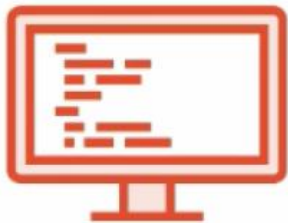


# DataBinding und EventHandling



DOM

`{{expression}}`

Interpolation

`[property] = "expression"`

One Way Binding

`(event) = "statement"`

Event Binding

`[(ngModel)] = "property"`

Two Way Binding



Component

# Stringinterpolation ⇔ Propertybinding

- Stringinterpolation (Expression-Binding) → Textersetzung

```
<h2>{{title}}</h2>
```

- Kurzschreibweise für Propertybinding
  - DOM-Property wird an Property der Component gesetzt

```
<h2 [textContent]=title></h2>
```

# Wenn DOM-Property und HTML-Attribut übereinstimmen

- Attribut src setzen

```

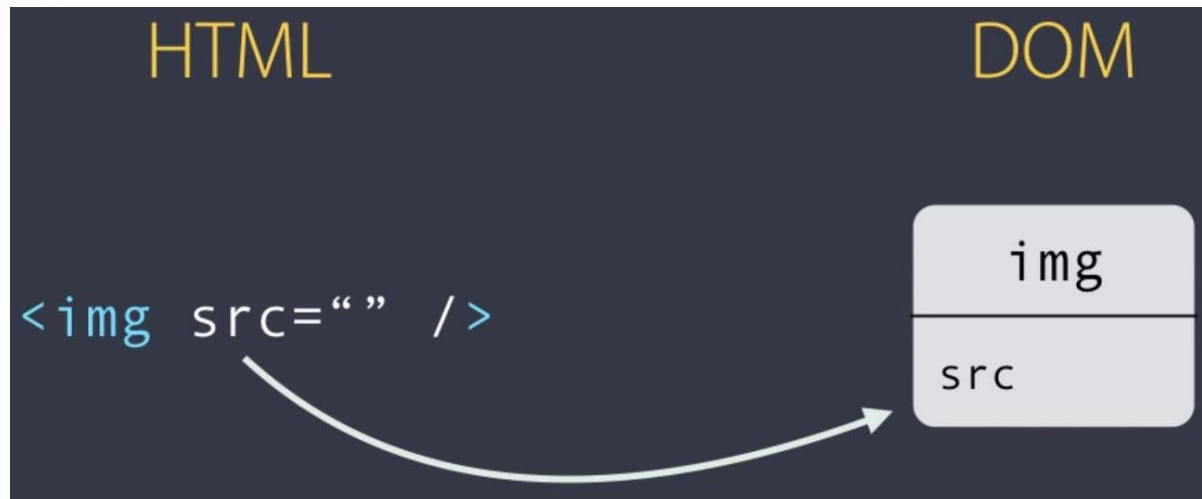
```

- DOM-Property src setzen

```
<img [src]="imageUrl" />
```

# HTML-Attribute ⇔ DOM-Properties

- Browser analysiert HTML und erstellt daraus das DOM (Document Object Model)



# Für Attribut gibt es kein Property

```
< [Angular] Can't bind to 'colspan' since it isn't a known proper  
ty of 'td'.  
<  
property colSpan of PupilsComponent  
  <td [colspan]="colSpan"></td>  
  </tr>  
</table>
```

- Fehlermeldung in Browser

```
✖ ▶ Uncaught Error: Template parse errors:  
Can't bind to 'colspan' since it isn't a known property of 'td'. ("  
  <table>  
    <tr>  
      <td [ERROR ->][colspan]="colSpan"></td>  
    </tr>  
  </table>
```

compiler.js:485

# Attribute speziell adressieren

| (tr) |
  | |

 colspan

## html attribute

charoff

# Bootstrap installieren

```
npm install bootstrap@3.3.7
```

- Installiert Bootstrap und trägt es in Dependencies ein

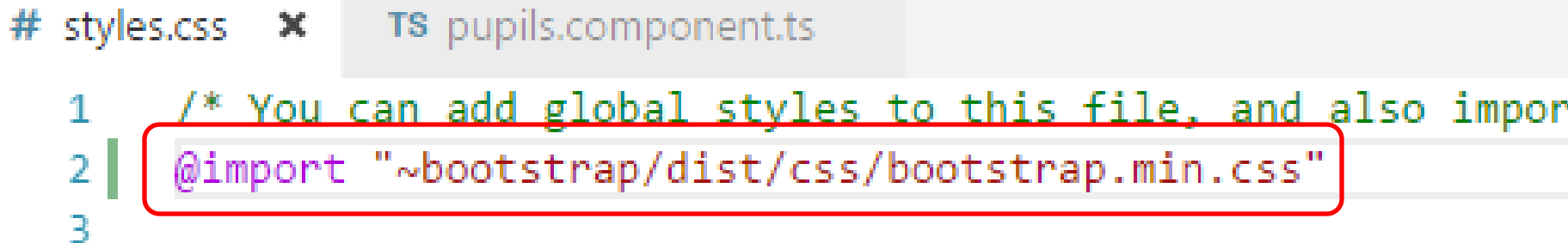
{} package.json x

```
14  "dependencies": {  
15    "@angular/animations": "^5.0.0",  
16    "@angular/common": "^5.0.0",  
17    "@angular/compiler": "^5.0.0",  
18    "@angular/core": "^5.0.0",  
19    "@angular/forms": "^5.0.0",  
20    "@angular/http": "^5.0.0",  
21    "@angular/platform-browser": "^5.0.0",  
22    "@angular/platform-browser-dynamic": "^5.0.0",  
23    "@angular/router": "^5.0.0",  
24    "bootstrap": "^3.3.7",  
25    "core-js": "^2.4.1",  
26    "rxjs": "^5.5.2",  
27    "zone.js": "^0.8.14"  
28  },
```

# Bootstrap-Import im styles.css

- Zusätzlich eintragen in styles.css

```
@import "~bootstrap/dist/css/bootstrap.min.css"
```

A screenshot of a code editor with two tabs: 'styles.css' and 'pupils.component.ts'. The 'styles.css' tab is active. The code in the editor is: 1 /\* You can add global styles to this file, and also import 2 @import "~bootstrap/dist/css/bootstrap.min.css" 3. The line '@import "~bootstrap/dist/css/bootstrap.min.css"' is highlighted with a red rounded rectangle. The line numbers 1, 2, and 3 are on the left side of the code block.

```
# styles.css x TS pupils.component.ts
1  /* You can add global styles to this file, and also import
2  |  @import "~bootstrap/dist/css/bootstrap.min.css"
3  
```



# Etwas einrücken

# styles.css ✕

```
1 | @import "~bootstrap/dist/css/bootstrap.min.css";  
2 | body { padding : 20px; }
```

Pupils demo

Save

# Gridsystem in Bootstrap

- Bildschirm wird in 12 Spalten eingeteilt
- Über Klassendefinition kann festgelegt werden, wie viele Spalten div benötigt
  - Z.B. `.col-md-3` → wird auf Bildschirm mit Auflösung  $\geq 992$  px Breite in 3 Spalten dargestellt
  - Werden mehr als 12 Spalten belegt, erfolgt automatisch ein Umbruch
- Offset → Einrückungen sind möglich (`.col-md-offset-3`)
- Responsive Design für verschiedene Auflösungen
  - Desktop: lg > 1200px
  - Tablet: md > 992px
  - Phablet sm > 768px
  - Handy xs < 768px
- Details auf Bootstrapseite: <http://getbootstrap.com/css/#grid>

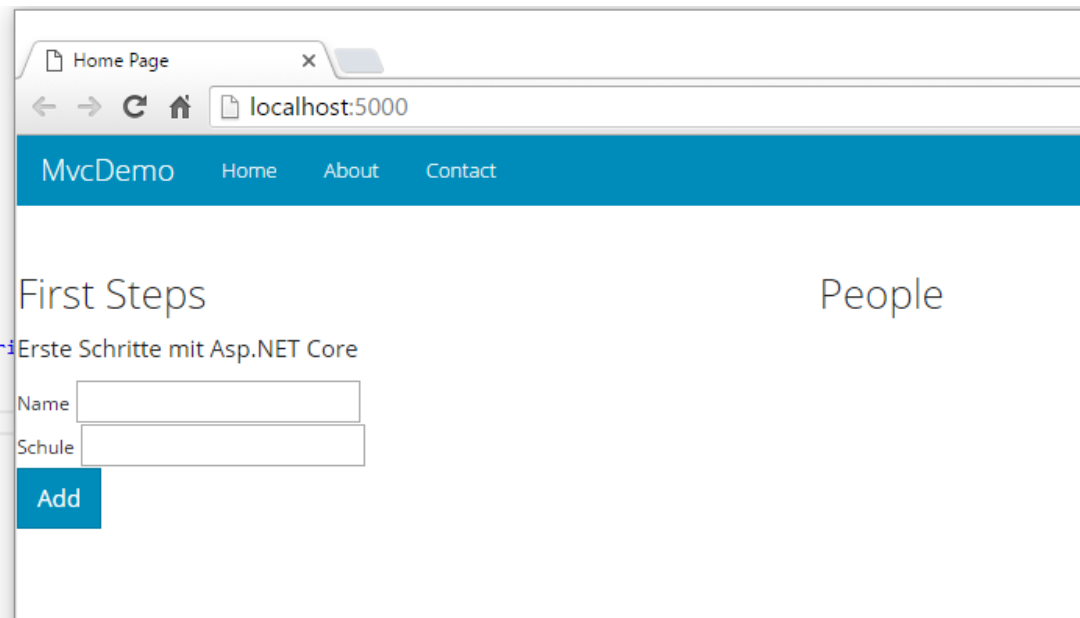
# Gridsystem - Beispiel

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-8								.col-md-4			
.col-md-4				.col-md-4				.col-md-4			
.col-md-6						.col-md-6					

# Indexseite mit zwei Spalten

- Links Formular, rechts Liste der Personen

```
<div class="row">
  <div class="col-md-6">
    <h3>First Steps</h3>
    <p>Erste Schritte mit Asp.NET Core</p>
    <form>
      <div>
        <label>Name</label>
        <input />
      </div>
      <div>
        <label>Schule</label>
        <input />
      </div>
      <input type="submit" class="btn btn-primary" value="Add" />
    </form>
  </div>
  <div class="col-md-6">
    <h3>People</h3>
  </div>
</div>
```



# class-Binding (im Beispiel mit Bootstrap)

```
@Component({
  selector: 'app-pupils',
  template: `
    <button class="btn" [class.btn-primary]="isPrimary">Save</button>

    get isPrimary(): boolean{
      return false;
    }
  `
})
```

Pupils demo

Save

Pupils demo

Save

# style-Binding

```
<button class="btn" [style.backgroundColor]="isPrimary ? 'blue' : 'red'">  
  Save  
</button>
```

Pupils demo

Save

Pupils demo

Save

# EventBinding

```
<button class="btn btn-primary" (click)="onSave()" >  
  Save  
</button>
```

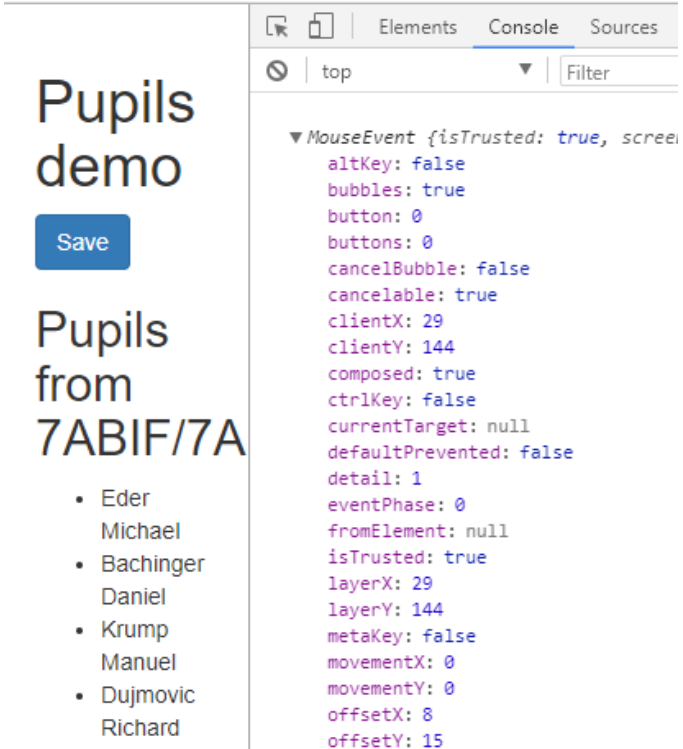
```
onSave() {  
  console.log('Save clicked!');  
}
```

# EventBinding - Parameter

```
<button class="btn btn-primary" (click)="onSave($event)" >  
  Save  
</button>
```

- \$event kommt von Standard-JavaScript

```
onSave(event) {  
  console.log(event);  
}
```



The screenshot shows a web application titled "Pupils demo" with a blue "Save" button. Below the button is a list of names: "Pupils from 7ABIF/7A" followed by "Eder Michael", "Bachinger Daniel", "Krump Manuel", and "Dujmovic Richard". The browser's developer console is open, displaying the details of a `MouseEvent` object. The console shows various properties such as `isTrusted: true`, `clientX: 29`, and `clientY: 144`.

Pupils demo

Save

Pupils from 7ABIF/7A

- Eder Michael
- Bachinger Daniel
- Krump Manuel
- Dujmovic Richard

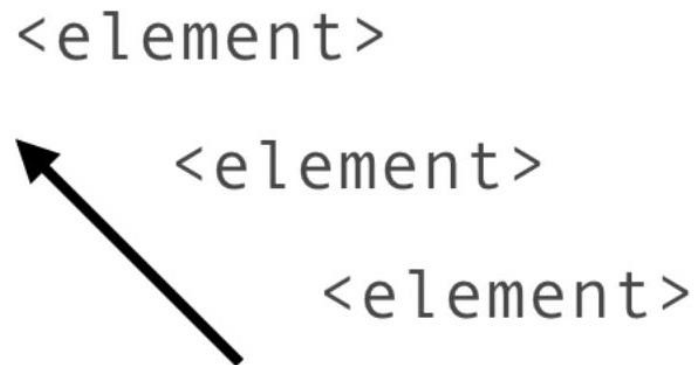
▼ MouseEvent {isTrusted: true, screenX: 29, screenY: 144, ...}

altKey: false  
bubbles: true  
button: 0  
buttons: 0  
cancelBubble: false  
cancelable: true  
clientX: 29  
clientY: 144  
composed: true  
ctrlKey: false  
currentTarget: null  
defaultPrevented: false  
detail: 1  
eventPhase: 0  
fromElement: null  
isTrusted: true  
layerX: 29  
layerY: 144  
metaKey: false  
movementX: 0  
movementY: 0  
offsetX: 8  
offsetY: 15



# EventBubbeling

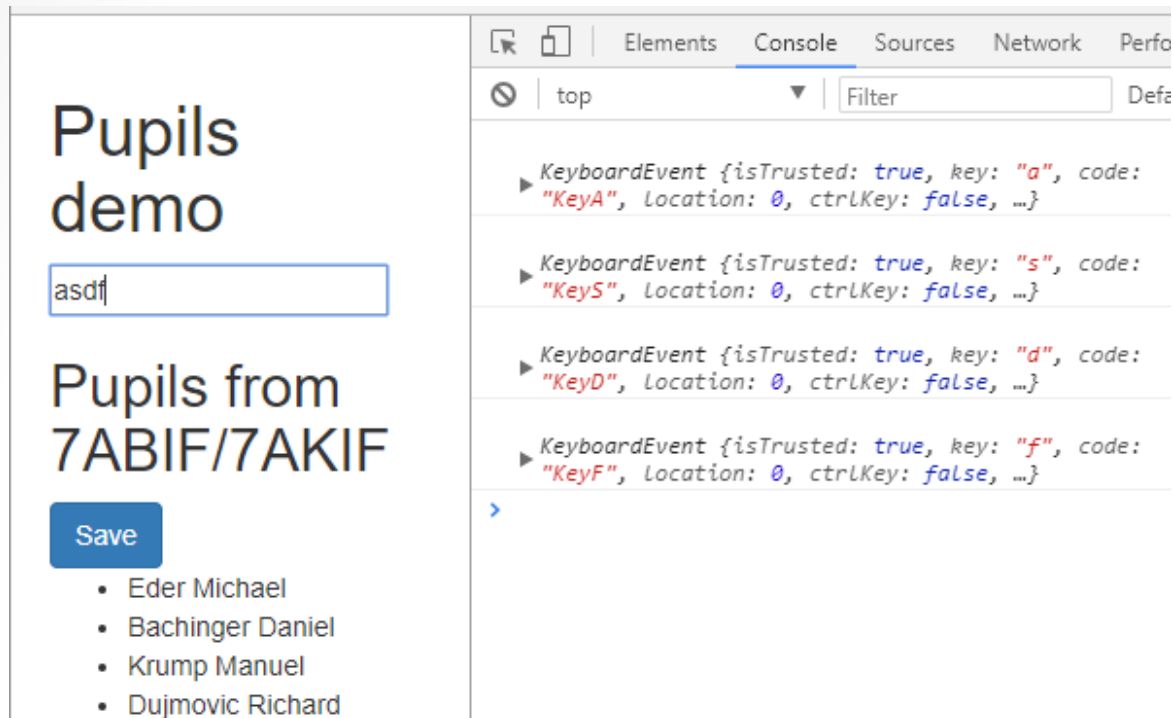
- Events „bubblen“ den DOM entlang nach oben
  - `event.stopPropagation()` beendet die Weiterreichung



# Input-Element - KeyStrokes

```
<input (keyup)="onKeyUp($event)"/>
```

```
onKeyUp(event) { console.log(event); }
```



The screenshot shows a web application on the left and a browser's developer console on the right. The web application has a title 'Pupils demo', an input field containing 'asdj', a subtitle 'Pupils from 7ABIF/7AKIF', a 'Save' button, and a list of names: Eder Michael, Bachinger Daniel, Krump Manuel, and Dujmovic Richard. The browser console on the right shows four log entries, each a KeyboardEvent object, corresponding to the key strokes 'a', 's', 'd', and 'f' in sequence.

**Pupils demo**

asdj

**Pupils from 7ABIF/7AKIF**

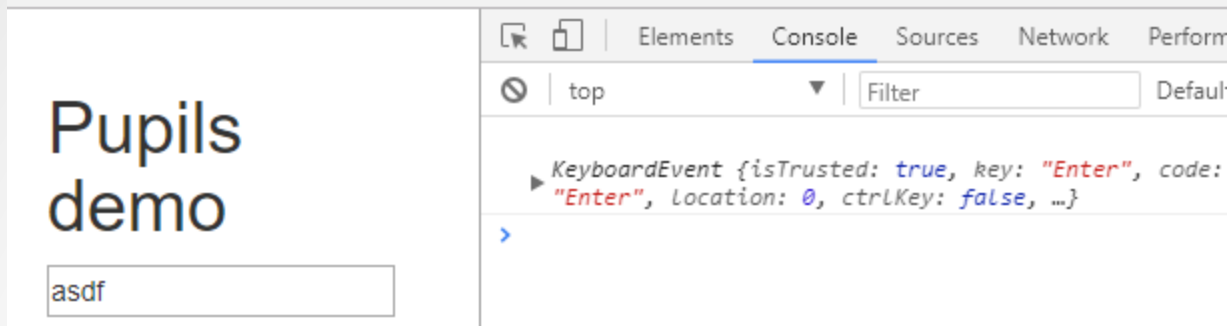
**Save**

- Eder Michael
- Bachinger Daniel
- Krump Manuel
- Dujmovic Richard

**Browser Console Log:**

- KeyboardEvent {isTrusted: true, key: "a", code: "KeyA", location: 0, ctrlKey: false, ...}
- KeyboardEvent {isTrusted: true, key: "s", code: "KeyS", location: 0, ctrlKey: false, ...}
- KeyboardEvent {isTrusted: true, key: "d", code: "KeyD", location: 0, ctrlKey: false, ...}
- KeyboardEvent {isTrusted: true, key: "f", code: "KeyF", location: 0, ctrlKey: false, ...}

# Nur bei Enter auslösen



```
onKeyUp(event) { if ( event.keyCode === 13) { console.log(event); } }
```

# Einfacher mit EventFilter

```
<input (keyup.enter)="onKeyUp($event)"/>
```

```
onKeyUp(event) { console.log(event); }
```

# TemplateVariables

- Lokale Variable im Template vom Typ input

```
<input #mail (keyUp.enter)="onEnter(mail.value)" />
```

- Nur mehr Value übergeben

```
onEnter(mail) { console.log(mail); }
```

# Property mail synchron mit input-Element

- Property in Component

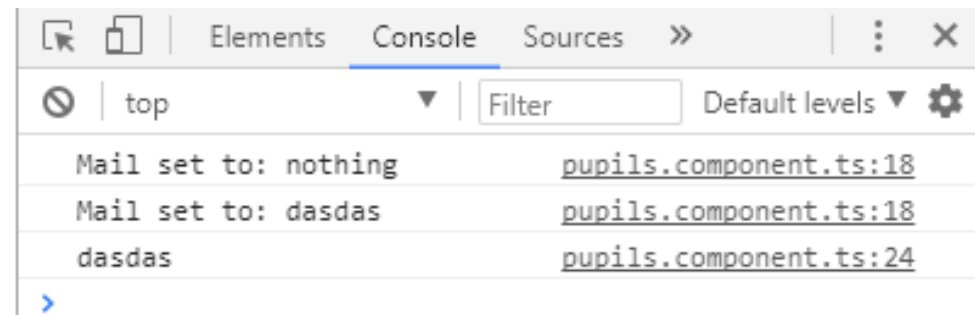
```
private _mail : string;  
get mail():string { return this._mail; }  
set mail(mail: string) { this._mail = mail; }
```

- Start mit Defaultinhalt
- Über Eingabe änderbar
- Button setzt Inhalt auf Standardtext
- TwoWayBinding mit EventBinding und PropertyBinding

## Pupils demo

Set Mail to nothing

## Pupils demo

Set Mail to nothing

# Umsetzung

- Html: Event- und PropertyBinding

```
<input #mailInput (keyUp.enter)="onEnter(mailInput.value)" [value]="mail" />
<button class="btn btn-primary" (click)="onSetMailToNothing()" >
| Set Mail to nothing
</button>
```

- Component: Property, EventHandler und ButtonHandler

```
private _mail : string = 'Default Mail';
get mail():string { return this._mail; }
set mail(mail: string) { console.log('Mail set to: '+mail); this._mail = mail; }
```

```
onSetMailToNothing(){ this.mail='nothing'}
```

```
onEnter(mailText) { console.log(this.mail = mailText); }
```

# Vereinfachung: TwoWayBinding (ngModel)

- Directive ngModel
  - Keine Strukturveränderung im DOM → kein \* wie (\*ngFor, \*ngIf)
- Merkhilfe „banana in a box“

```
<input [(ngModel)]="mail" />  
<button class="btn btn-primary" (click)="onSetMailToNothing()" >  
| Set Mail to nothing  
</button>
```

- Nur für DOM-Properties namens value einsetzbar
  - Sonst Property- und EventBinding



# ngModel ist kein DOM-Property

```
<input (keyUp.enter)="onEnter(mailInput.value)" [(ngModel)]="mail" />  
<button class="btn btn-primary" (click)="onSetMail()">  
  Set Mail to nothing
```

[Angular] Can't bind to 'ngModel' since it isn't a known property of 'input'.

- FormsModule muss in app.module.ts registriert werden

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

```
import { AppComponent } from './app.component';  
import { PupilComponent } from './pupil/pupil.component';  
import { PupilsComponent } from './pupils.component';  
import { PupilsService } from './pupils.service';
```

```
@NgModule({  
  declarations: [  
    AppComponent,  
    PupilComponent,  
    PupilsComponent  
  ],  
  imports: [  
    BrowserModule,  
    FormsModule
```

# Ausgabeformatierung → Pipes

## ■ Mitgelieferte Pipes (angular.io)

**P** AsyncPipe

**P** DecimalPipe

**P** DeprecatedDecimalPipe

**P** I18nSelectPipe

**P** PercentPipe

**P** UpperCasePipe

**P** CurrencyPipe

**P** DeprecatedCurrencyPipe

**P** DeprecatedPercentPipe

**P** JsonPipe

**P** SlicePipe

**P** DatePipe

**P** DeprecatedDatePipe

**P** I18nPluralPipe

**P** LowerCasePipe

**P** TitleCasePipe

# Ausgangspunkt: Klasse Pupil

- Properties mit unterschiedlichen Datentypen

```
constructor() {  
  this.firstName='Max';  
  this.lastName='Mustermann';  
  this.birthDate=new Date('2000-06-15');  
  this.rating=4.96;  
  this.description='Lorem ipsum dolor sit amet, consectetur adipiscing elit.'  
  ' Aenean commodo ligula eget dolor. Aenean massa. Cum sociis et  
  ' Aenean commodo ligula eget dolor. Aenean massa. Cum sociis et  
}
```

<> pupil.component.html ✕

```
1  {{firstName}} {{lastName}} <br/>  
2  {{rating}} <br/>  
3  {{birthDate}} <br/>  
4  {{description}} <br/>
```

# Ausgabe im Browser

- Aufgaben
  - Rating mit einer Nachkommastelle gerundet
  - Geburtsdatum als Datum
  - Description mit gekürztem Text

## Angular demo

Max Mustermann

4.96

Thu Jun 15 2000 02:00:00 GMT+0200

(Mitteleuropäische Sommerzeit)

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Aenean commodo ligula eget dolor. Aenean massa. Cum sociis

## Angular demo

Max Mustermann

5.0

15.6.2000

Lorem ipsum dolor sit amet, consectetur adipiscing...

# Zahlen formatieren → DecimalPipe

- <https://angular.io/api/common/DecimalPipe>

```
{{rating | number:'1.1-1'}} <br/>
```

# CustomPipe erzeugen

- Wieder mit CLI

```
PS D:\Angular\Uebungen\03_FirstAngular\pupils-demo> ng g p shortenText
create src/app/shorten-text.pipe.spec.ts (208 bytes)
create src/app/shorten-text.pipe.ts (211 bytes)
update src/app/app.module.ts (679 bytes)
```

TS shorten-text.pipe.ts ✕

```
1  import { Pipe, PipeTransform } from '@angular/core';
2
3  @Pipe({
4    name: 'shortenText'
5  })
6  export class ShortenTextPipe implements PipeTransform {
7
8    transform(value: any, args?: any): any {
9      return null;
10    }
11
12  }
```

# Shorten mit Ziellänge als Parameter

```
{{description | shortenText: '50'}} <br/>
```

```
@Pipe({
  name: 'shortenText'
})
export class ShortenTextPipe implements PipeTransform {
  transform(text: string, limit?: number): any {
    if (!text) return null;
    let length = limit ? limit : 20;
    return text.substring(0,length)+'...';
  }
}
```

# Components stylen über css

- Standardeinstellungen werden von styles.css geerbt
- Design der Component bleibt auf Komponente gekapselt
  - ShadowDOM

```
rating.component.css ●  
  
1  .glyphicon {  
2      color : darkblue;  
3      cursor: pointer;  
4  }
```