulbrich (Zalari)

SHAMELESS SELF-PLUG

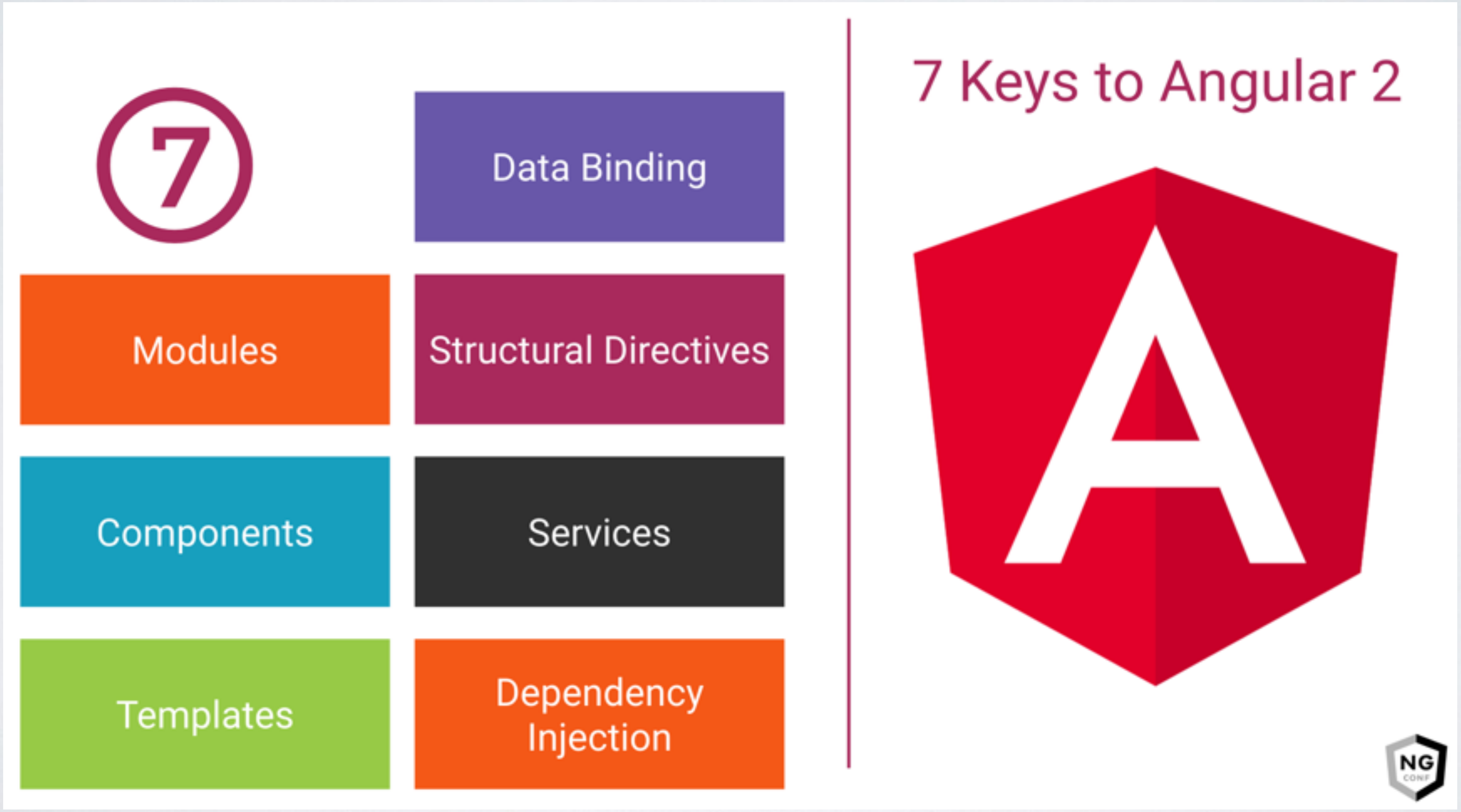
- Zalari UG
- we do Consulting, Developing, Architecturing
- AngularJS is our bread and butter
- we can be hired

AGENDA

- Angular 2, the Strong
- AngularJS, the Oppressed
- Time for revolution?
- Angular 2 core concepts
- Demo Time
- Migration?
- Résumé

ANGULAR 2 THE STRONG

ANGULAR 2 THE STRONG



ANGULAR 2 - MARKETING

- *Angular 2 is a Platform!*
- *Angular is simpler!*
- *Angular 2 is faster!*
- *Angular Material for Angular 2!*
- *Angular Universal!* (server-side rendering, SEO heaven)

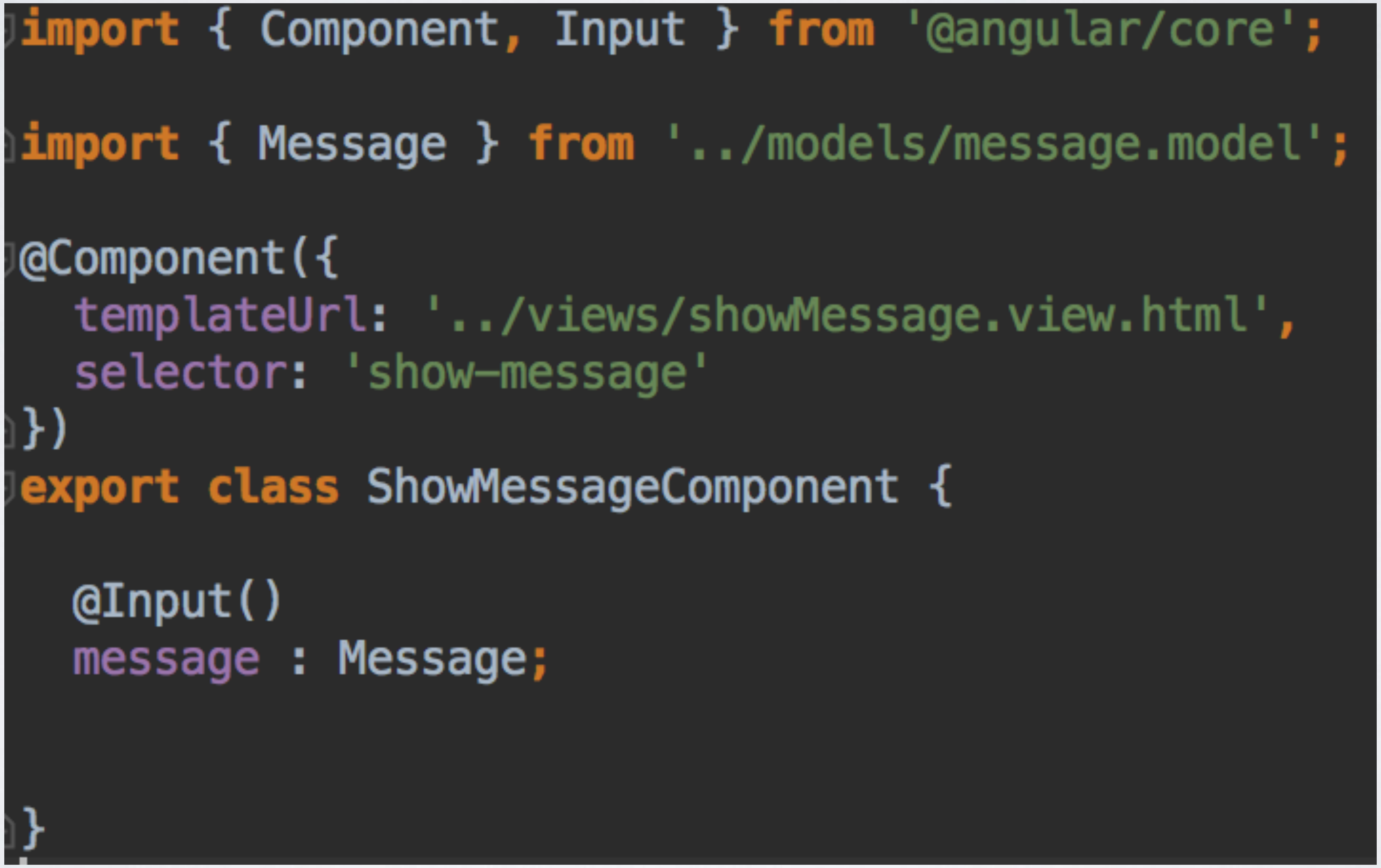
ANGULAR 2
BIASED KEY POINTS

- Angular allows for a better software architecture
- Angular 2 is faster!
- Angular Material for Angular 2!
- Angular 2 is the future…
- but… is is something totally *new*

ANGULAR 2
PNEJSC

- everything is a *decorated* class now
- directives + controllers -> components
- services -> injectables
- components, injectables are **P**lain **N**ew **E**cma**S**cript 2015 **J**ava**S**cript **C**lasses!

ANGULAR 2 „DIRECTIVE“



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

import { Message } from '../models/message.model';

@Component({
  templateUrl: '../views/showMessage.view.html',
  selector: 'show-message'
})
export class ShowMessageComponent {

  @Input()
  message : Message;


}
```

ANGULARJS THE OPPRESSED

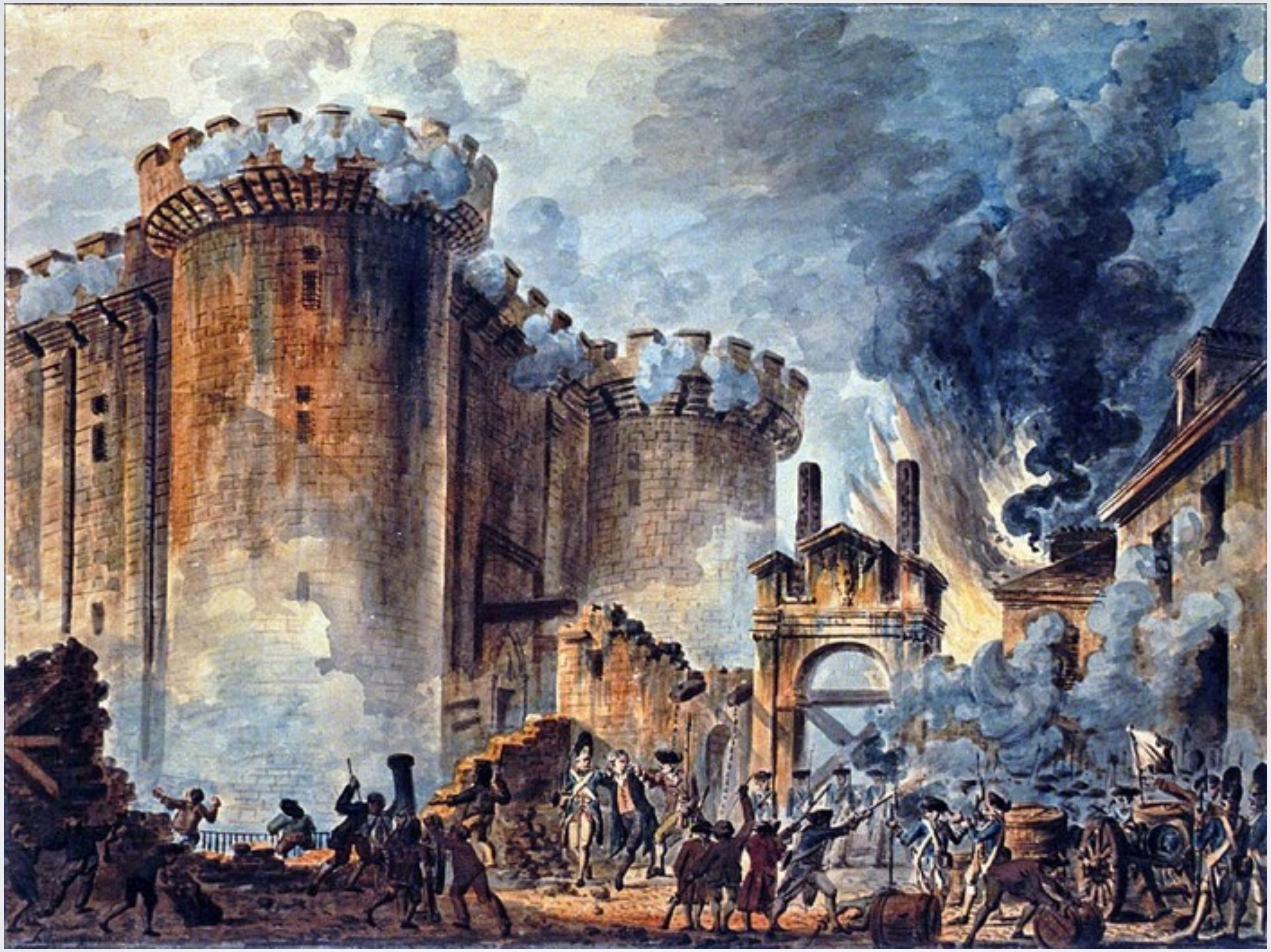
ANGULARJS THE OPPRESSED

- -> Why we hate AngularJS…
- verrrry specific syntax for directives, controllers, services (with factory, service, …)
  - hinders reusability big time
- too much batteries (aka magic) included!
- too old, way to mature!

ANGULARJS THE OPPRESSED

- not perfomance problems, but performance limits
    - 2.500 watchers is the (soft) limit
- working around the watcher limit means working around the framework itself

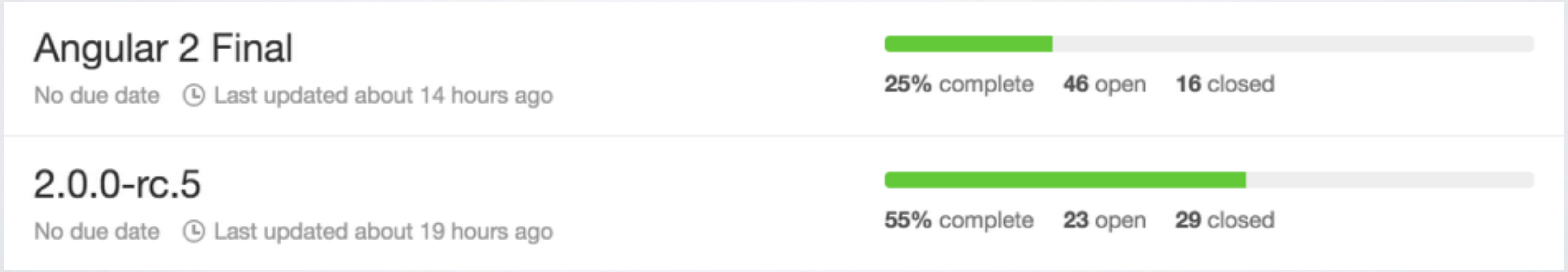
TIME FOR REVOLUTION?



TIME FOR REVOLUTION?

TIME FOR REVOLUTION?

- *AngularJS to be or not to be?*
- right now, you should think twice: Angular 2 is still a *release candidate*



Angular 2 Final
No due date · Last updated about 14 hours ago
25% complete **46** open **16** closed

2.0.0-rc.5
No due date · Last updated about 19 hours ago
55% complete **23** open **29** closed

TIME FOR REVOLUTION?

- The *new* new component router is in *beta*
    - *„we are announcing version 3.0.0-alpha.3 of @angular/router and are deprecating version 2“* on **June, 9th**
- ui-router is in *alpha* for Angular 2
- Angular 2 is complex and you need developers for - and you do not have so many answers yet

TIME FOR REVOLUTION?

- Angular 2 forces you to use TypeScript* or ES6
- you have to learn TypeScript / ES6
- you need proper tooling for TypeScript / ES6
- you need experienced developers for that…

ANGULAR 2
CORE CONCEPTS

- Components, Injectables
- Dependency Injection
- Component Router

ANGULAR 2 COMPONENTS

- directives are *components* now
- there is no implicit *$scope* anymore
- components form component hierarchies
- those hierarchies determine the flow of data between them
- defined „*ports*“ are decorated with @Input or @Output

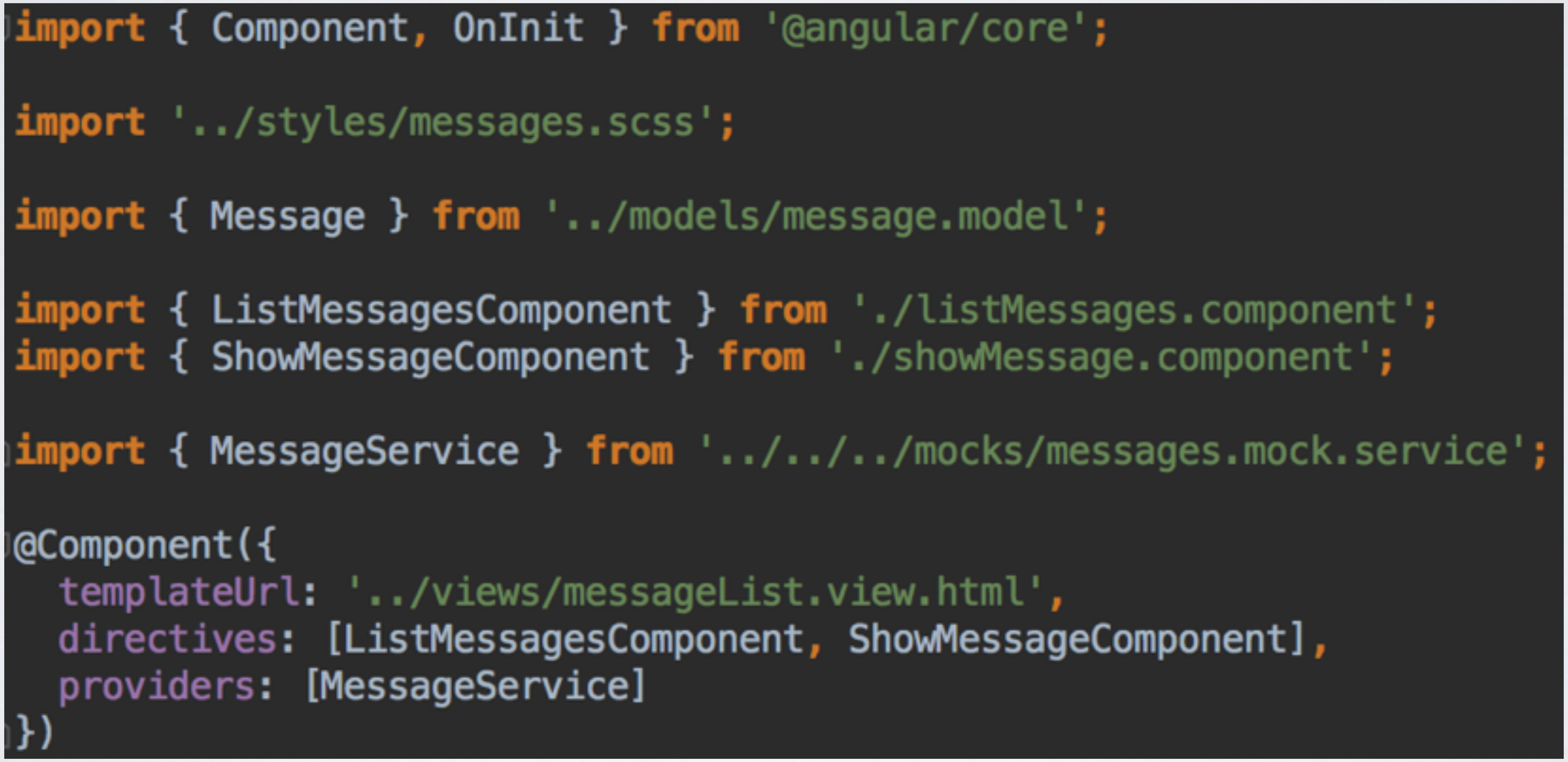
ANGULAR 2 COMPONENTS

- components have an associated *template* and optional *style*
- changes in component’s fields are propagated to template

ANGULAR 2 DEPENDENCY INJECTION

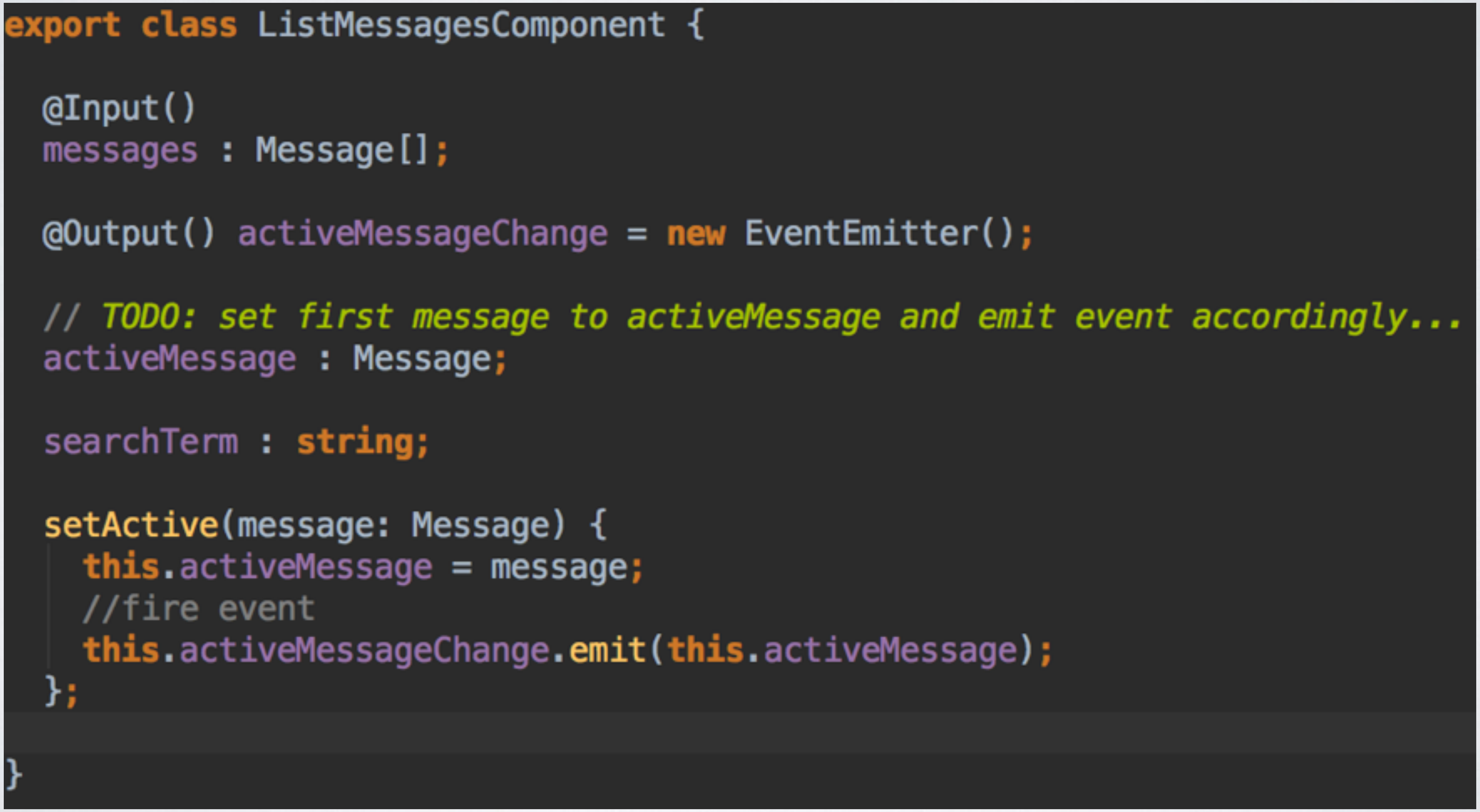
- TypeScript / ES6 imports are used
- all needed „things“ need to be imported and they need to be explicitly set for the component

ANGULAR 2 DEPENDENCY INJECTION



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

import '../styles/messages.scss';

import { Message } from '../models/message.model';

import { ListMessagesComponent } from './listMessages.component';
import { ShowMessageComponent } from './showMessage.component';

import { MessageService } from '../../../mocks/messages.mock.service';

@Component({
  templateUrl: '../views/messageList.view.html',
  directives: [ListMessagesComponent, ShowMessageComponent],
  providers: [MessageService]
})
```

ANGULAR 2 COMPONENTS

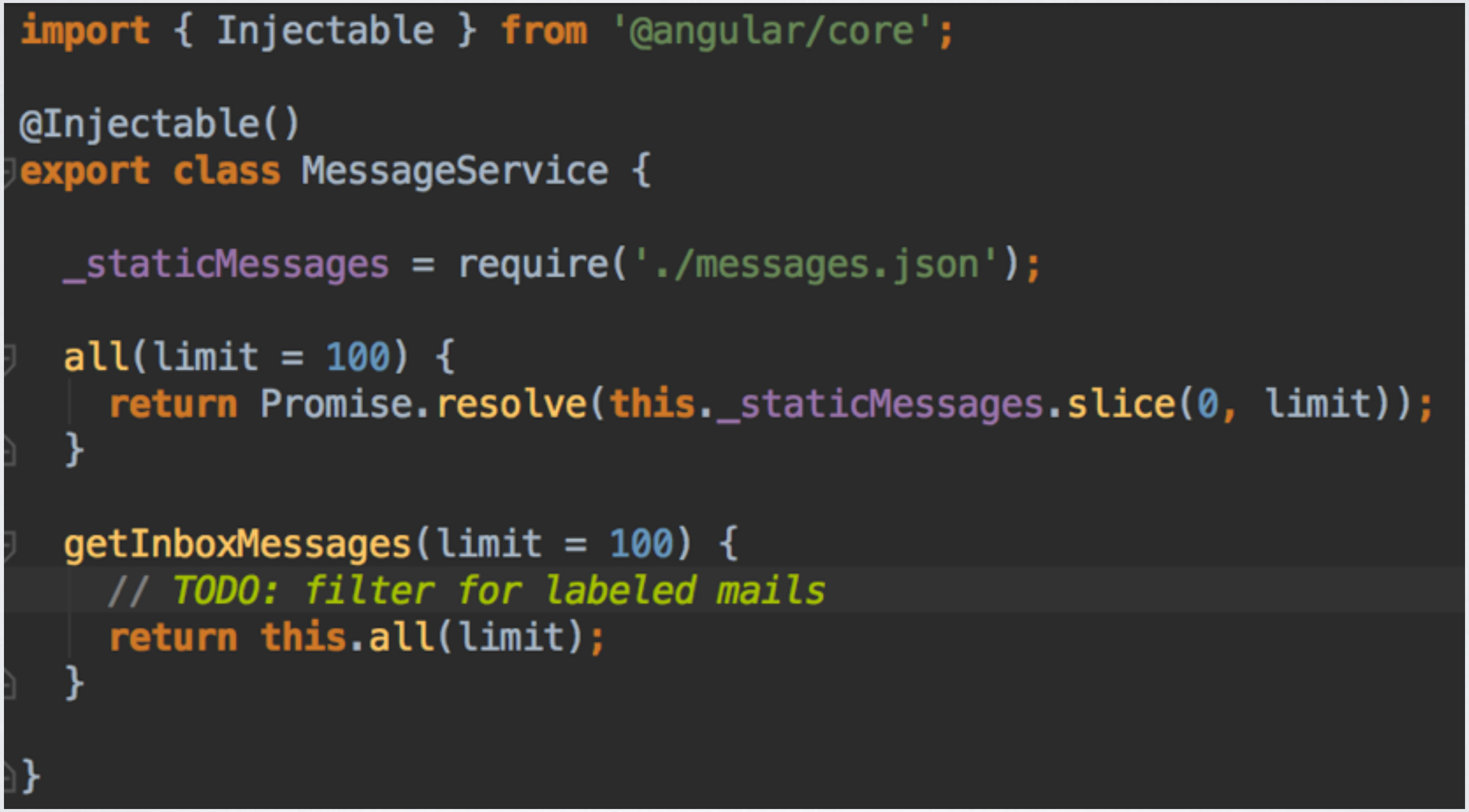


```
export class ListMessagesComponent {

  @Input()
  messages : Message[];

  @Output() activeMessageChange = new EventEmitter();

  // TODO: set first message to activeMessage and emit event accordingly...
  activeMessage : Message;

  searchTerm : string;

  setActive(message: Message) {
    this.activeMessage = message;
    //fire event
    this.activeMessageChange.emit(this.activeMessage);
  };

}
```

ANGULAR 2 INJECTABLES

- „Services“ are now *Injectables*
- Dependency Injection is mostly done using ES 6 / TypeScript syntax

ANGULAR 2 INJECTABLES

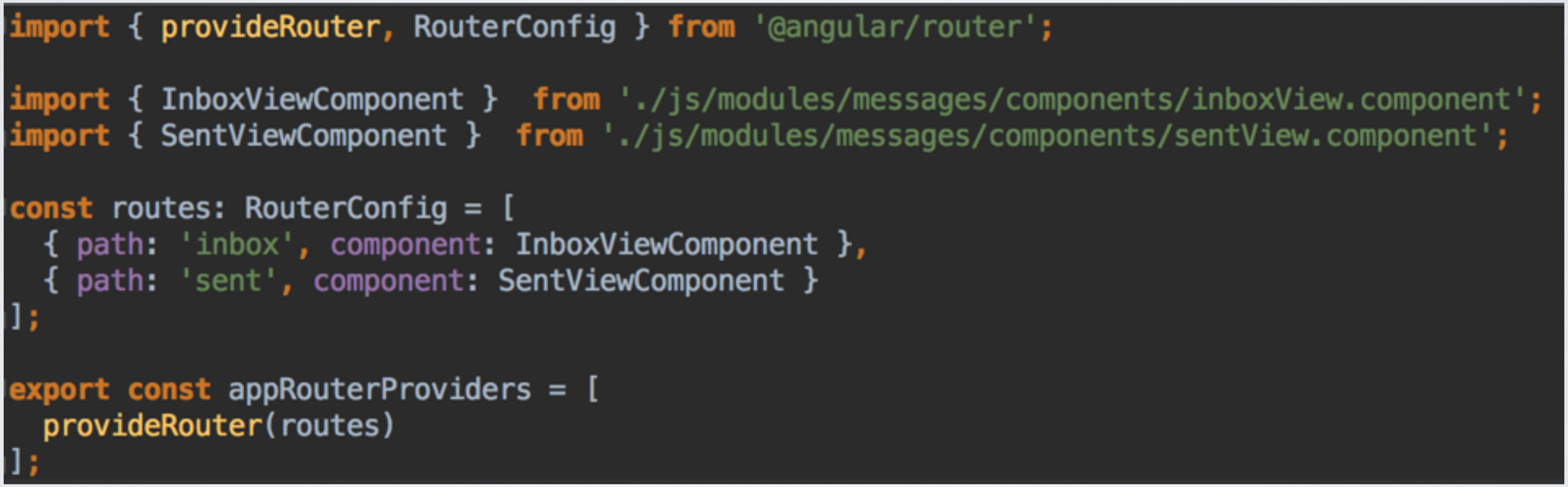


```
import { Injectable } from '@angular/core';

@Injectable()
export class MessageService {

  _staticMessages = require('./messages.json');

  all(limit = 100) {
    return Promise.resolve(this._staticMessages.slice(0, limit));
  }

  getInboxMessages(limit = 100) {
    // TODO: filter for labeled mails
    return this.all(limit);
  }

}
```

ANGULAR 2 COMPONENT ROUTER

- URLs are now matched to a component
- Param passing still possible
- Component Router abstracts resolve functionality as *Guards*

ANGULAR 2 COMPONENT ROUTER



```
import { provideRouter, RouterConfig } from '@angular/router';

import { InboxViewComponent }  from './js/modules/messages/components/inboxView.component';
import { SentViewComponent }  from './js/modules/messages/components/sentView.component';

const routes: RouterConfig = [
  { path: 'inbox', component: InboxViewComponent },
  { path: 'sent', component: SentViewComponent }
];

export const appRouterProviders = [
  provideRouter(routes)
];
```

ANGULAR 2 DEMO

- a small SPA use case - mailbox client
  - implemented with AngularJS:

    https://github.com/zalari/mailbox-demo-angular2

  - implemented with Angular 2:

    https://github.com/zalari/mailbox-demo-angularjs

ANGULAR 2 DEMO

DEMO TIME

(aka let’s hope, we do not have to do live coding)

ANGULAR 2 DEMO $SCOPE VS. EXPLICIT FLOW

- layer of abstraction in AngularJS is hiiiigher
- -> in Angular 2 we need to explicitly model the data flows of our Components, using @Input and @Output and *EventEmitter*
- -> event-driven architecture

ANGULAR 2 MIGRATION OPTIONS

- there is no real migration path available
- general idea is to have a hybrid

ANGULAR 2 MIGRATION OPTIONS

ANGULAR 2 MIGRATION OPTIONS

- combine both AngularJS and Angular 2 in one app
- migrate feature by feature, service by service
- use *component directives*, that can be used from Angular 2

ANGULARJS JUDGEMENT DAY

ANGULARJS JUDGEMENT DAY

The good news is, it has not come yet!

ANGULAR 2
PREPARE FOR JUDGEMENT DAY!

- structure your applications in a better way:
  - use a bundler and ES6 / TypeScript
  - use POJSO / PNEJSC for controllers a like…
  - allows for easier migration for *the next big thing*

ANGULAR 2 RESUME

- it is bound to happen anyway
- but it will take time for the framework to mature
- invest in Angular 2 if you can live with the consequences:
  - lower productivity
  - lack of best practices

ANGULAR 2 RESUME

- as time goes by, we will have more Angular 2 talks with DresdenJS
- ~~Victims~~, ~~Volunteers~~, Pioneers are welcome!

ANGULAR 2 READER'S DIGEST

- Upgrading from AngularJS http://bit.ly/1YixNzE
- *Official* Angular 2 Style Guide http://bit.ly/1SZxd9T
- Build Angular 2 with ES5 http://bit.ly/2acruNP
- Angular 2 Router revisited http://bit.ly/29APBnJ
- Angular (2) Router http://bit.ly/29APvwp