HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN UNIVERSITY OF APPLIED SCIENCES

Introduction to **Angular**

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

Introduction to Angular (formerly AngularJs)

- short History
- basic approach
- Available technologies and examples
- Pros & Cons

Sources and Links

Official website: https://angular.io/
Tutorials : https://www.edureka.co/blog/angular-tutorial/
https://coursetro.com/courses/12/Learn-Angular-4-from-Scratch **(Video)**
(the http://www.w3schools.com/angular/default.asp is about Angular 1.x)

AngularJS / Angular – Overview 1/2

History

- 2009 Misko Hevery started to work on AngularJS
- he was a Google employee in a project called “Feedback”
- This project was based on very complex, highly ajaxified code written in 6 months
- Misko Hevery told that he can rewrite the whole thing in just 2 weeks using a OpenSource library he'd created as a Hobby.
- It took him 3 weeks (instead of 2 ;-)) to rewrite the entire application, the code decreased from 17000 to 1500 lines
- afterwards Google formed a team and started to develop **Angular.js**
- 2012 Version 1.0 officially released
- new (heavy changed) version 2.0 since summer 2016 as **Angular**
- new version 4 (version 3 was skipped) in March 2017
- **roadmap of Google: each 6 month a major new version !!**
- version Angular 5 (Nov 2017) with support for PWA
- version Angular 6 (May 2018) new cli-comands / optimizations
- version Angular 7 (Oct 2018)

Software type – Opensource under

- MIT – Licence: free also for closed SW with copyright notice

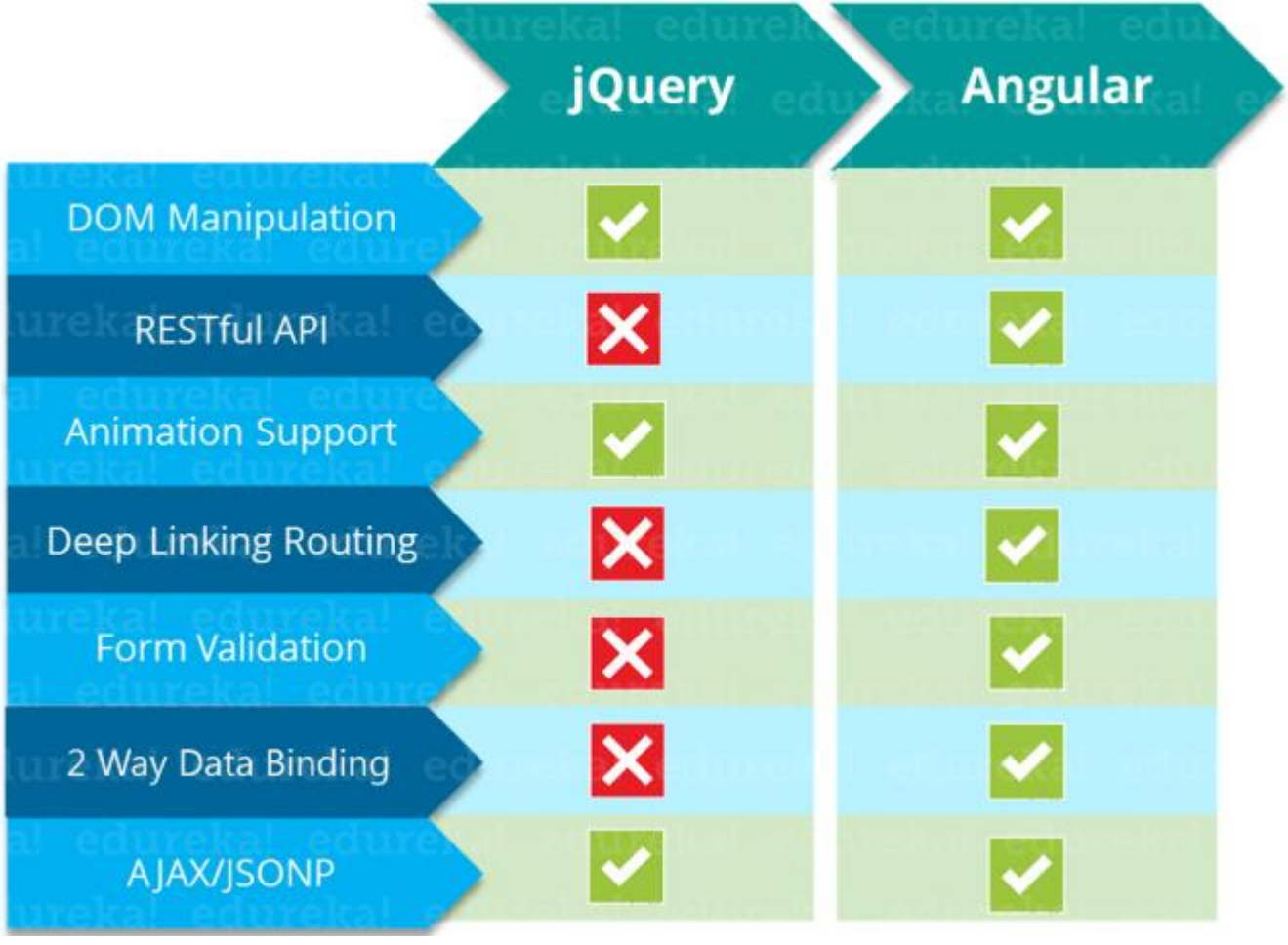
Specific features

- Unlike JQuery or ReactJS, Angular is a complete framework for building web apps (including JQuery Lite)
- Complete client-side solution → separation of client & server-side logic
- based on MVVM- Architecture (Model-View-ViewModel)
- Two-way data binding (data synchronisation between view and model)
- it is simply possible to extend the HTML with directives
  → produces better readable HTML-code
- The main purpose of jQuery is DOM manipulation. Compared to jQuery Angular offers a comprehensive toolset for a wide range of problems (routing, CRUD – Server side storage , etc.)

- **Angular is a framework for simplify building dynamic web apps and single page applications**

Angular – Comparison with JQuery

- Angular is a complete framework for building web apps (including JQuery Lite) and contains also server-oriented communication



Picture Source: https://www.edureka.co/blog/angular-tutorial/

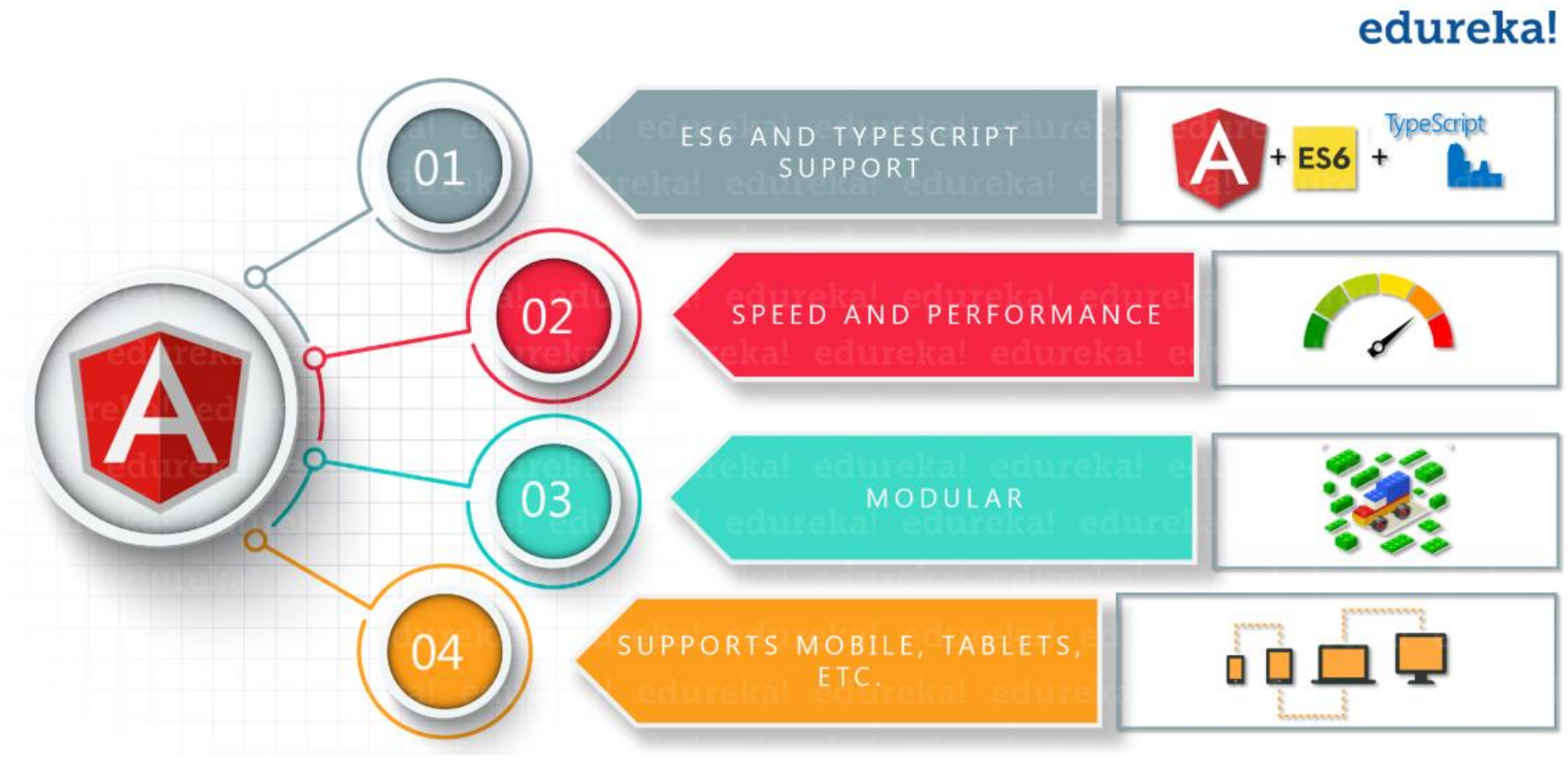
Differences between AngularJS and Angular 4 (5)

Angular is a complete rewrite of AngularJS (from Eureka).

- The Angular- architecture application is different from AngularJS.
- The main building blocks for Angular are modules, components, templates, metadata, data binding, directives, services and dependency injection.
- Angular does not have a concept of “scope” or controllers instead, it uses a hierarchy of components as its main architectural concept.
- Angular has a simpler expression syntax, focusing on “[ ]” for property binding, and “( )” for event binding

- **Mobile development** – Desktop development is much easier when mobile performance issues are handled first. **Thus, Angular first handles mobile development.**
- **Modularity** – Angular follows modularity. Similar functionalities are kept together in same modules. **This gives Angular a lighter & faster core.**

Main features of Angular 4 (5...7)



Picture Source: https://www.edureka.co/blog/angular-tutorial/

Install

- Home page and download from https://angular.io/
- For installation details see Quickstart at

  https://angular.io/guide/quickstart

- Load Angular it with the npm –installer with :

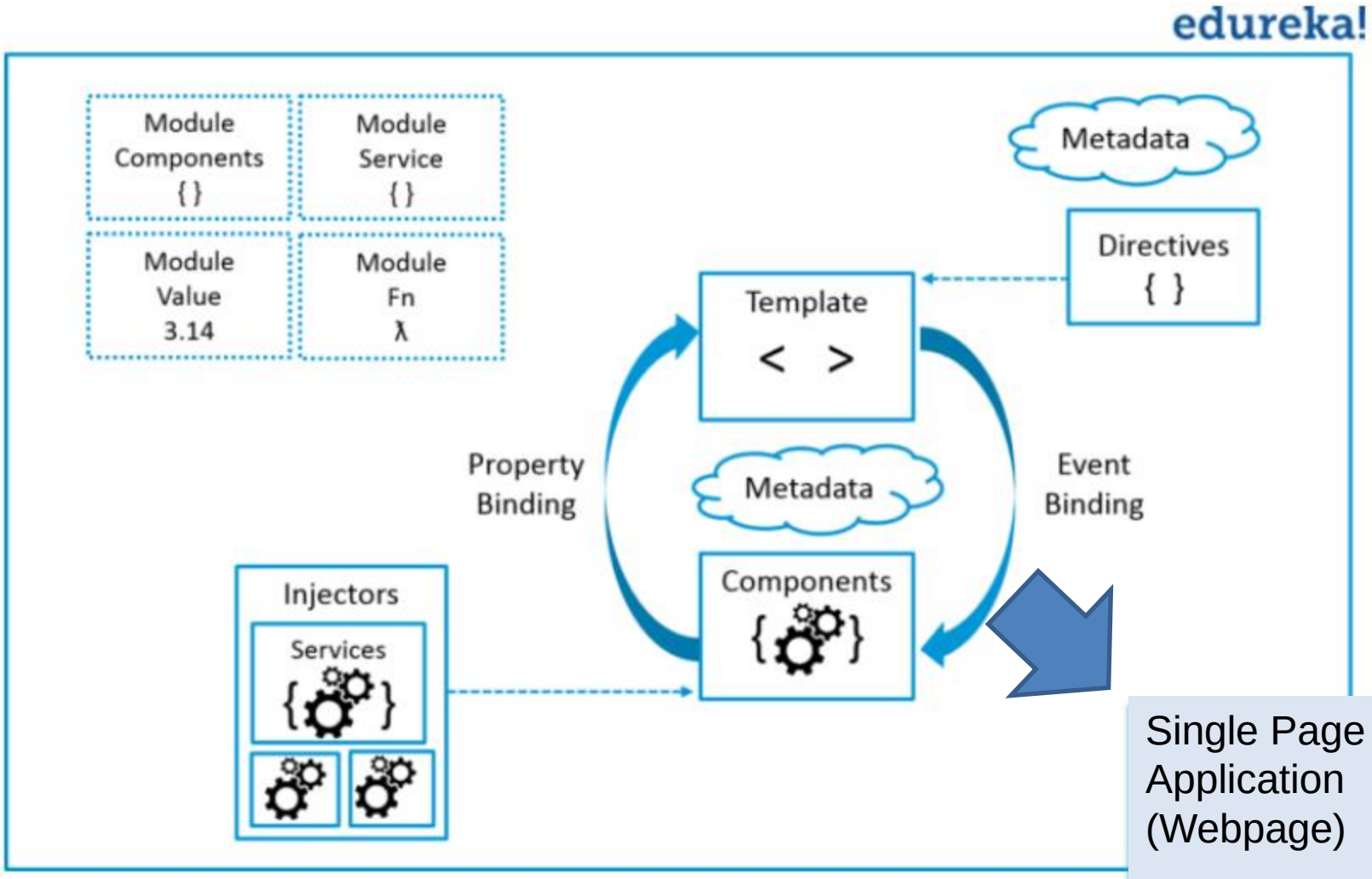
  npm install -g @angular/cli

CLI is the comand line interface of Angular
(the old Version 1.x is still at https://angularjs.org/ **do not use !)**

Versions – current 7.1.x

- Full version (well commented) or min. version (no comments./ compressed)
- **Attention:** The versions have different loading times!

Main architecture of Angular



Picture Source: https://www.edureka.co/blog/angular-tutorial/

Angular elements – Overview

Angular is a very complex framework based on a variety of concepts.

The most important concepts/terms are briefly mentioned here:

- **Templates define globally the design of the web app (mostly HTML)**

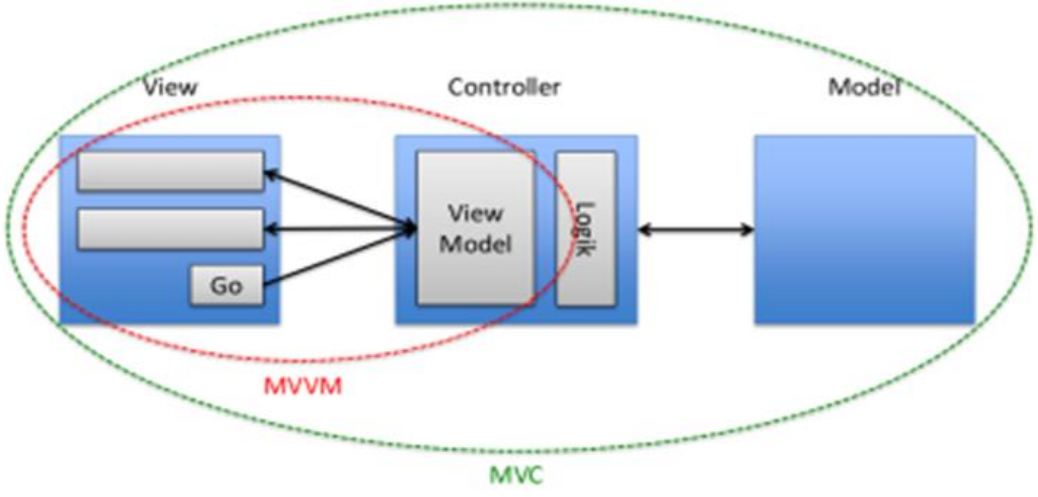
- **Model-View-ViewModel (the man structure of the apps -see next pages)**
- **Directives : extend the HTML tags**

- **Data Binding – link the view elements with the data model**
- **Expressions: for defining the output inside the HTML area**
- **Service: Global conversion services and HTTP-connections**

- **Filter: allow filtering of data lists**
- **Modules : link all parts together**
- **Dependency Injection – injects (includes) services**

Angular-concepts – MVVM 1/2

- Angular combines several design patterns
  - **M**odel-**V**iew-**C**ontroller
  - **M**odel-**V**iew-**V**iew**M**odel
- Therefore its also called a MVW-Model (Model-View-Whatever)
- The two-way data binding allows the view to change only if the associated model also changes
- Manual DOM manipulations as usual in jQuery, are not (or just hardly) necessary.



Angular - Creating and starting an application

- If the installation was successful, than a new Angular – application can be created by calling

  ng new my-app
  (This could take some time)

- After creation you start the new (demo) application by

  cd my-app
  ng serve –open

- In general, all ng *-commands are Angular-Command-line-interface tools ! **ng** itselfs refers to Angular !

- The **ng serve** compiles the TypeScript and starts a NodeJS-webserver ! In the mode above it stands open and watches for changes on the source files and compiles them automatically !

Angular – the main app directory structure

- each Angular app has the following root directory
- **my-app**
  - e2e — for testing
    - app.e2e-spec.ts
    - app.po.ts
    - tsconfig.e2e.json
  - node_modules/... — The additional modules of the Node-Server
  - **src/...** — **The source code (see next page)**
  - .angular-cli.json
  - .editorconfig
  - .gitignore
  - karma.conf.js — for testing
  - package.json — for testing
  - protractor.conf.js — for testing
  - README.md
  - tsconfig.json — TypeScript configuration

Angular – the main app directory structure

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN
UNIVERSITY OF APPLIED SCIENCES

- each Angular app consists of the following source code dirs and files:
- src — Main Source directory (dir)
  - app — the app dir
    - app.component.css — App CSS
    - **app.component.html** — **App HTML-Template**
    - app.component.spec.ts
    - app.component.ts — Typescript components
    - **app.module.ts** — **Typescript root module**
  - assets
    - .gitkeep
  - environments
    - environment.prod.ts
    - environment.ts
  - index.html
  - main.ts, polyfills.ts, test.ts — Global TS-modules
  - styles.css — Global CSS
  - favicon.ico
  - tsconfig.app.json, tsconfig.spec.json — Typescript Config

- The templates consist of HTML-code without JavaScript or TS –code !
- If you change the HTML-code, the ng serve will detect this, recompiles the app and refreshes it in the browser :

```
<h1>  Welcome  to Tom´s  {{title}} !  </h1>
```

- The **{{title}}** is the placeholder for TS-properties
- You can define the properties in the **app.component.ts** file:

```
…
export class AppComponent
{  title = ‘ Super Angular App';   }
```

- In the same way you can change the CSS of the webpage in **app.component.css**

```
h1
{ color: #369;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 250%;
 }
```

Angular-concepts : Modules

- An AngularJS module defines an application → min. 1 module required per app
- Modules are containers which can contain various parts of the application, such as further modules, controllers, services, filters, directives, etc.
- This is especially good for structuring and organizing the application, eg:
  - One module for each function
  - Modules for reusable components

- Angular is also divided into several modules (animate, route, cookies, etc.)
- Instead of a main method that instantiates and links the different parts of the application, Angular modules declaratively determine how an application should be booted

Angular – expressions and directives

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN
UNIVERSITY OF APPLIED SCIENCES

Expressions are included in the template code and define outputs:

- They are defined by {{ }} and can also do any calculations:

```
<p>The sum of 1 + 1 is {{1 + 1}}</p>
```

- Please note: inside expressions you can write most of JavaScript comands, but not: assignments with side effects like (=, +=, -=, ...), new … , or (inc / dec ++ and --)

Directives are included in the template code and control the output:

- Conditional display with ***ngIf**

```
<p *ngIf="super > 0">  The page is SUPER !<p>
```

- Loops with ***ngfor**

```
<ul><li *ngFor="let student of students">
                    {{ student }}    </li>  </ul>
```

  - where students is a vector defined in the TS-code:

```
students = ['Li', 'Shen', 'Zhou'];
```

- There are much more directives, like NgSwictch, Ngtemplate (please look at angular.io)

Angular – Binding events and data pipes

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN
UNIVERSITY OF APPLIED SCIENCES

- Any events of the HTML-DOM can be used for creating Angular actions:

```
<ul><li *ngFor="let student of students“  (click)="onSelect(student)"  >
                {{ student }}    </li>  </ul>
```

- The new event must be defined in the TS-code with

```
selectedStudent = " - - - ";   // Var. definition and default value
onSelect(student: string): void
        {  this.selectedStudent = student;   }
```

- The selected student can be displayed with

```
<h2>Details of {{ selectedStudent | uppercase }}
 </h2><div><span>The last student selected was </span>
                        {{selectedStudent}}.</div>
```

- In this example the additional option **| uppercase**
  is a so called PIPE ! It formats the expression !
  There are a lot of predefined pipes and you can define also your own pipes (see the Angular-tutorial) !

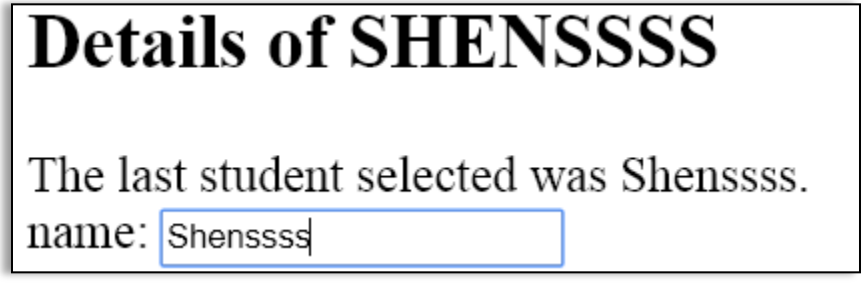
Angular – 2-way data binding

- Between the data variable and the View-elements can be definied **Two-way-data-bindings,** where changes are published to all instances
  <label>name: <input **[(ngModel)]="selectedStudent"** > </label>

- the new NgModel is also an NgDirective, but must be activated with

```
import { FormsModule } from '@angular/forms';
...
imports: [    BrowserModule,
              FormsModule  ],
```

in the file app.module.ts



Angular – Server-side data –import / injections

- It is also possible to communicate with the server

- First import the HttpClientModule into

```
// Import HttpClientModule from @angular/common/http
import {HttpClientModule} from '@angular/common/http';
… imports: [    BrowserModule,          …  HttpClientModule,
```

- Then the HTTP-connect is **INJECTED** into the component

```
results: string[];   // Inject HttpClient into your component or service.
 constructor( http: HttpClient) { }

ngOnInit(): void {    // Make the HTTP request:
this.http.get('/api/studentlist').subscribe(data => {
    // Read the result field from the JSON response.
      this.results = data['results'];      });
```

- The Request reads the external to data to the result-vector !

AngularJS vs. Angular

- There are many conceptual and syntactical differences between Angular 1 and Angular . The main differences are:

  - **Performance:**
    - ✓ Better performance by a lot of code optimization function in the build process f the Angular-CLI
  - **Mobile Support:**
    - ✓ In AngularJS 1 was no mobile support
    - ✓ In Angular there are libraries for fast mobile development
  - **TypeScript:**
    - ✓ In Angular Dart and TypeScript can be used
    - ✓ AngularJS1 use plain JavaScript
  - **Scope:**
    - ✓ Scopes are replaced by constructor()
  - **Component based Programming:**
    - ✓ AngularJS 1 based on modular programming
    - ✓ Angular is using component based programming

PROS

- Development is fast and simple once you are familiar with it
- Need less code for the same result as with lot of other frameworks/libraries
- easy to test
- very good for apps with lot of interactive client side code
- Two-way data binding
- Dependency Injection
- the possibility to extend HTML

CONS

- Basics are easy but with an growing complexity it is hard to learn
- Directives are powerful but difficult to use
- search engine indexability not always possible