IK WIL

Angular Fundamentals Module 1 - Introduction

vijfhart IT-OPLEIDINGEN

Agenda

- Introduction
- Angular Platform
- Hello World in Angular – insights in boilerplate-code
- Angular 2 in depth:
  - Components
  - ECMAScript 2015 + TypeScript
  - Data binding
  - Services
  - Http and Observables (RxJS)

About me…

- Instructor: Stijn Janssen
- Microsoft MCT
- Freelance IT trainer and course writer
- Current courses: JavaScript, Angular, Power Platform, Azure, Java, Linux
- Previous and current projects: Order systems, BlockChain, Games + interest in Genetic Algorithms
- Interests: Car tuning, Boardgames, Music, Woodworking

About you...

Knowledge of webdevelopment, (mobile/web)?

Knowledge of AngularJS 1.x?

Knowledge of other (web)languages?

Expectations for this session?

Concrete projects?

Materials

Software: Downloads

Handout: This presentation, after this session

Exercises: @Github

Demo file: @Github

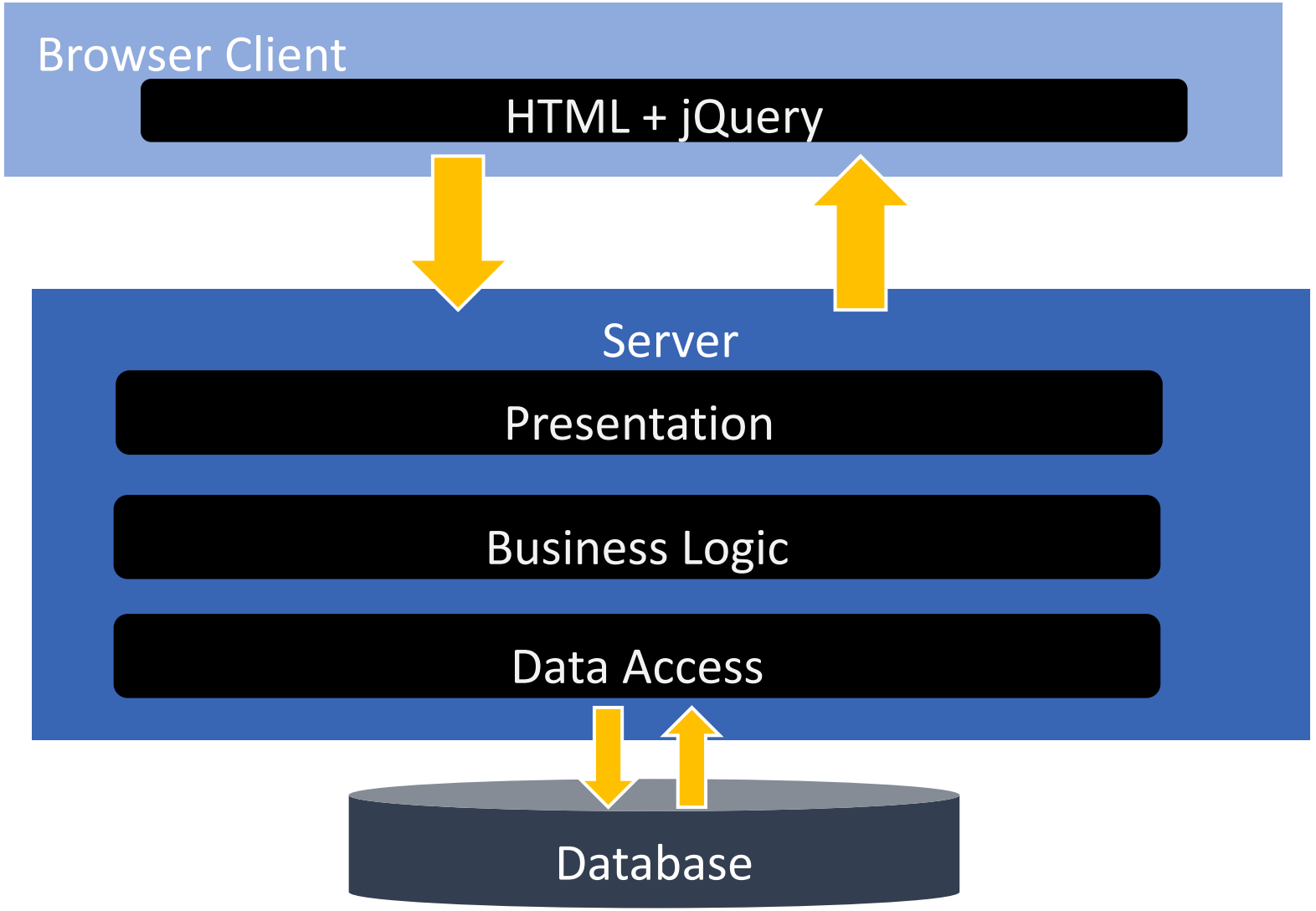
angular.io/

Angular Platform

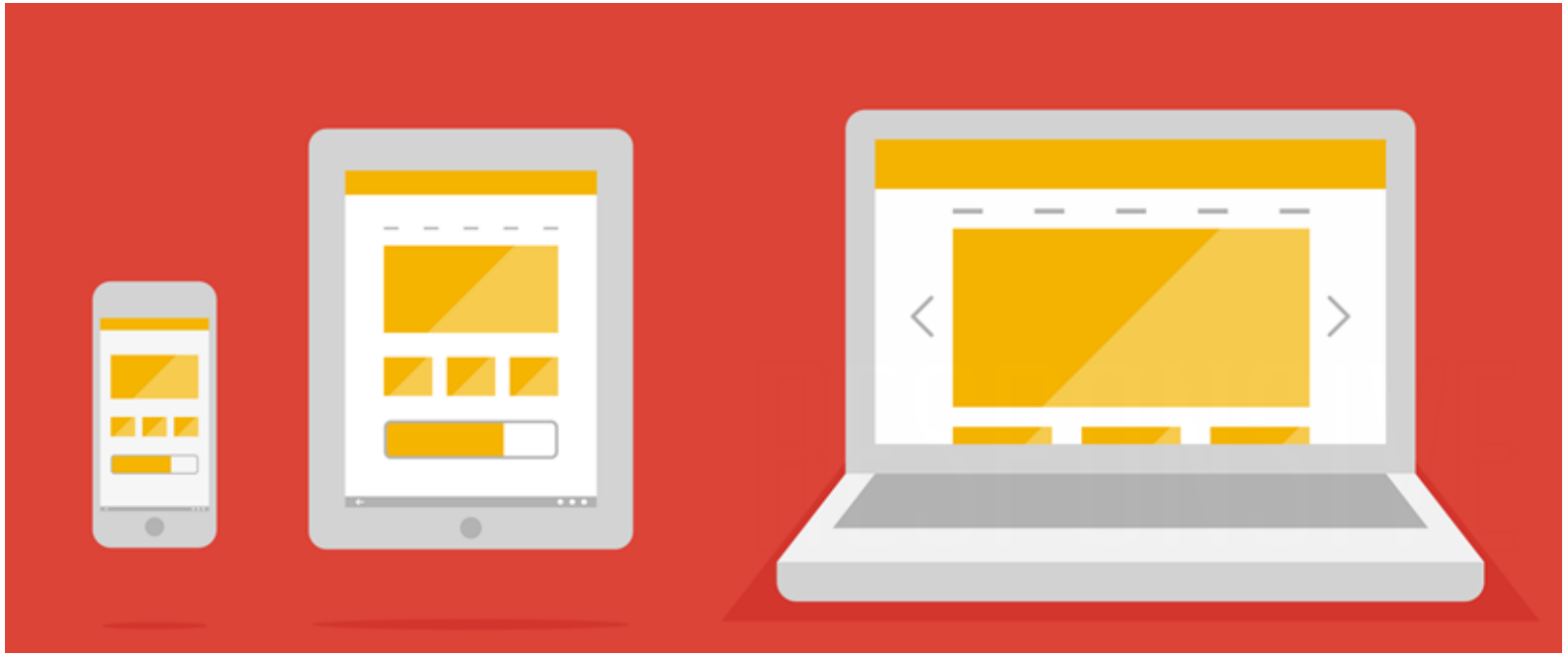
What and why

Conventional Web App

2000 - 2013

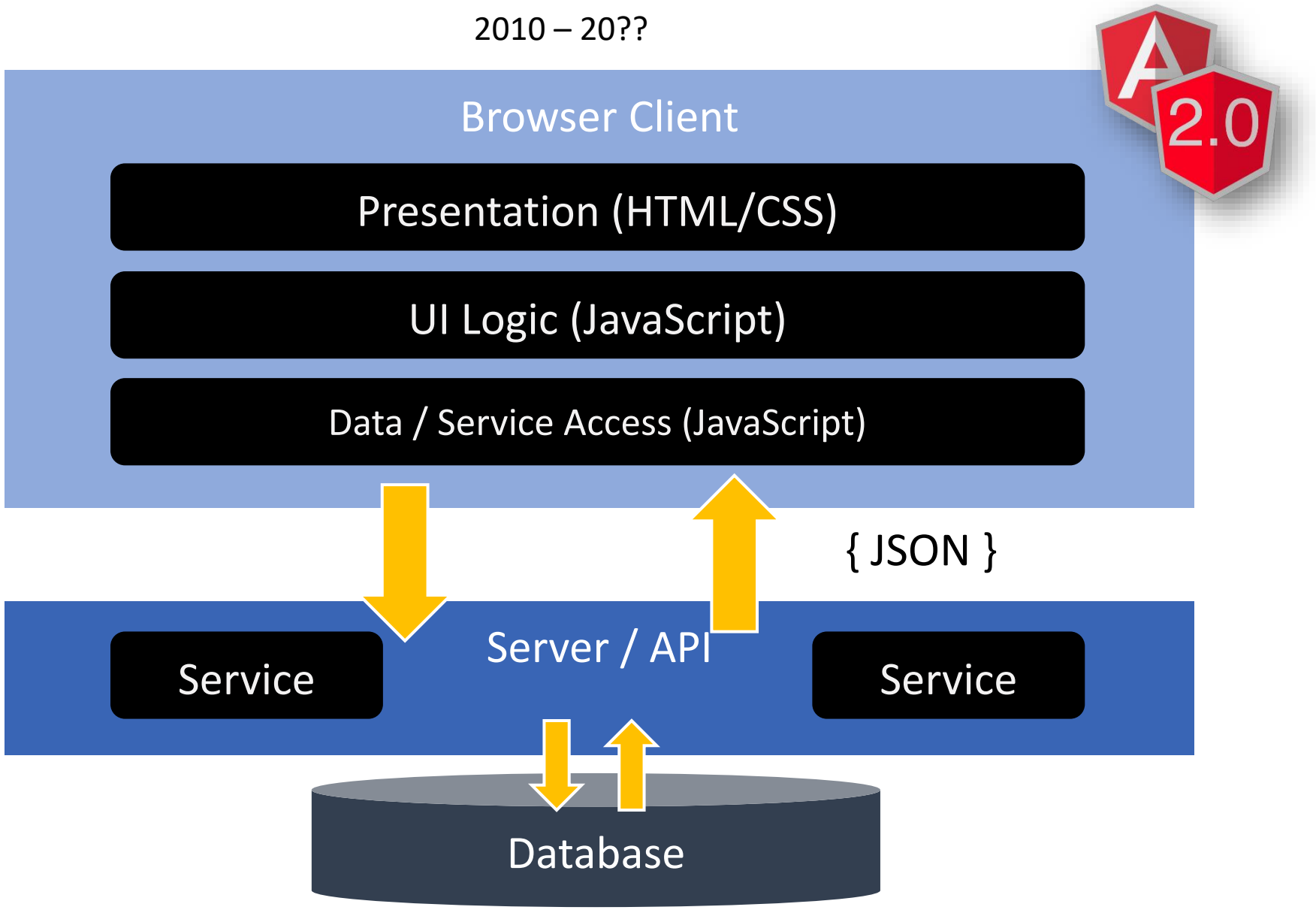


But from around 2010:



Single Page Application

2010 – 20??



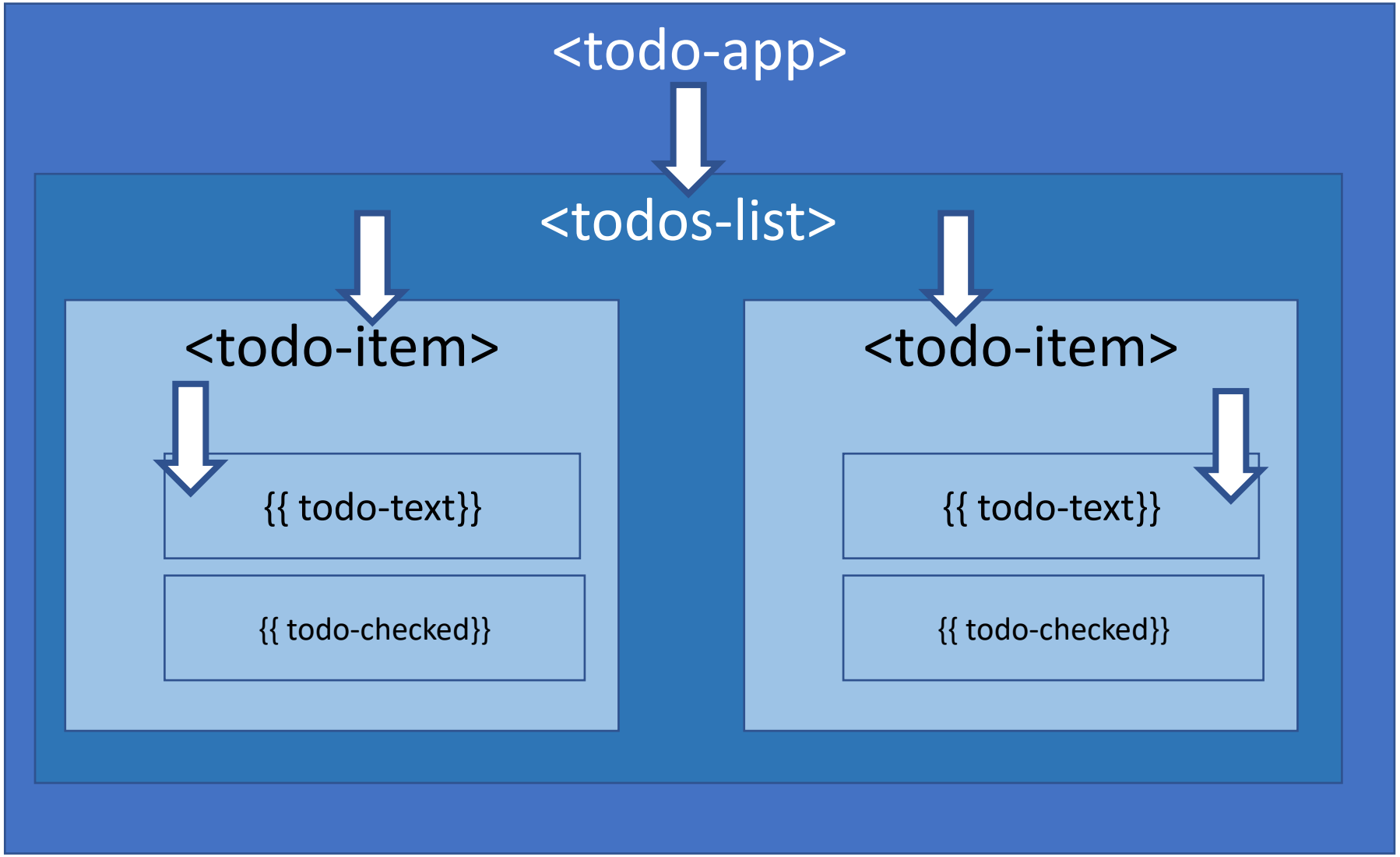
“It's just *Angular*”

But what is a single page application?

Framework to Platform

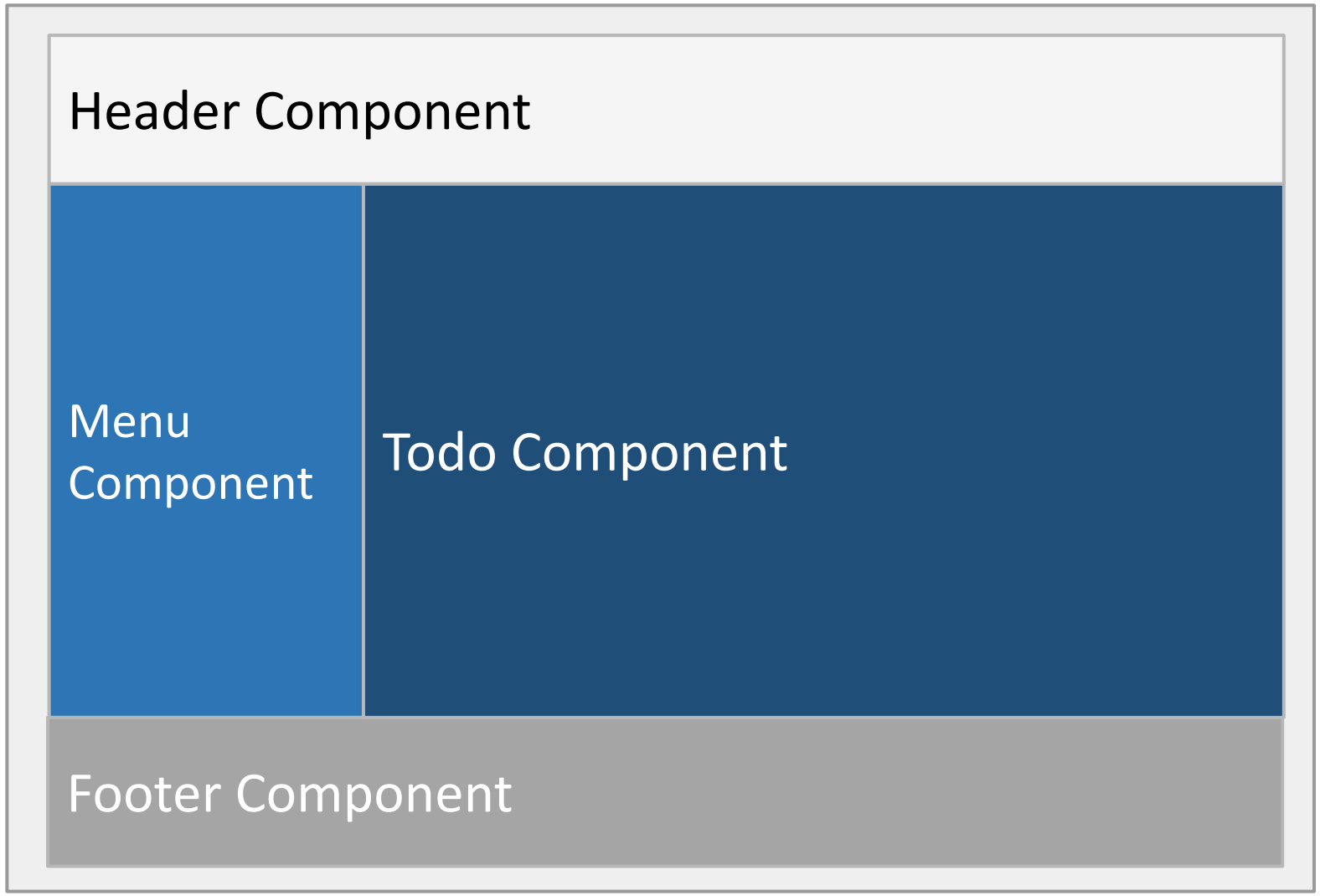
	Scaffolding	Code completion & Refactoring	Debugging
Tooling	Angular CLI	Language Services	Augury
Libraries	Material 2	Mobile	Universal
	(AOT)Compile	Change Detection	Renderer
Core	Components & Dependency Injection	Decorators	Zones

Angular 2 - components



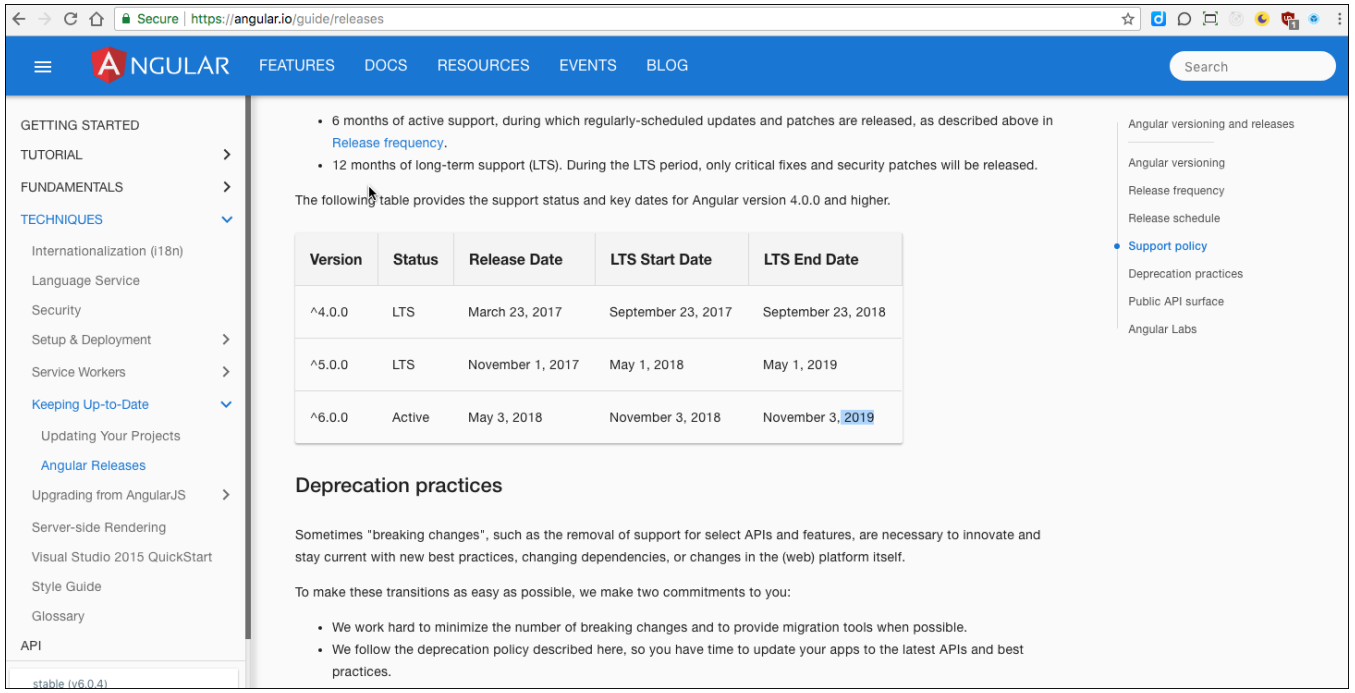
“An Angular-app is a tree of components”

Components – visually



Angular Versies and -Long Time Support

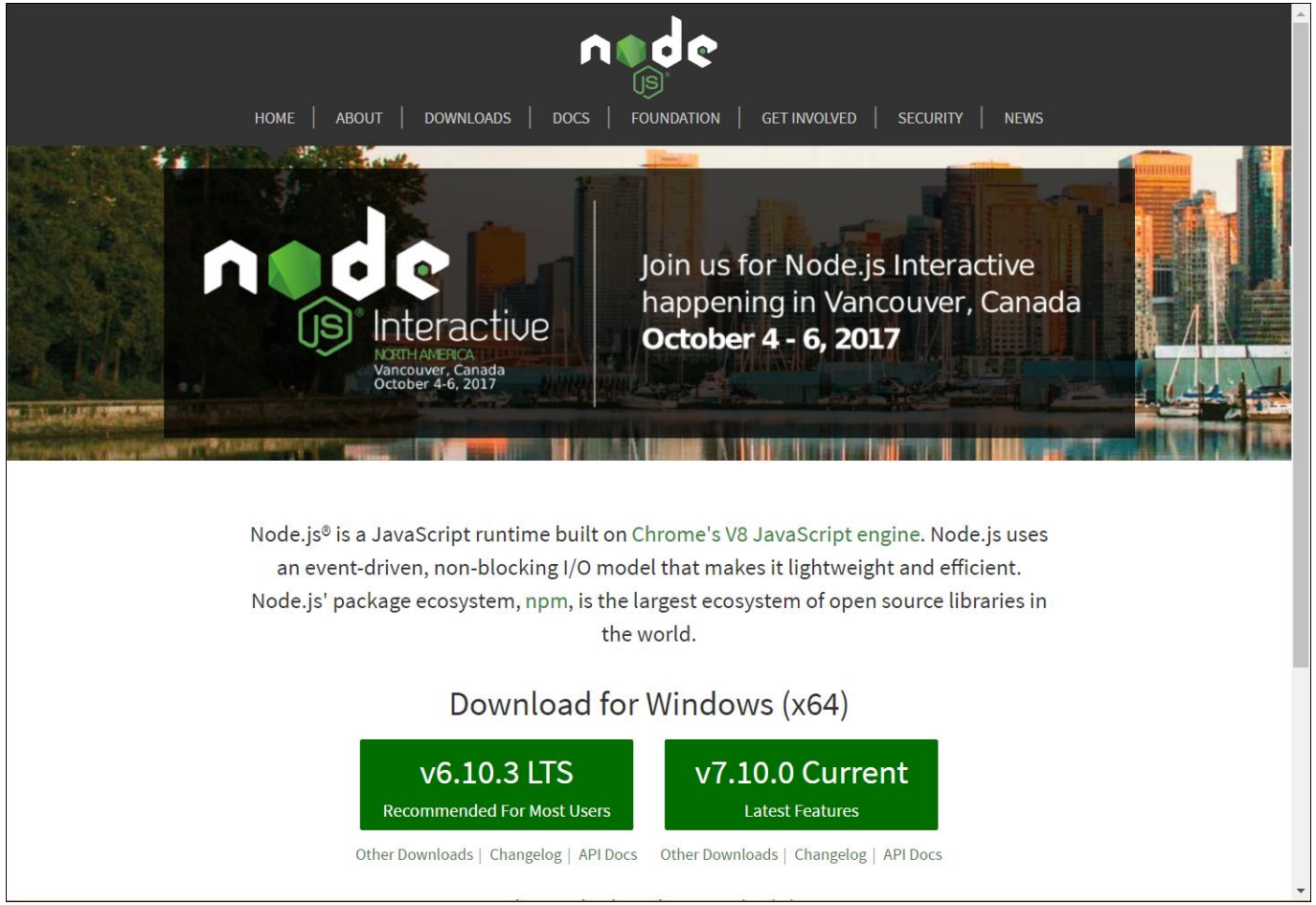
- https://angular.io/guide/releases
- Upgrade/update guide here



Let’s write some code

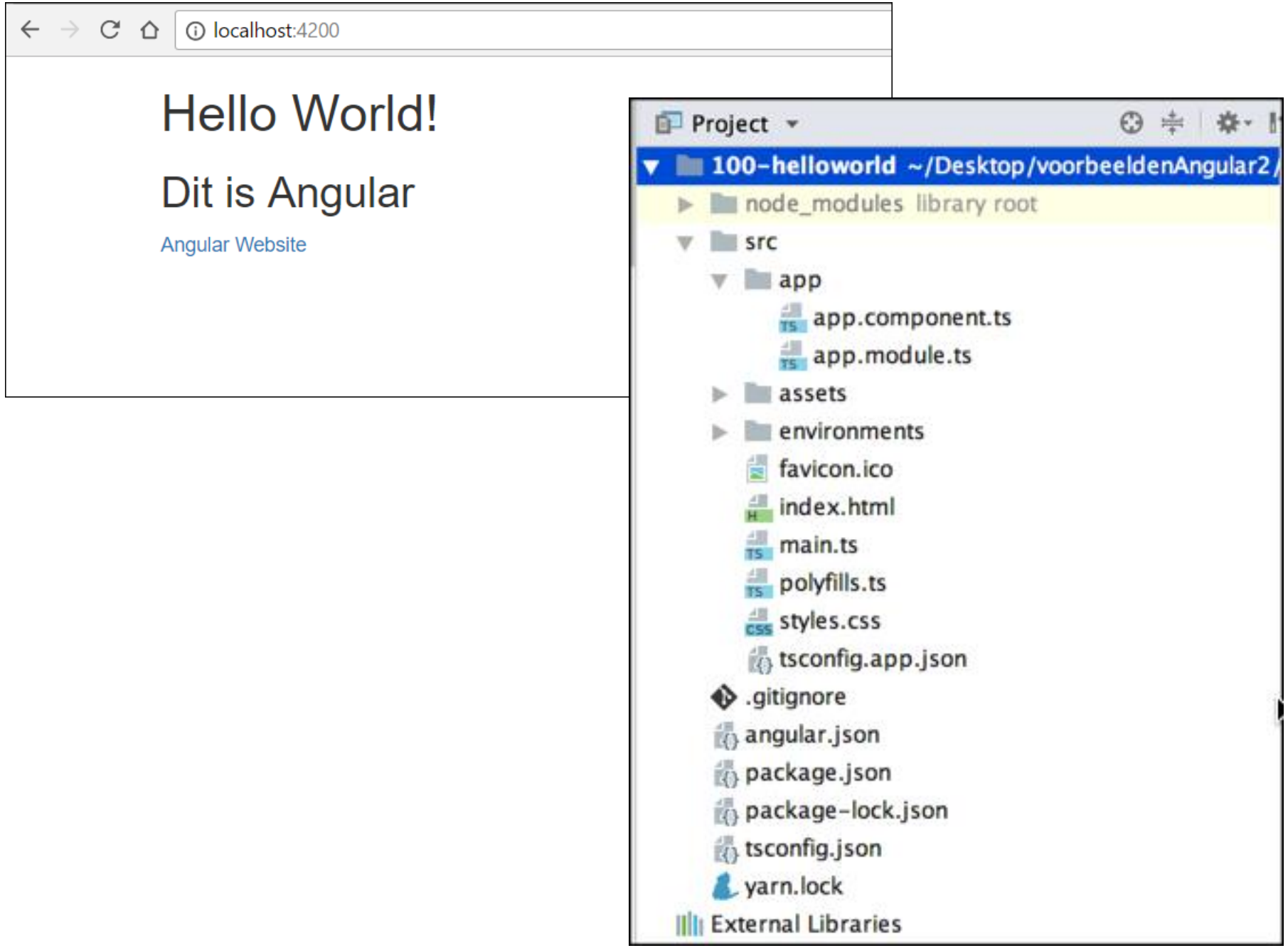
Hello World in Angular 2

Angular development dependency: NodeJS 8.x +



Exercise

- https://github.com/TeacherStijn/AngularVoorbeelden/blob/master/Exercises.pdf
- Exercise 1a
- Go to browser: http://localhost:4200


localhost:4200
Hello World!
Dit is Angular
Angular Website
Project
100-helloworld ~/Desktop/voorbeeldenAngular2/
node_modules library root
src
app
app.component.ts
app.module.ts
assets
environments
favicon.ico
index.html
main.ts
polyfills.ts
styles.css
tsconfig.app.json
.gitignore
angular.json
package.json
package-lock.json
tsconfig.json
yarn.lock
External Libraries

Boilerplate files #1 - `package.json`

```
{
  "name": "hello-angular",
  "description": "Voorbeeldproject bij de training Angular (C) - info@kassenaar.com",
  "version": "0.0.1",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "6.0.0",
    "@angular/common": "6.0.0",
    "@angular/compiler": "6.0.0",
    "@angular/core": "6.0.0",
    "@angular/forms": "6.0.0",
    "rxjs": "^6.1.0",
    "zone.js": "^0.8.26"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~0.6.0",
    "@angular/cli": "6.0.0",
    "typescript": "2.7.2"
  },
  "author": "Peter Kassenaar <info@kassenaar.com>"
}
```

Boilerplate files #2 - `tsconfig.json`

```
{
  "compileOnSave"  : false,
  "compilerOptions": {
   "outDir"                : "./dist/out-tsc",
   "baseUrl"               : "src",
   "sourceMap"             : true,
   "declaration"           : false,
   "moduleResolution"      : "node",
   "emitDecoratorMetadata" : true,
   "experimentalDecorators": true,
   "target"                : "es5",
   "typeRoots"             : [
     "node_modules/@types"
   ],
   "lib"                   : [
     "es2016",
     "dom"
   ]
  }
}
```

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "helloworld": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist",
            "index": "src/index.html",
            "main": "src/main.ts",
            "tsConfig": "src/tsconfig.app.json",
      …
}
```

(Angular < 6.0.0: `.angular-cli.json`)

Step 2 – Component

Convention - components in directory `/src/app`

Or: edit in `angular.json`

Filename: `src/app/app.component.ts`

```
import {Component} from ‘@angular/core';
@Component({
   selector: 'hello-world',
   template: '<h1>Hello Angular 2</h1>'
})
export class AppComponent {


}
```

Step 3 – `@ngModule`

Convention - filename: `/src/app.module.ts`

```
// Angular Modules
import {NgModule}      from '@angular/core';
import {BrowserModule} from '@angular/platform-browser';

// Custom Components
import {AppComponent} from './app.component';

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

Entry entry point of the application

Step 4 - bootstrap component

Best practice: bootstrap app in separate component

Convention: `main.ts`, or `app.main.ts`.

```
import {enableProdMode} from '@angular/core';
import {platformBrowserDynamic} from '@angular/platform-browser-dynamic';

import {AppModule} from './app/app.module';
import {environment} from './environments/environment';

if (environment.production) {
   enableProdMode();
}

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

Step 5 – index.html

`index.html` - simple HTML file - expanded at runtime by WebPack

```html
<html>

<head>
  <meta charset="utf-8">
  <title>Helloworld</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
```

Body of index.html

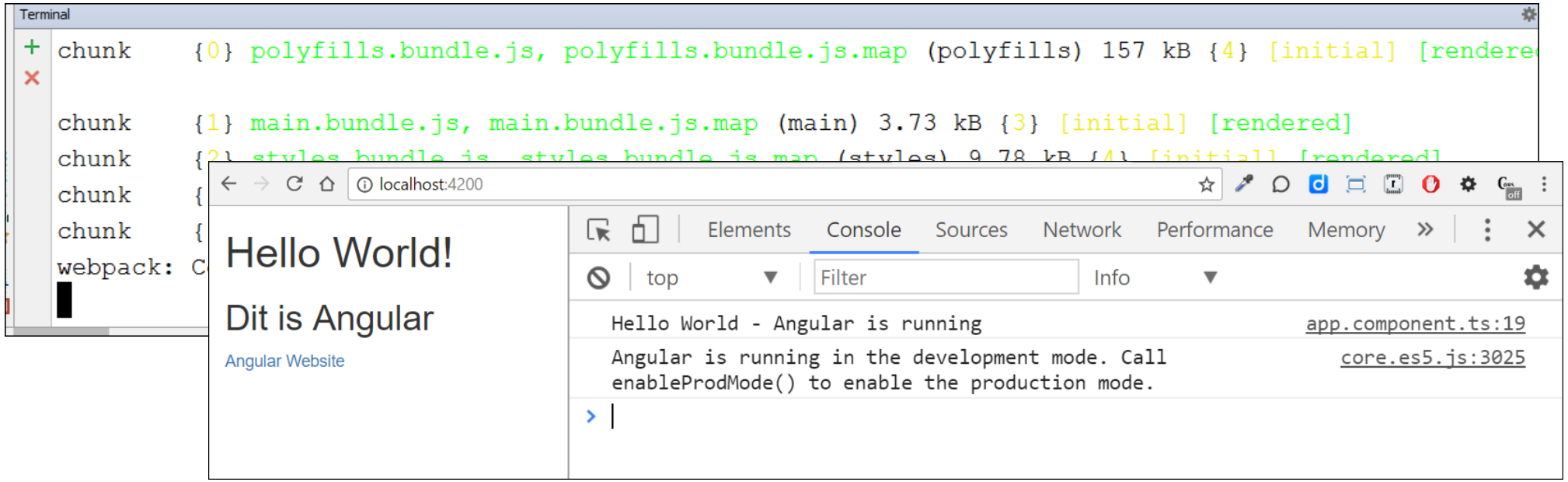
Pointer to the root-component selector:

```
<body>
  <hello-world>
    Loading…
  </hello-world>
</body>
```

Run the app

`npm start` – runs the script command ‘start’ from `package.json`.

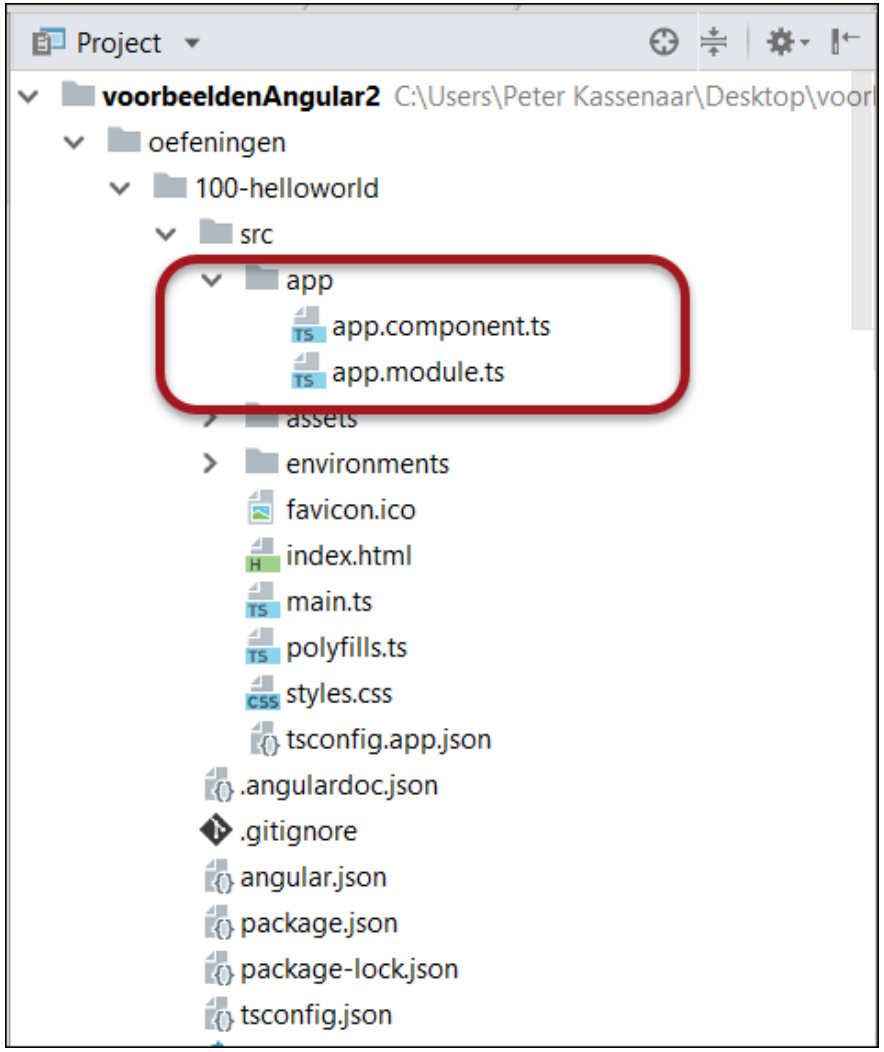
`ng serve` - start a global angular-cli instance

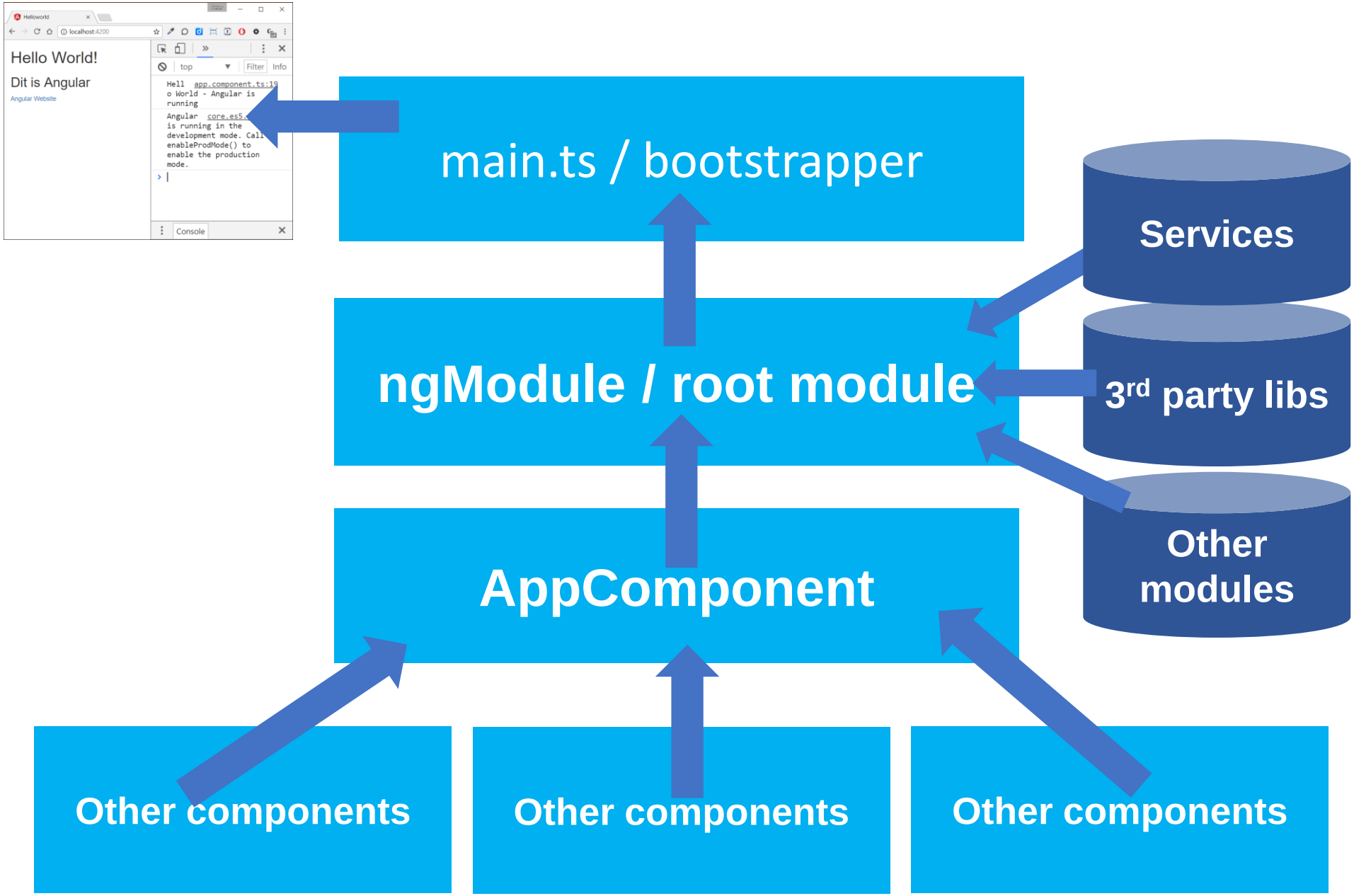


After that: make first changes in: `app.component.ts`

– will be picked up via Live Reload

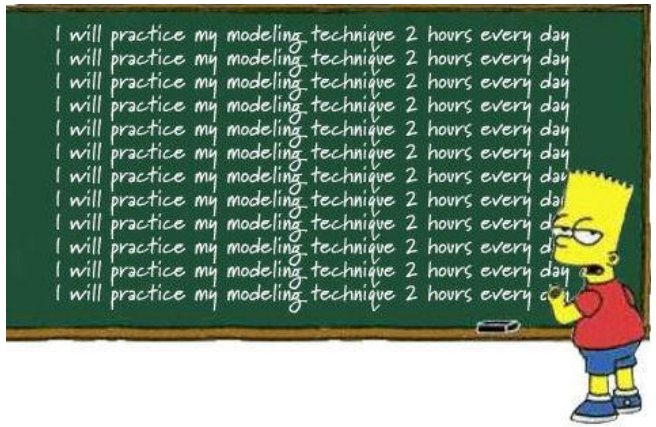
Structure


Project
voorbeeldenAngular2 C:\Users\Peter Kassenaar\Desktop\voor
oefeningen
100-helloworld
src
app
app.component.ts
app.module.ts
assets
environments
favicon.ico
index.html
main.ts
polyfills.ts
styles.css
tsconfig.app.json
.angulardoc.json
.gitignore
angular.json
package.json
package-lock.json
tsconfig.json


Helloworld
localhost:4200
Hello World!
Dit is Angular
Angular Website
top
Filter
Info
Hell app.component.ts:19
o World - Angular is
running
Angular core.es5.
is running in the
development mode. Call
enableProdMode() to
enable the production
mode.
Console
main.ts / bootstrapper
ngModule / root module
AppComponent
Services
3rd party libs
Other
modules
Other components
Other components
Other components

Checkpoint

- There is a lot of boilerplate code for an Angular-app
- Steps:
  1. Set up environment, boilerplate & libraries
  2. Write Angular Root Component for the app
  3. Bootstrap the component
  4. Write HTML-pagina (`index.html`)
- After that: extend the app…



Angular CLI

Quickly configure new projects via the command line

Angular-CLI to the rescue

- It *is* possible to start new Angular-projects from scratch
- This is *way* easier with the Angular CLI
- CLI-options:
  - Scaffolding (global setup)
  - Generating
  - Testing
  - Building
  - AOT-Compiling
  - ...

Scaffolding - Angular CLI

Define projects, components, routes and more from the command line

https://github.com/angular/angular-cli

and

https://angular.io/cli

```
npm install -g @angular/cli
```

Main commands

```
ng new PROJECT_NAME
cd PROJECT_NAME
ng serve
```

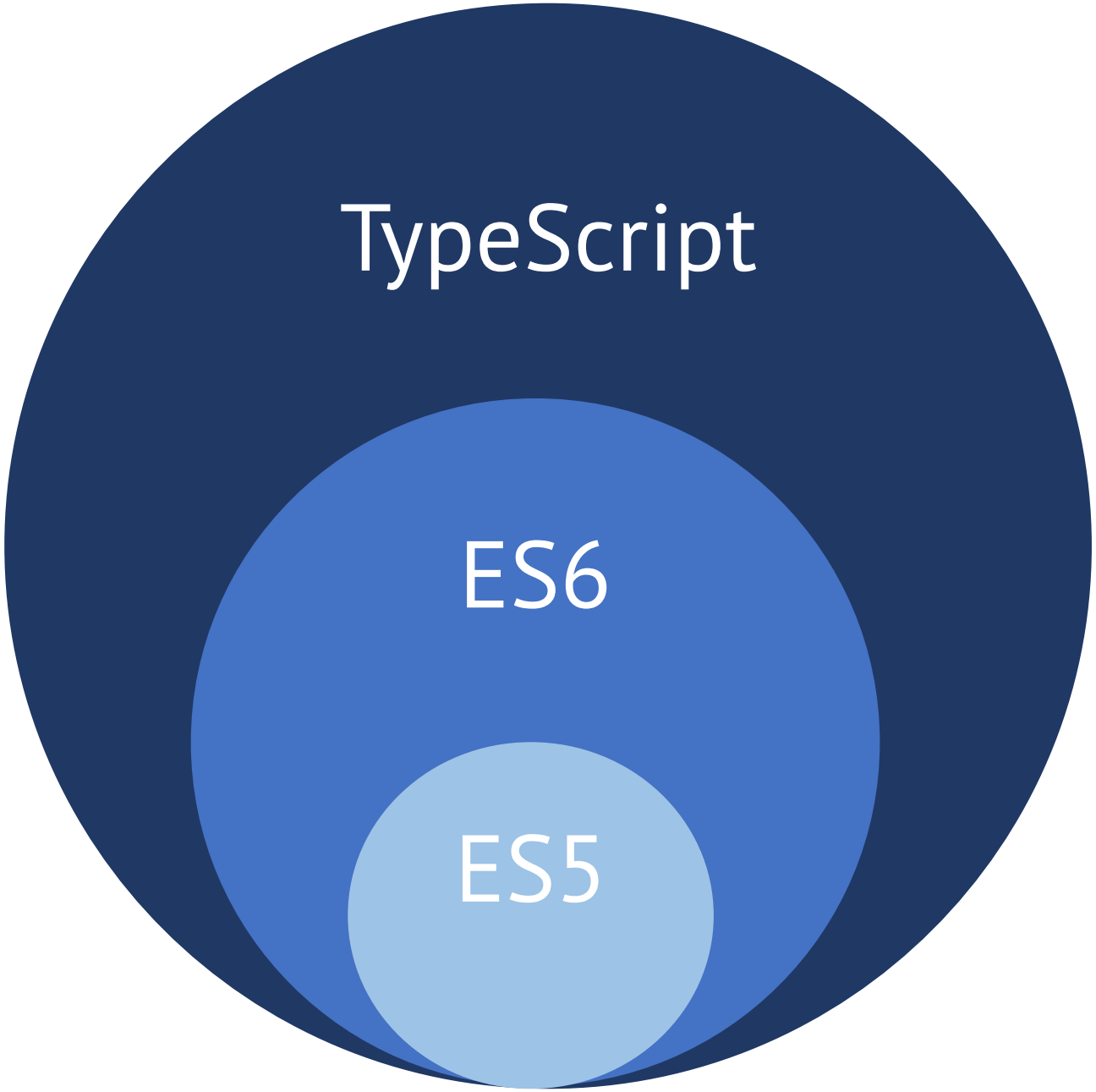
Project is served on `http://localhost:4200`

Exercise

- https://github.com/TeacherStijn/AngularVoorbeelden/blob/master/Exercises.pdf
- Exercise 1b, 1c
- Go to browser: http://localhost:4200

Angular 2 Code Languages

TypeScript and ES6


TypeScript
ES6
ES5

ES6 and TypeScript

The future of JavaScript is ES6/ES2015

- Major update of JavaScript as programming language
- Modules, classes and more
- Helps with developing in Angular 2

TypeScript extends ES6 further with:

- Annotations & types
- Interfaces
- Compiler

Great tool support!

Parts of a Component Class

imports

```
import { Component } from '@angular/core';
import { DataService } from './services/data-
service';
```

Decorators/ annotations

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

class

```
export class
OrdersComponent {
        …
}
```

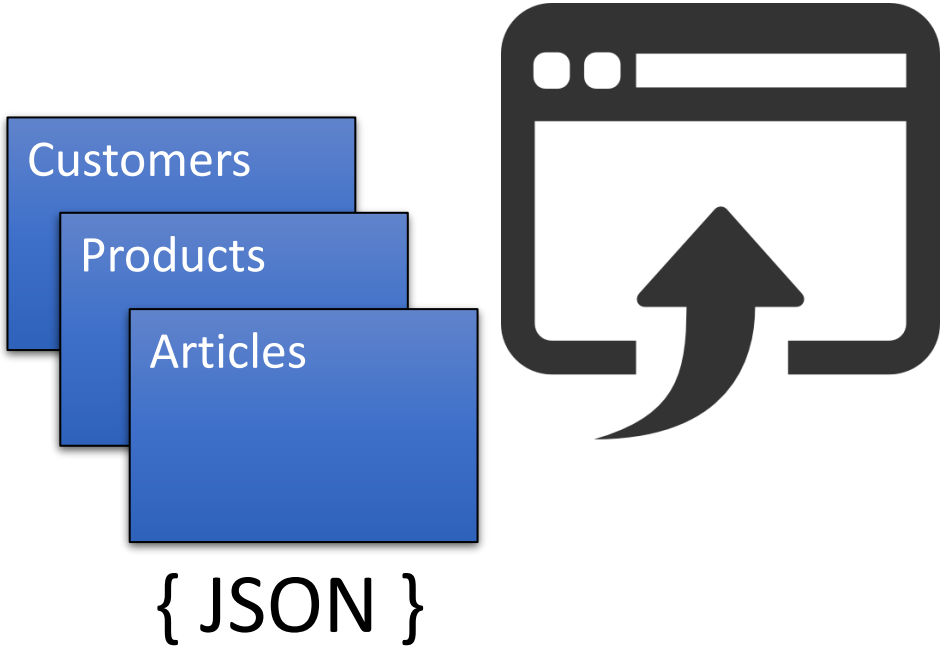
IK WIL

Angular Fundamentals Module 2 - Databinding

vijfhart IT-OPLEIDINGEN

What is databinding

- Showing data in the user interface
- Data coming from:
  - Controller / class
  - Database
  - User input
  - Other systems



Declarative syntax

- New notation in HTML-templates
    1. Simple data binding
    2. Event binding
    3. One-way data binding a.k.a. attribute binding (**other course**)
    4. Two-way data binding (**other course**)

1. Simple data binding syntaxis

Unchanged from Angular 1. So still double braces:

```
<div>City: {{ city }}</div>
```

```
<div>Firstname: {{ person.firstname }}</div>
```

Always: collaboration with component/class

```
import {Component} from '@angular/core';
@Component({
   selector: 'hello-world',
   template: `<h1>Hello Angular 2</h1>
      <h2>My naam is : {{ name }}</h2>
      <h2>My favorite city is : {{ city }}</h2>
   `
})
export class AppComponent {
   name = 'Stijn Janssen';
   city = 'Veenendaal'
}
```

Or: properties via constructor

```
export class AppComponent implements OnInit {
   name: string;
   city: string;

   constructor() {
      this.name = 'Stijn Janssen';
      this.city = 'Veenendaal'
   }
}
```

Binding through a loop: `*ngFor`

Template:

```
<h2>My favorite cities are:</h2>
<ul>
    <li *ngFor="let city of cities">{{ city }}</li>
</ul>
```

Class:

```
// Class met properties, array met cities
export class AppComponent {
   name:string;
   cities:string[];

   constructor() {
      this.name   = 'Dummy name';
      this.cities = ['Groningen', 'Hengelo', 'Den Haag', 'Enschede'];
   }
}
```

Making a model (as in: MVC)

Class with properties which gets exported:

```
export class City{
   constructor(
      public id: number,
      public name: string,
      public province: string
   ){ }
}
```

Note the shorthand notation at: `public id : number:`

1. Makes a local parameter
2. Makes a public property with same name
3. Initializes property with instantiazing of the class with `new`

P. 86 en verder

Model usage

1. Import model-class

```
import {City} from './shared/city.model’;
```

2. Edit component

```
export class AppComponent {
      name: string   = ‘Dummy name';
      cities: City[] = [
         new City(1, 'Groningen', 'Groningen'),
         new City(2, 'Hengelo', 'Overijssel'),
         new City(3, 'Den Haag', 'Zuid-Holland'),
         new City(4, 'Enschede', 'Overijssel'),
      ];
}
```

3. Edit view

```
<li *ngFor=”let city of cities">{{ city.id}} - {{ city.name }}</li>
```

Display conditionally with `*ngIf`

Use the `*ngIf` directive (note the asterisk!)

```
<h2 *ngIf="cities.length > 3">You have many favorites!</h2>
```

External templates

If you don't like inline HTML:

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

File `app.html`

```
<!-- HTML in external template -->
<h1>Hello Angular 2</h1>
<p>This is an external template</p>
<h2>My naam is : {{ name }}</h2>
<h2>My favorite cities are:</h2>
…
```

Exercise

- https://github.com/TeacherStijn/AngularVoorbeelden/blob/master/Exercises.pdf
- Exercise 2a, 2b, 2c, (2d=attribute binding)
- Go to browser: http://localhost:4200

User input and event binding

Responding to mouse, keyboard, hyperlinks and more

Event binding syntaxis

Use parentheses for events:

Angular 1:

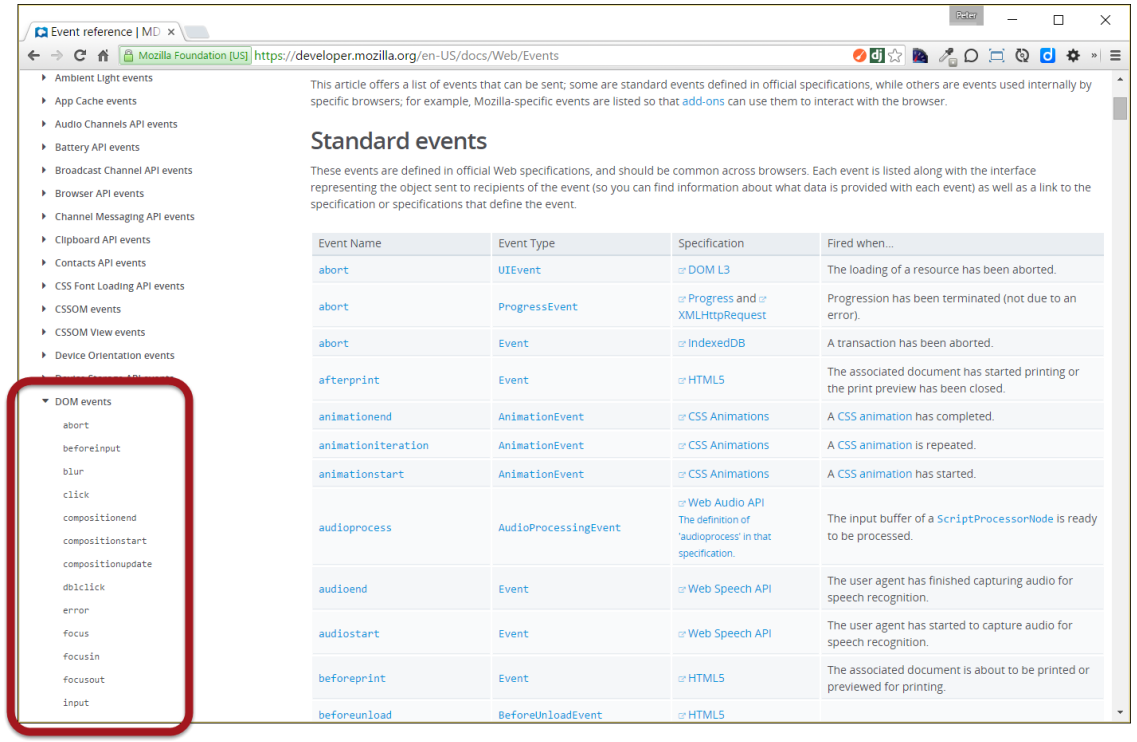
```
<div ng-click="handleClick()">…</div>
```

Angular 2:

```
<div (click)="handleClick()">…</div>

<input (blur)="onBlur()" />
```

DOM-events

Angular2 can listen to *each* DOM-event, without the use of a seperate directive:



https://developer.mozilla.org/en-US/docs/Web/Events

Example of event binding

HTML

```
<!-- Event binding for a button -->
<button class="btn btn-success"
      (click)="btnClick()">I am a button</button>
```

Class

```
→  export class AppComponent {
      …
      counter: number = 0;

      btnClick(){
         alert(‘You have clicked '+ ++this.counter +’
   times');
      }
    }
```


Hello Angular 2
Mijn favoriete steden zijn :
1 - Groninger
2 - Hengelo
3 - Den Haag
4 - Enschede
De pagina op localhost:3000 meldt het volgende:
Je hebt 1 keer geklikt
Voorkom dat deze pagina extra dialoogvensters weergeeft.
OK
Ik ben een button

Event binding with $event

HTML

```
<input type="text" class="input-lg" placeholder="“City..."
      (keyup)="onKeyUp($event)"><br>
<p>{{ txtKeyUp}}</p>
```

Class

```
// 2. Binding to a keyUp-event in the textbox
onKeyUp(event:any){
   this.txtKeyUp += event.target.value + ' - ';
}
```

Binding with local template variable

Declare *local template variable* with # → The whole element will be given to the component

Please note: bind to event, otherwise nothing happens.

```
<input type="text" class="input-lg" placeholder="City..."
      #txtCity (keyup.enter)="betterKeyUp(txtCity)">
<h3>{{ txtCity.value }}</h3>
```

Class:

```
  // 3. Binden aan keyUp-event via local template variable
 betterKeyUp(txtCity:string){
    //... Handle txtCity
 }
```

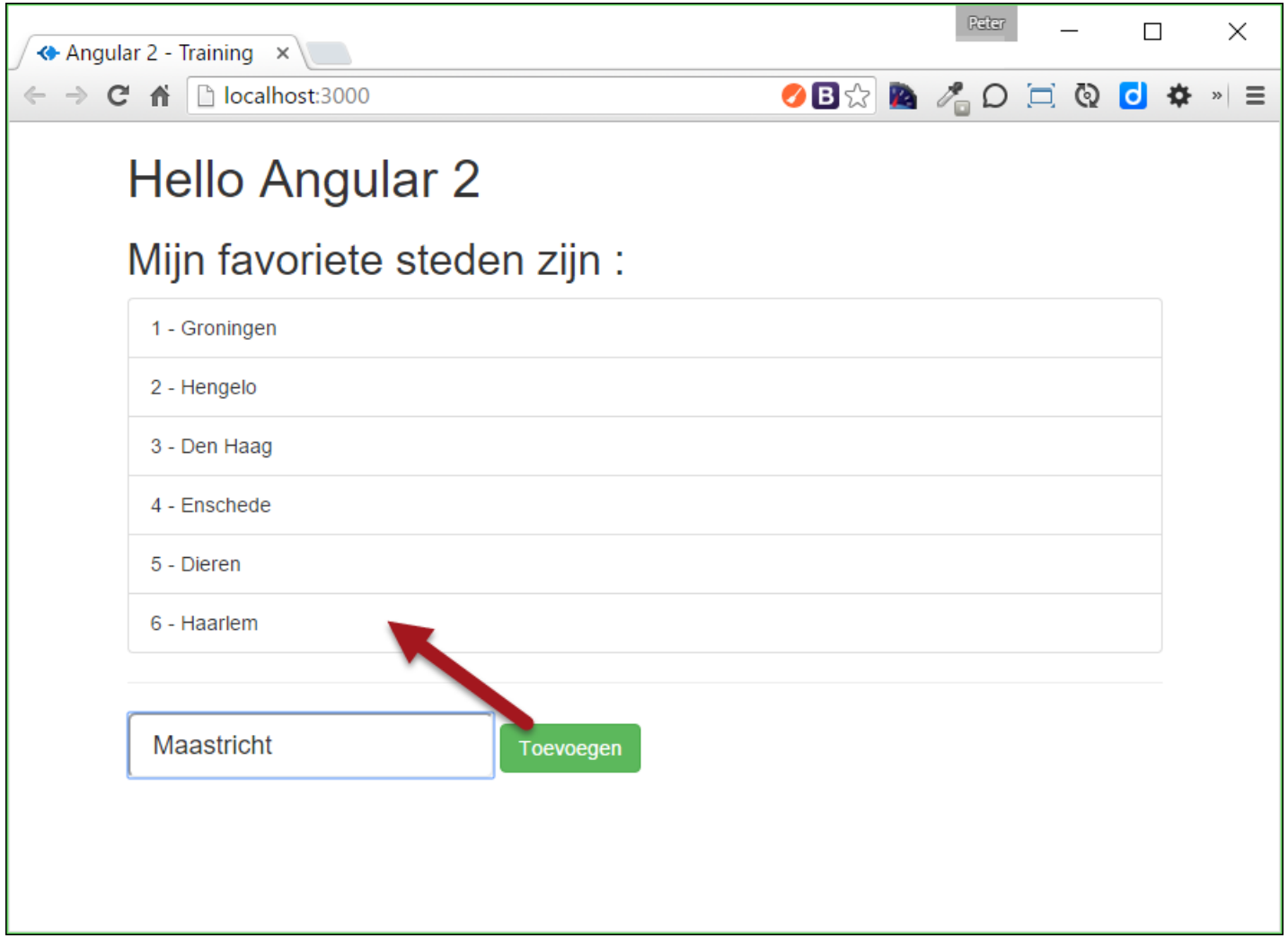
Putting it all together…

HTML

```
<input type="text" class="input-lg" placeholder="City..." #txtCity>
<button class="btn btn-success"
      (click)="addCity(txtCity)">Add
</button>
```

Class

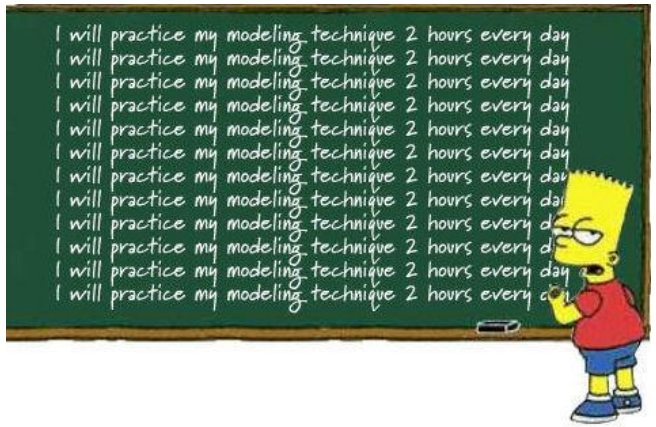
```
export class AppComponent {
   // Properties voor de component/class
   …
   addCity(txtCity) {
      let newID   = this.cities.length + 1;
      let newCity = new City(newID, txtCity.value, 'Onbekend');
      this.cities.push(newCity);
      txtCity.value = '';
   }
}
```



More info: https://angular.io/docs/ts/latest/guide/user-input.html

Checkpoint

- Event binding is done with: `(eventName)="…"`
- Events are processed with an event handler-function in the component
- Use # to declare a local template variable.
- This way we can easily create CRUD-operations



Exercise

- https://github.com/TeacherStijn/AngularVoorbeelden/blob/master/Exercises.pdf
- Exercise 2e, 2f, 2g
- Go to browser: http://localhost:4200

IK WIL

Angular Fundamentals Module 3 - Services

vijfhart IT-OPLEIDINGEN

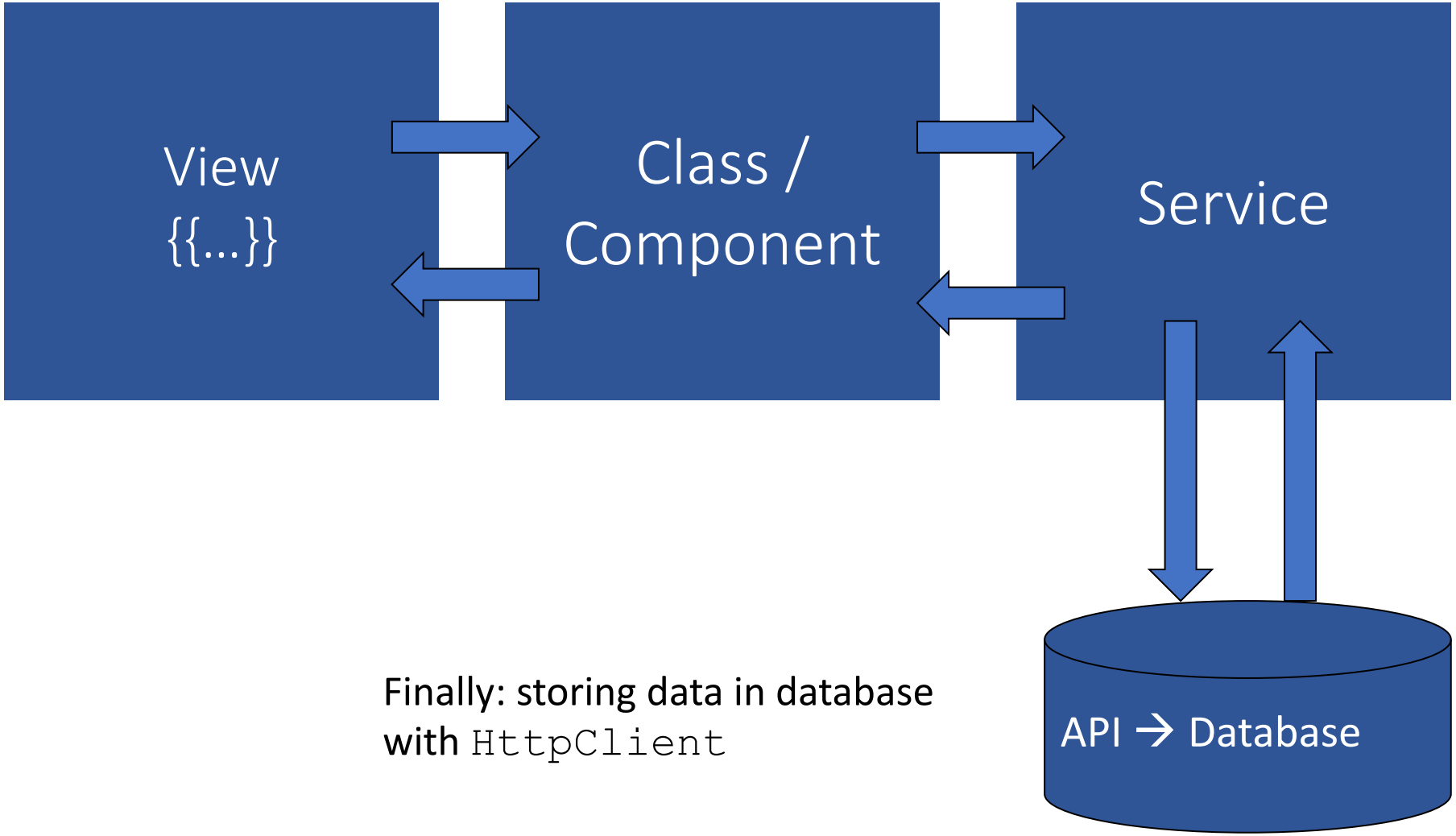
Services

Purpose: reusable datafunctionality for multiple components

- Data retrieval
- Data caching
- Data Storage,
- …

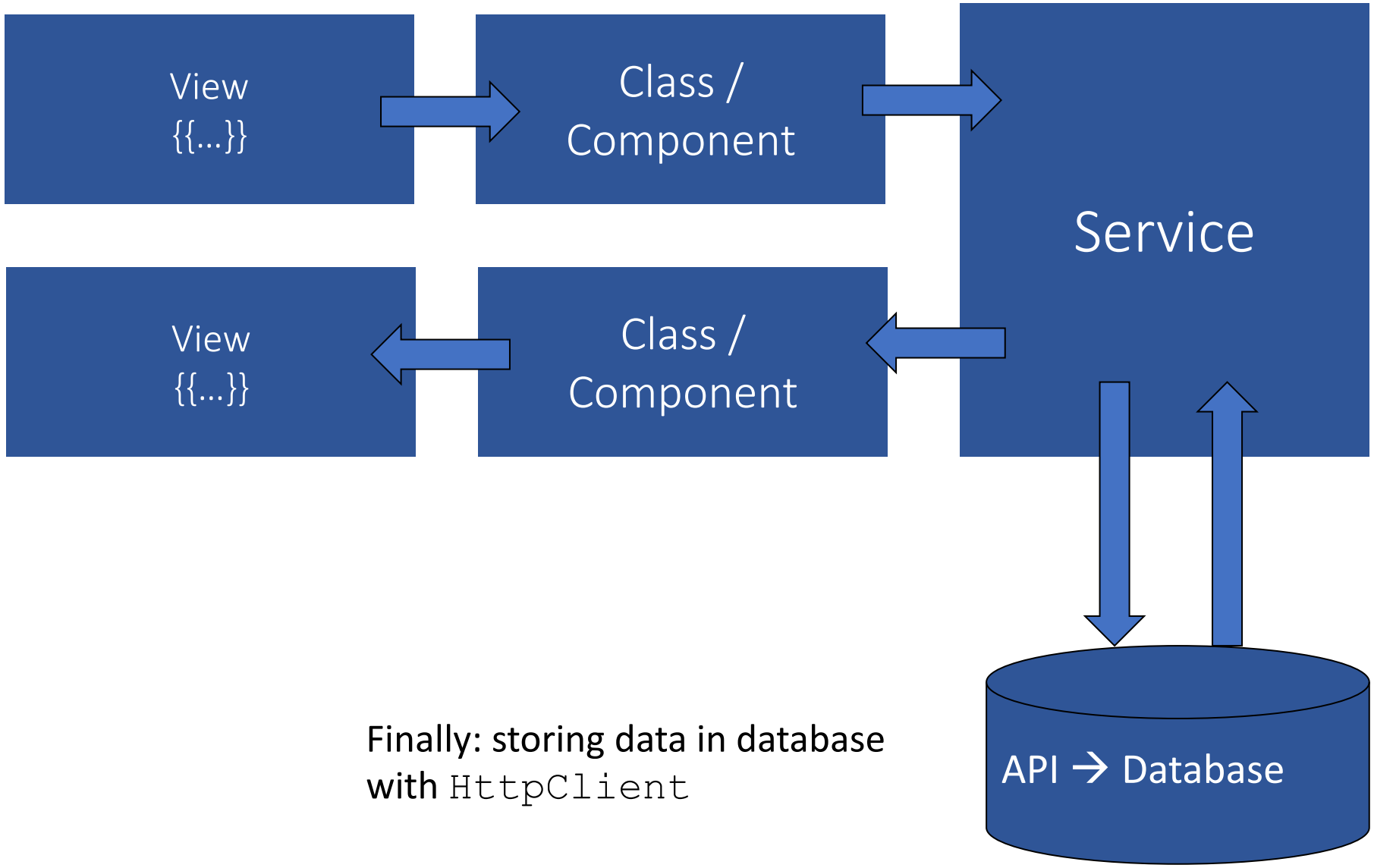
- In Angular 2 we have one option:
  - `export class myDataService { … }`

Data flow



Finally: storing data in database
with `HttpClient`

Data flow



Finally: storing data in database
with `HttpClient`

Services in Angular 2

Data services in AngularJS:

```
angular.module('myApp')
  .service(…)
  .factory(…)
  .provider(…)
```

Data services in Angular:

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

@Injectable()
export class CityService{
   //....
}
```

Role of @Injectable

Why? – Dependency Injection (DI) and metadata!

"TypeScript sees the @Injectable() decorator and emits metadata about our service, metadata that Angular may need to inject other dependencies into this service."

https://angular.io/docs/ts/latest/tutorial/toh-pt4.html

Step 1 – Making a service (static data)

```
import { Injectable } from '@angular/core';
import { City } from './city.model'

@Injectable()
export class CityService {
   cities:City[] = [
      new City(1, 'Groningen', 'Groningen'),
      …
   ];

   // retourneer alle cities
   getCities() {
      return this.cities
   }

   // retourneer city op basis van ID
   getCity(id:number) {
       return this.cities.find(c => c.id === id);
   }
}
```

Step 2 – Inject / consume the Service

```
…
import {CityService} from "./city.service";

@Component({
   selector   : 'hello-world',
   templateUrl: 'app/app.html',
})

export class AppComponent implements OnInit {
   // Properties voor de component/class
   currentCity: City;
   cities: City[];
   cityPhoto: string;

   constructor(private cityService: CityService) {

   }

   ngOnInit() {
      this.cities = this.cityService.getCities();
   }

   getCity(city: City) {
      this.currentCity = this.cityService.getCity(city.id);
      this.cityPhoto   = `img/${this.currentCity.name}.jpg`;
      console.log('City opgehaald:', this.currentCity);
   }
}
```

Constructor: shorthand for new private variable + instantiazing!

local variables

Call to the cityService

Instantiation?

- Important: nó `new()` instance of the Service!
  - Services are Singletons
  - Are retrieved from the Module and/*or* instantiazed in a `constructor()`

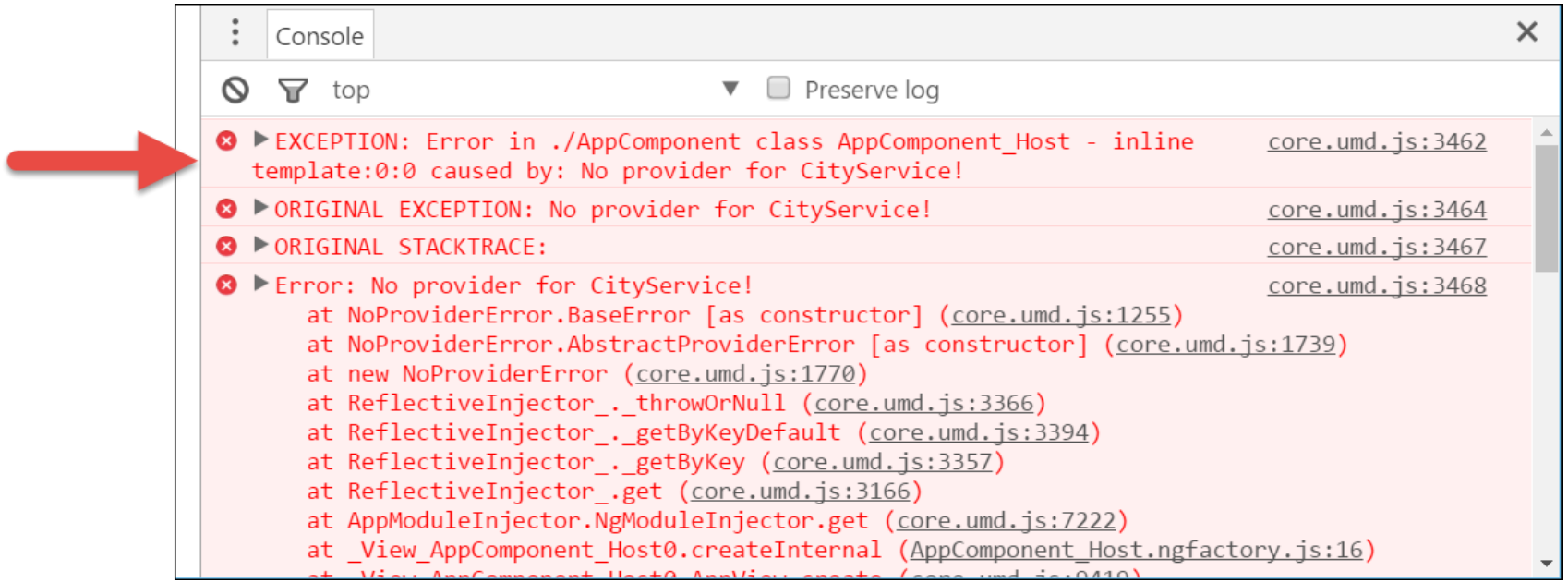
"The constructor itself does nothing.

The parameter simultaneously defines a private cityService property and identifies it as a CityService injection service."

```
constructor(private cityService:CityService) { … }
```

“No provider for CityService”

→ Solution: inject in `app.module.ts`



Option 1: service injecteren in Module

- Only the *reference* to CityService is not enough.
- Angular has to *inject* the service into the module
- Use the de annotation: `providers: [ … ]`

```
// Module declaration
@NgModule({
   imports      : [BrowserModule],
   declarations: [AppComponent],
   bootstrap    : [AppComponent],
   providers    : [CityService] // DI voor service
})
export class AppModule {
}
```

Array with Service-dependencies

Option 2 : Angular 6+, use `providedIn`

"Tree shakeable providers" – for optimizing performance with unused code parts

Not letting the module know whats services will be used, but instead provide the module name in which the service will be used in the service's @Injectable() annotation.

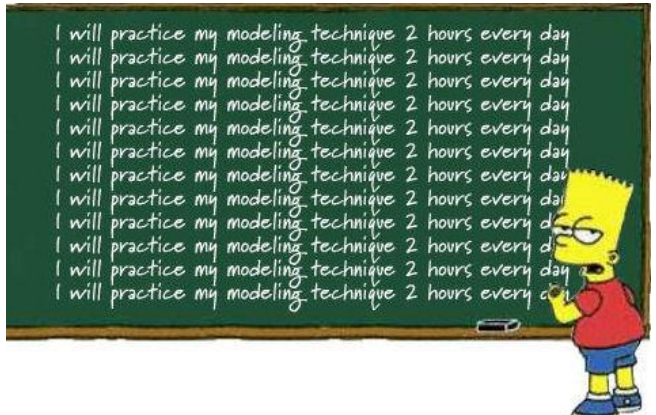
```
@Injectable({
   providedIn: 'root'
})
export class CityService {
       …
}
```

```
@NgModule({
   imports      : [BrowserModule],
   declarations: [AppComponent],
   bootstrap    : [AppComponent],
   // providers    : [CityService]
})
```

https://blog.angular.io/version-6-of-angular-now-available-cc56b0efa7a4

Checkpoint

- Each service in Angular 2 is a `class`
- Services are annotated with `@Injectable()`
- Import service in the component which uses it.
- Instantiate or get reference in `constructor()`
- Insert service in Module at `providers: []`



Exercise

- https://github.com/TeacherStijn/AngularVoorbeelden/blob/master/Exercises.pdf
- Exercise 3a, 3b, 3c
- Go to browser: http://localhost:4200

IK WIL

Angular – Module Observables

vijfhart
IT-OPLEIDINGEN

Async services with RxJS/Observables

Reactive programming with asynchronous streams

Async Services

- Retrieving static data: *synchronous* action
- Working with `HttpClient`: *Angular 4.3+*
- Angular 1: `Promises`
- Angular 2: `Observables`

Also in Angular 2: ReactiveX library `RxJS`

ReactiveX Introduction Docs Languages Resources Community

ReactiveX

An API for asynchronou
with observable stream

Choose your platform

ReactiveX Introduction Docs Languages Resources Community

Languages

- Java: RxJava
- JavaScript: RxJS
- C#: Rx.NET
- C#(Unity): UniRx
- Scala: RxScala
- Clojure: RxClojure
- C++: RxCpp
- Ruby: Rx.rb
- Python: RxPY
- Groovy: RxGroovy
- JRuby: RxJRuby
- Kotlin: RxKotlin
- Swift: RxSwift

ReactiveX for platforms and frameworks

- RxNetty
- RxAndroid
- RxCocoa

DOCUMENTATION	LANGUAGES	RESOURCES	COMMUNITY
Observable	RxJava	Tutorials	GitHub
Operators	RxJS		Twitter
Single	Rx.NET		Others

http://reactivex.io/

Why Observables?

We can do much more with observables than with promises.

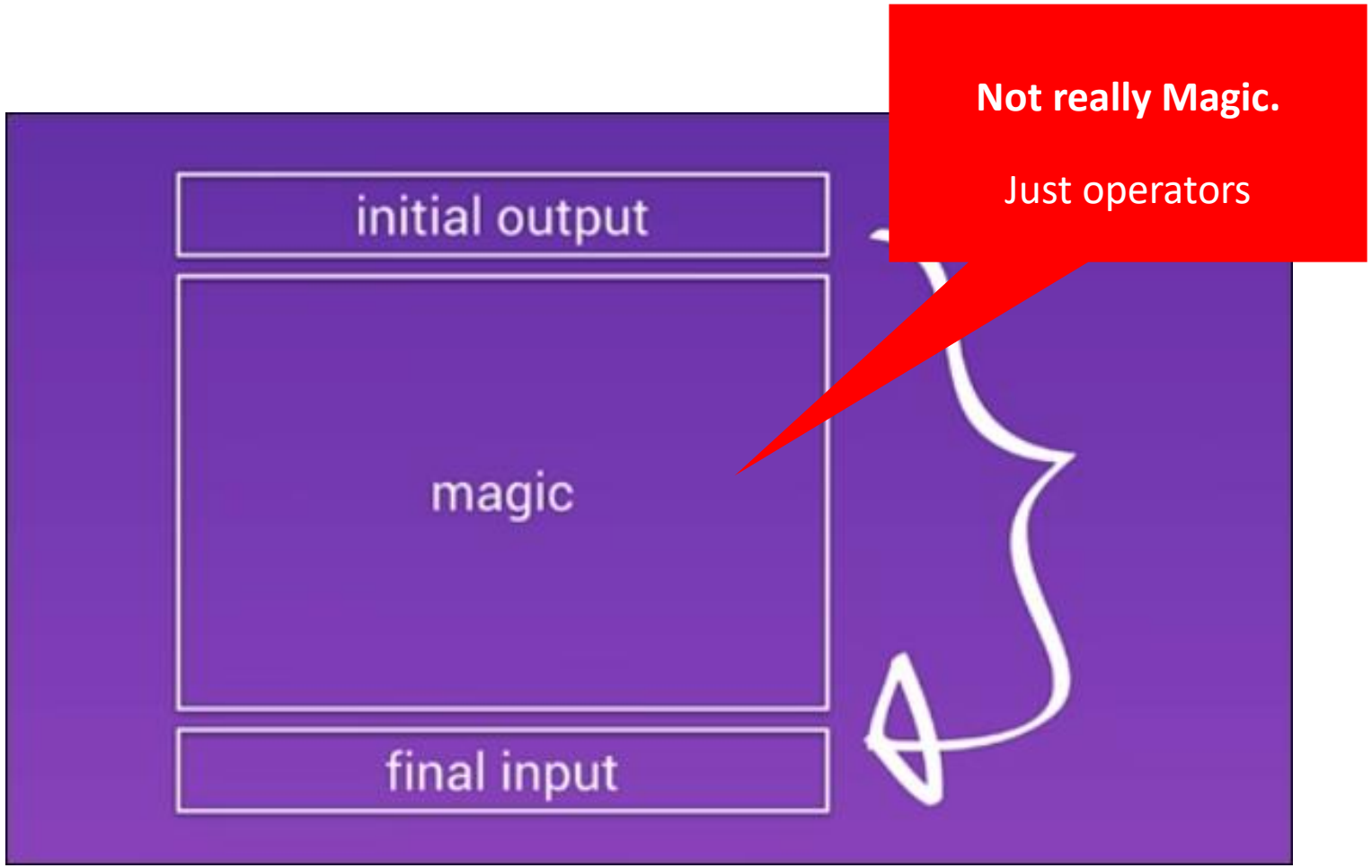
With observables, we have a whole bunch of operators to pull from, which let us customize our streams in nearly any way we want.

https://auth0.com/blog/2015/10/15/angular-2-series-part-3-using-http/

Observables and RxJs

- "Reactive Programming"
  - *"Reactive programming is programming with asynchronous data streams."*
  - https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
- Observables have more possibilities than Promises:
  - Mapping, Filtering, Combining, Cancel, Retry
  - ...
- Which means: no `.success()`, `.error()` and `.then()` chaining anymore!

"The observable sandwich"



In code

```
this.http.get<City[]>('assets/data/cities.json')
      .pipe(
         delay(…),
         map(…)
      )
     .subscribe((result) => {
       //... Do something
    });
```

Initial Output

Optioneel:

operator(s)

Final Input

Also: import `HttpClientModule` in `@ngModule`

```
// Angular Modules
...
import {HttpClientModule} from '@angular/common/http';
// Module declaration
@NgModule({
  imports     : [BrowserModule, HttpClientModule],
  declarations: [AppComponent],
  bootstrap   : [AppComponent],
  ...
})
export class AppModule {
}
```

Subscribe - only once per block!

Three parameters:

- success()
- error() -- Optional
- complete() -- Optional

```
this.cityService.getCities()

   .subscribe(cityData => {
         this.cities = cityData;
      },
      err => console.log(err),
      ()=> console.log('Getting cities complete...')
   )
```

RxJS-operators in de service

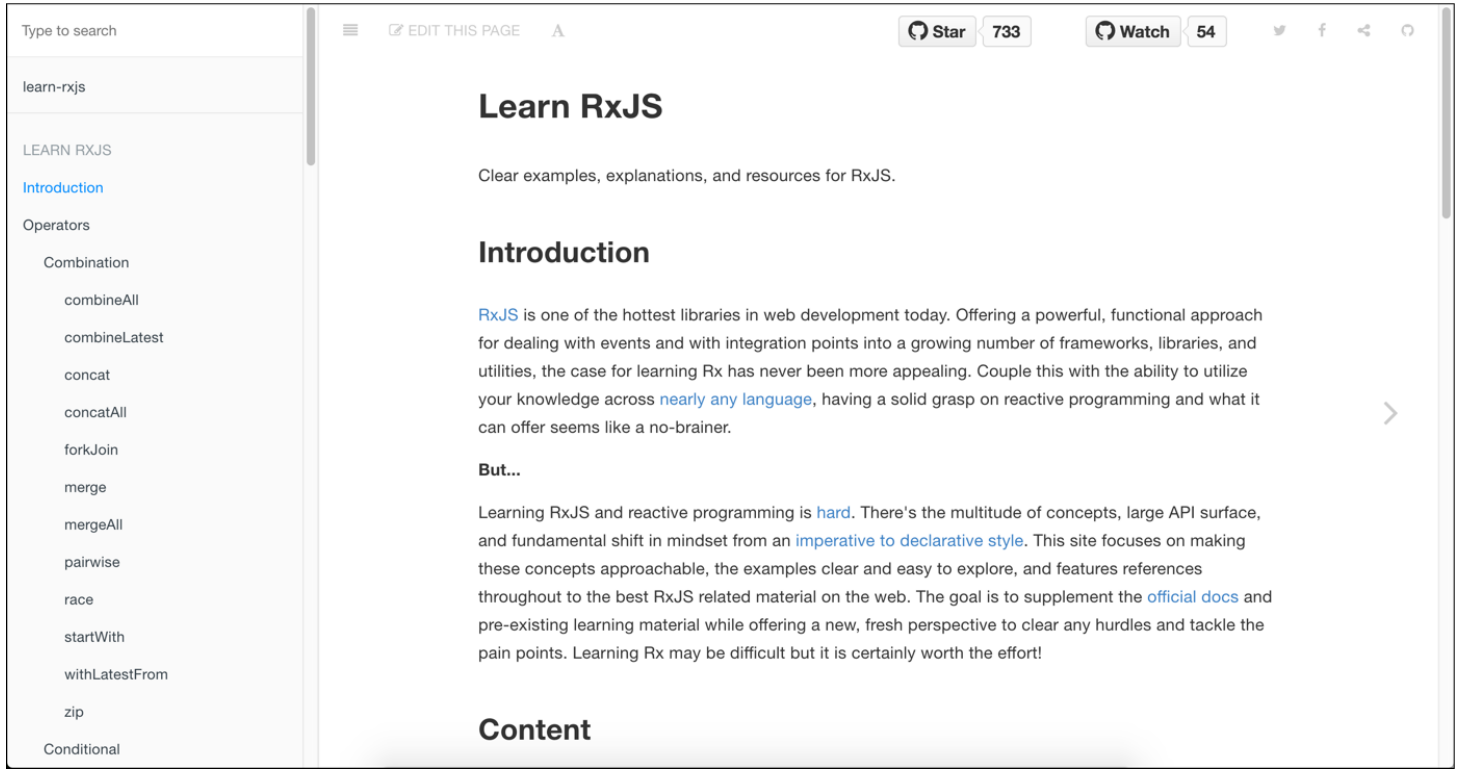
```
import {Injectable} from '@angular/core';
import {HttpClient} from "@angular/common/http";
import {map, delay, takeUntil, …} from "rxjs/operators";

@Injectable()
export class CityService {

   constructor(private http: HttpClient) {

   }

   // retourneer alle cities
   getCities(): Observable<Response> {
      return this.http.get('shared/data/cities.json')
         .pipe(…);
   }
}
```

Pipes: result of a function is the input of another

```
getCities() {
   if (!this.cities) {
      this.cityService.getCities()
         .pipe(
                delay(3000),
                retry(3)
                map(result => …),
                takeUntil(…condition…)
         )
      .subscribe(cityData => {
             this.cities = cityData;
       })
   }
}
```

Operators in .pipe()

https://www.learnrxjs.io/



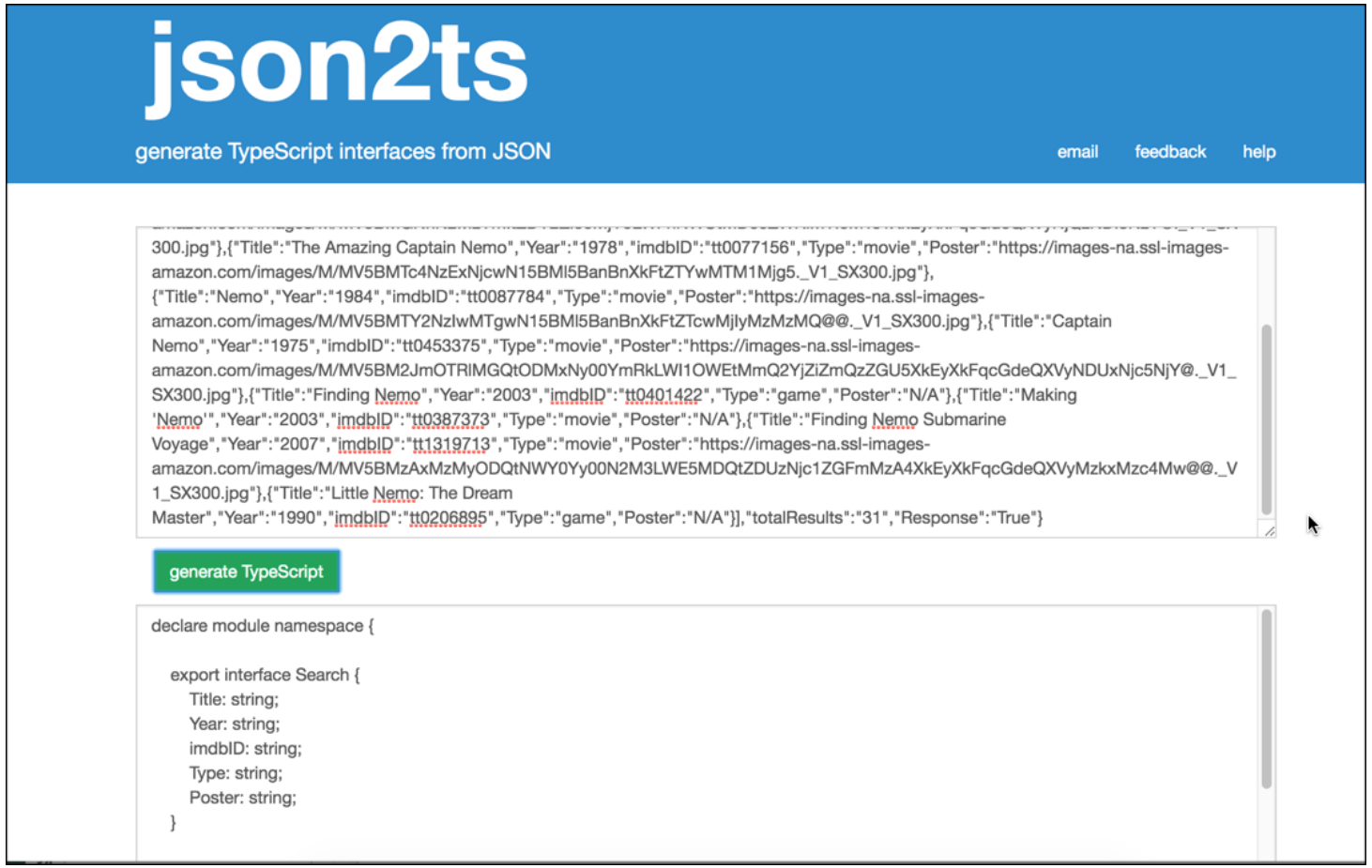
Voorbeeld API's

- **In oefen project:** **https://bgg-json.azurewebsites.net/collection/stinow**
- **See `GitHub` for more examples**
- https://pokeapi.co/ - Pokemon API
- http://openweathermap.org/API (weerbericht)
- http://filltext.com/ (random NAW-gegevens)
- http://ergast.com/mrd/ - Ergast Motor (F1) API
- http://www.omdbapi.com/ - Open Movie Database
- http://swapi.co/ - Star Wars API

More info + tooling

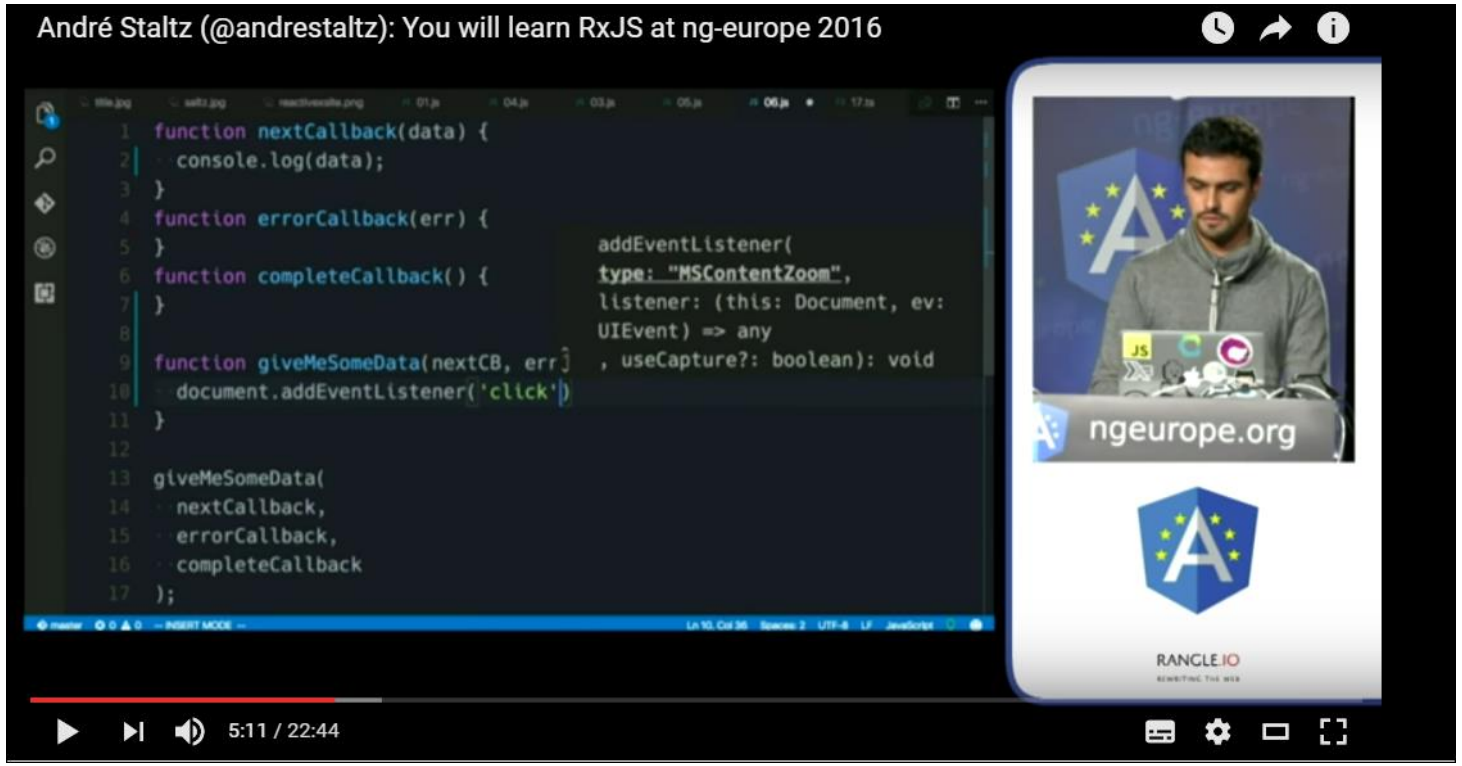
Some pointers to more information on the internet

Online JSON to TypeScript converter



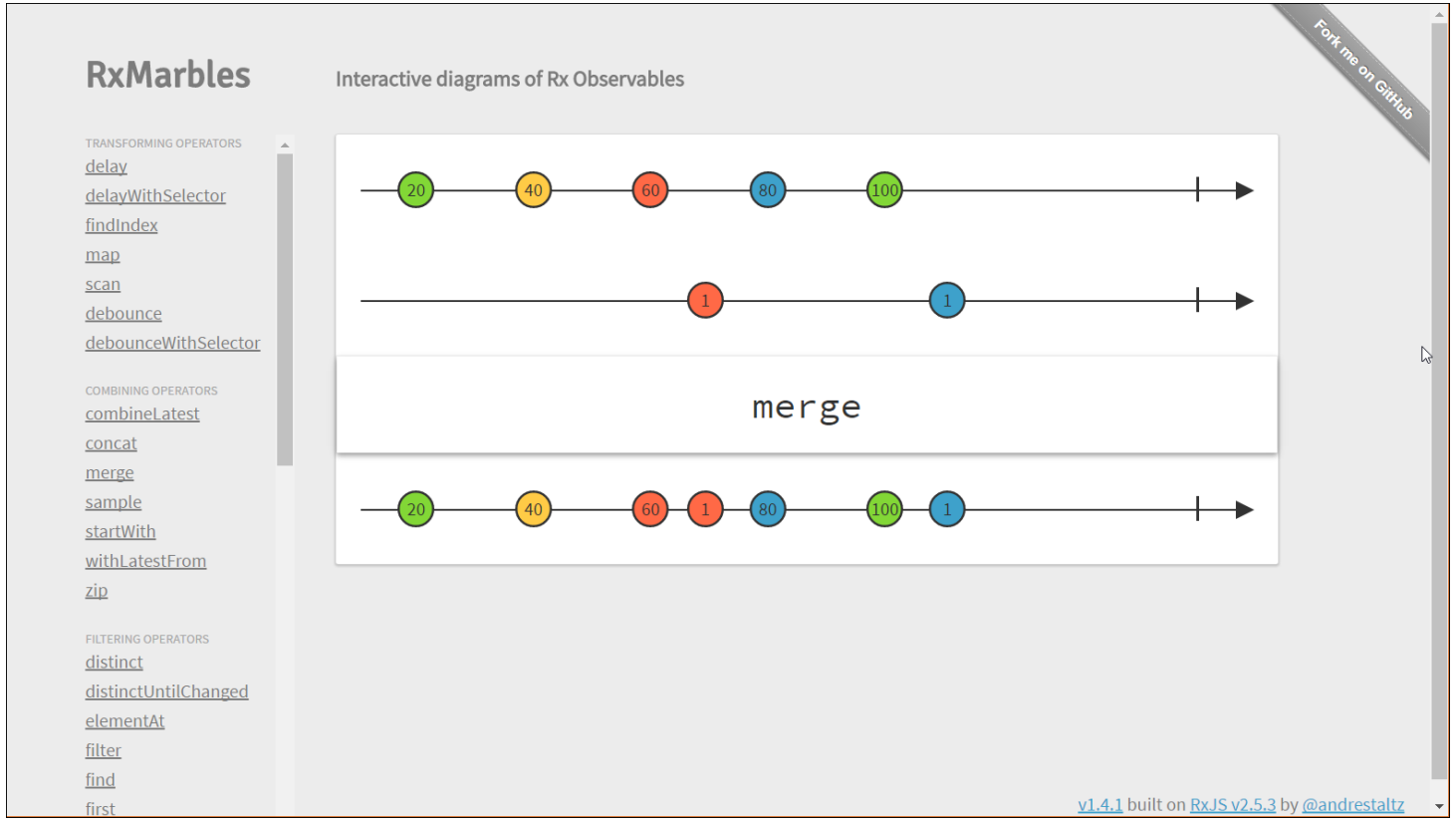
http://json2ts.com/

Creating Observables from scratch - André Staltz



https://www.youtube.com/watch?v=uQ1zhJHclvs

RxMarbles



http://rxmarbles.com/

Questions?

Let us know what you think in the comments! :-]

Vijfhart

Rokus Janssen, advisor and accountmanager with Vijfhart for KPN

If you have any questions or comments, please contact me for appropriate advice. You can reach me via:

E. r.janssen@5hart.nl
T. 088 542 78 88

See you soon!

Thank you for your attention!