Directive

- Directives erlauben es, den DOM zu manipulieren
- Änderung der Struktur ➔ structural directives (Präfix *)
  - *ngIf
  - *ngFor
- Änderung des Inhalts oder Verhaltens von Elementen ➔attribute directives
  - ngModel
- Components sind Directives mit einem Template
- CustomDirectives sind ebenfalls möglich

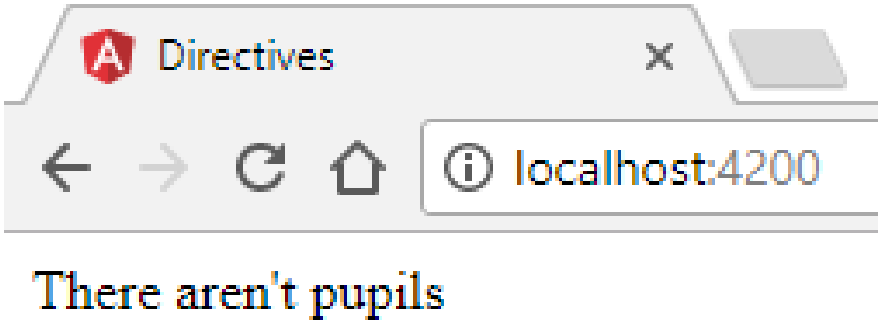
*nglf

app.component.html

```
1  <div *ngIf="pupils.length > 0">
2    There are pupils
3  </div>
4  <div *ngIf="pupils.length == 0">
5    There aren't pupils
6  </div>
```

app.component.ts

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9  //  pupils = ["Pupil1", "Pupil2"];
10   pupils = [];
11 }
```



HTL LEONDING

Zweites div wird gar nicht gerendert

```
▼<body>
  ▼<app-root _nghost-c0 ng-version="5.1.2">
      <!--bindings={
        "ng-reflect-ng-if": "false"
      }-->
      <!--bindings={
        "ng-reflect-ng-if": "true"
      }-->
••    <div _ngcontent-c0>
        There aren't pupils
      </div> == $0
    </app-root>
    <script type="text/javascript" src="inline.bundle.js"></script>
    <script type="text/javascript" src="polyfills.bundle.js"></script>
    <script type="text/javascript" src="styles.bundle.js"></script>
    <script type="text/javascript" src="vendor.bundle.js"></script>
    <script type="text/javascript" src="main.bundle.js"></script>
  </body>
```

Verwendung des Properties hidden

- Wird im DOM gerendert aber ausgeblendet

app.component.html | app.component.ts

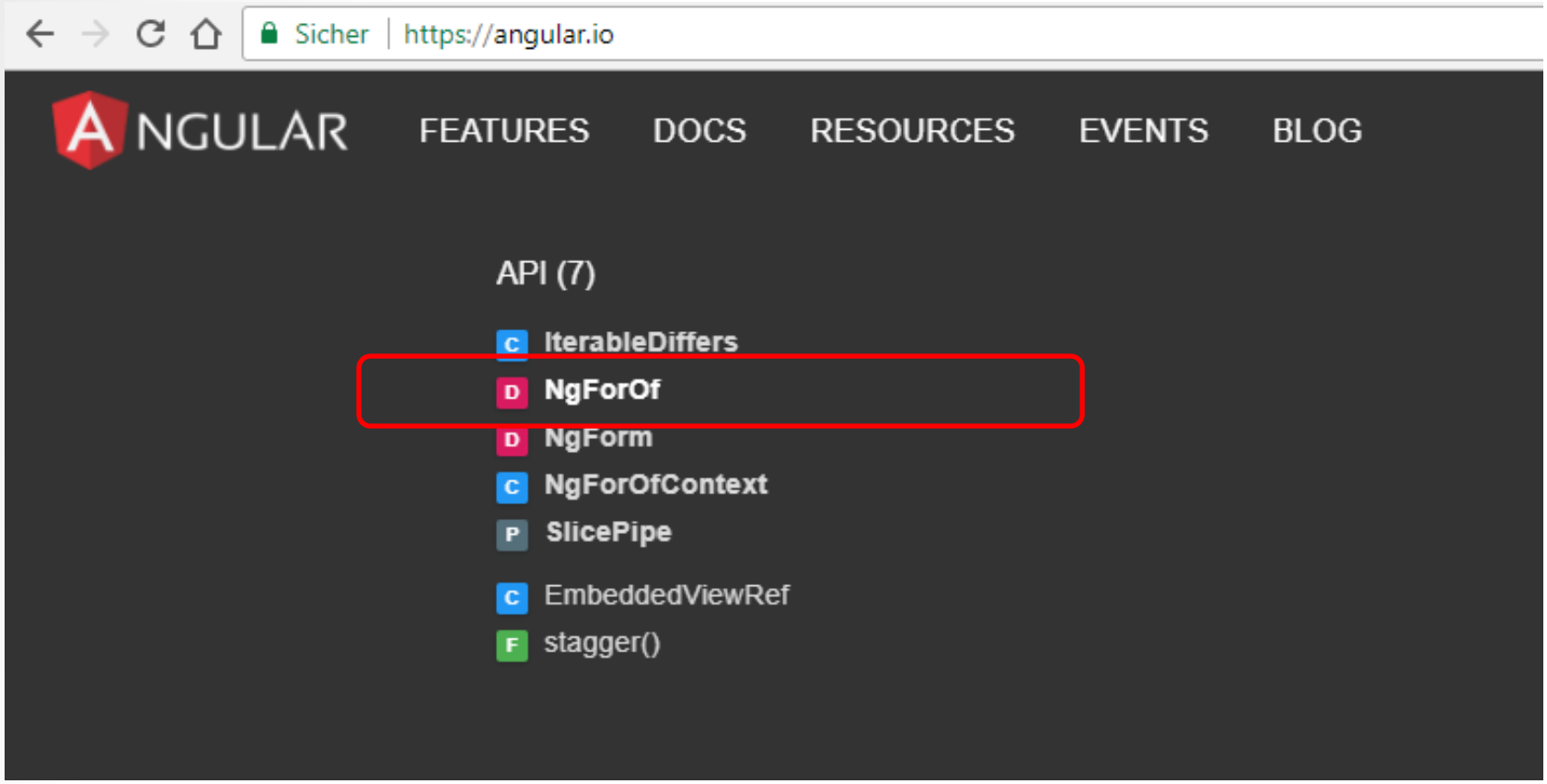
```
1  <div [hidden]="pupils.length == 0">
2    There are pupils
3  </div>
4  <div [hidden]="pupils.length > 0">
5    There aren't pupils
6  </div>
```

```
▼<app-root _nghost-c0 ng-version="5.1.2">
    <div _ngcontent-c0 hidden>
      There are pupils
    </div>
    <div _ngcontent-c0>
      There aren't pupils
    </div> == $0
  </app-root>
```


HTL LEONDING

*ngFor – Erzeugung mehrerer Elemente

- Hilfe unter angular.io ➔ NgForOf



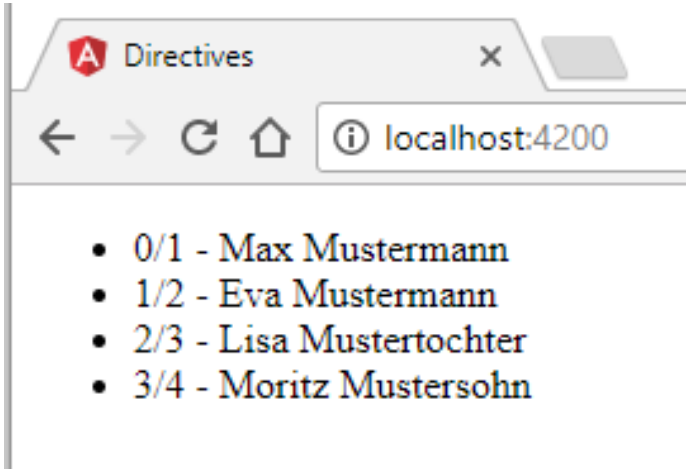
HTL LEONDING

Verwendung exportierter Werte

- Speichern in lokalen Variablen

```
<ul>
  <li *ngFor="let pupil of pupils; index as i">
    {{i}}/{{pupil.id}} - {{pupil.name}}
  </li>
</ul>
```

```
pupils = [
  {id:1, name:'Max Mustermann'},
  {id:2, name:'Eva Mustermann'},
  {id:3, name:'Lisa Mustertochter'},
  {id:4, name:'Moritz Mustersohn'},
];
```



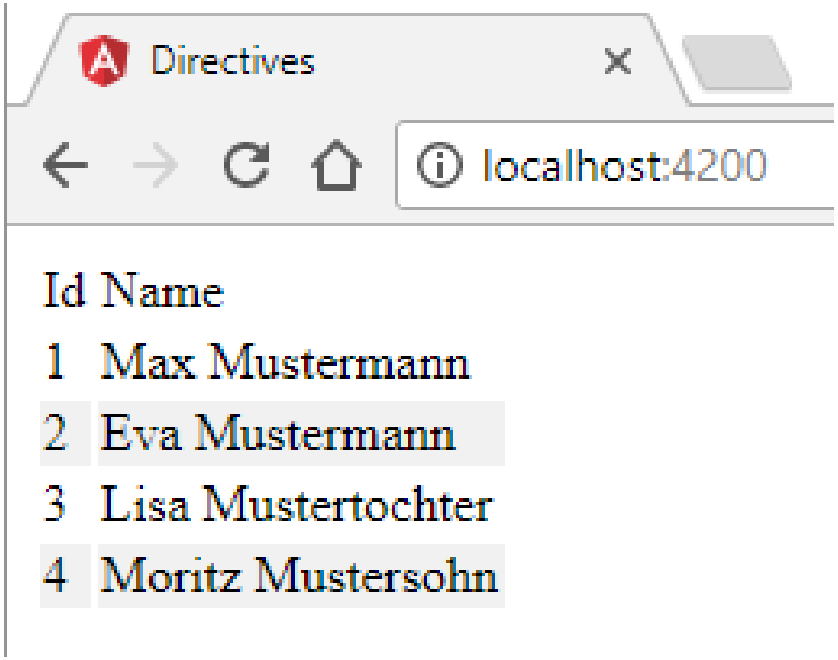
Weitere Werte verwendbar

`NgForOf` provides several exported values that can be aliased to local variables:

- `$implicit: T` : The value of the individual items in the iterable (`ngForOf`).
- `ngForOf: NgIterable<T>` : The value of the iterable expression. Useful when
- `index: number` : The index of the current item in the iterable.
- `first: boolean` : True when the item is the first item in the iterable.
- `last: boolean` : True when the item is the last item in the iterable.
- `even: boolean` : True when the item has an even index in the iterable.
- `odd: boolean` : True when the item has an odd index in the iterable.

Übung *ngFor

- Tabelle der Schüler ausgeben
  - Jede zweite Zeile bekommt hellgrauen Hintergrund

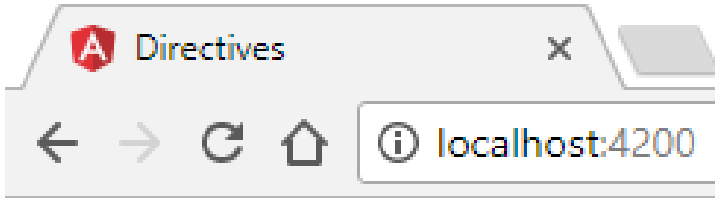


Übung

Besser mit css

app.component.css

```css
table, th , td {
  border: 1px solid grey;
  border-collapse: collapse;
  padding: 5px;
}
table tr:nth-child(odd) {
  background-color: #f1f1f1;
}
table tr:nth-child(even) {
  background-color: #ffffff;
}
```

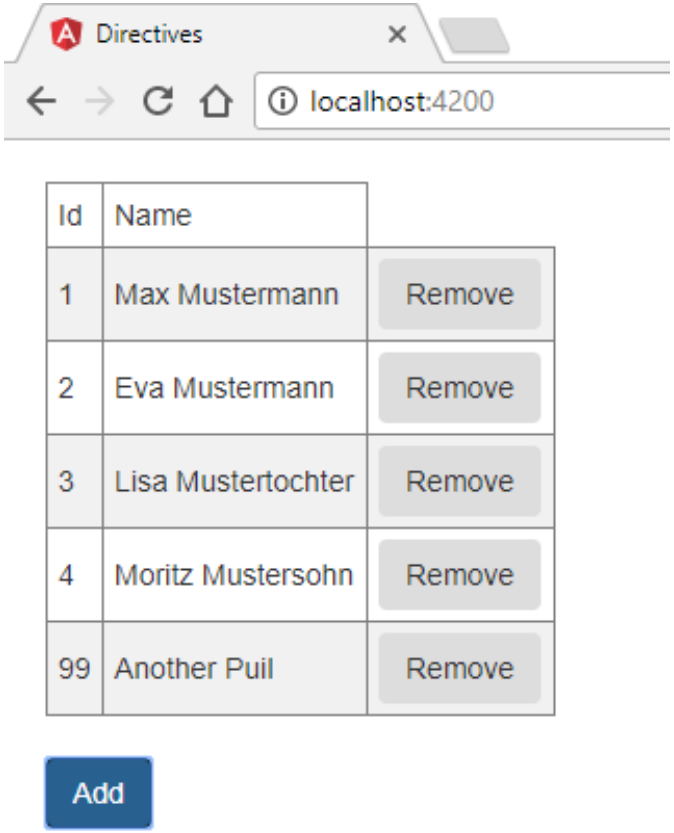


Id	Name
1	Max Mustermann
2	Eva Mustermann
3	Lisa Mustertochter
4	Moritz Mustersohn

HTL LEONDING

Zusatzübung – Add/Remove Pupil

- ChangeTracker überwacht das Binding auf Basis von Objektreferenzen
- Bei Performanceproblemen kann Tracking konfiguriert werden (trackBy)



Übung

HTL LEONDING

Directive ngClass

- Statt classBinding für mehrere Klassen

favorite.component.html

```
1   <span
2     class="glyphicon"
3     [class.glyphicon-star]="isSelected"
4     [class.glyphicon-star-empty]="!isSelected"
5     [ngClass]="{
6       'glyphicon-star': isSelected,
7       'glyphicon-star-empty': !isSelected
8     }"
9     (click)="onClick()"
10    ></span>
```

Directive ngStyle

- Statt mehrerer style-Bindings
- Besser über css stylen

```
<button
    [style.backgroundColor]="canSave ? 'blue': 'gray'"
    [style.color]="canSave ? 'white': 'black'"
    [style.fontWeight]="canSave ? 'bold': 'normal'"
    [ngStyle]="{
        'backgroundColor': canSave ? 'blue': 'gray',
        'color': canSave ? 'white': 'black'
    }"
```

CustomDirective `ng new custom-directive-demo`

- Verhalten eines Elements verändern
- Z.B. Text ➔ UpperCase/LowerCase
- Directive mit CLI anlegen

```
ng g d text-upper-lower
```

Fokusänderungen dokumentieren

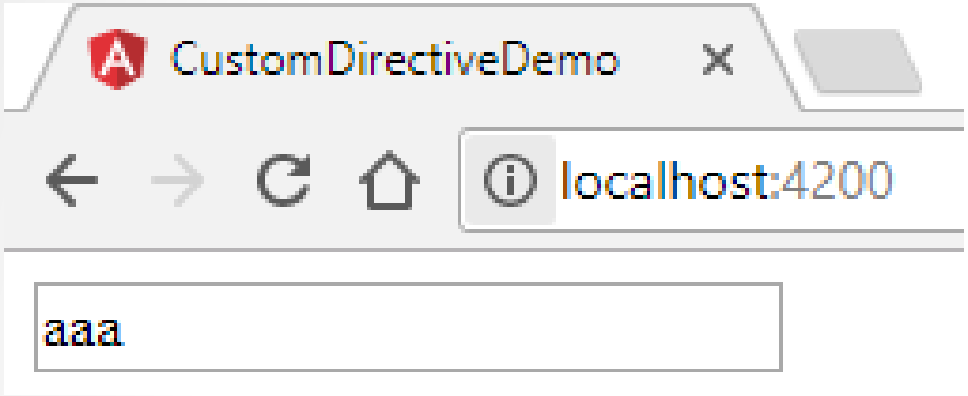
- a

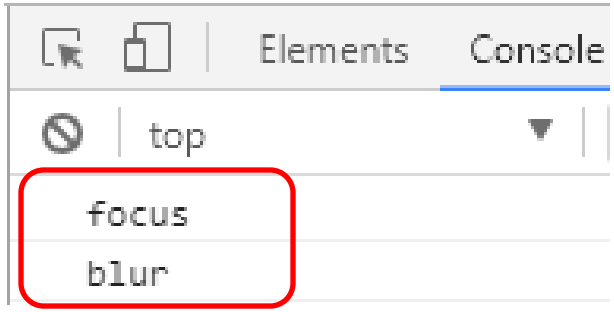
text-upper-lower.directive.ts

```ts
import { Directive, HostListener } from '@angular/core';

@Directive({
  selector: '[appTextUpperLower]'
})
export class TextUpperLowerDirective {
  @HostListener('focus') onFocus(){
    console.log('focus');
  }
  @HostListener('blur') onBlur(){
    console.log('blur');
  }
}
```

app.component.html

```html
<div>
  <input type="text" appTextUpperLower >
</div>
```





HTL LEONDING

Attributiertes Element als Parameter

TS text-upper-lower.directive.ts ●

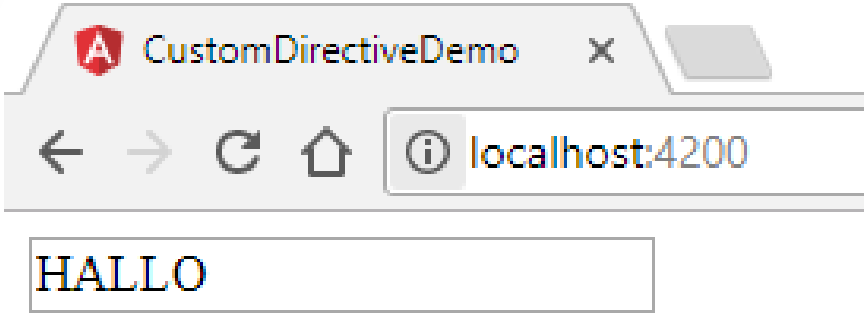
```ts
import { Directive, HostListener, ElementRef } from '@angular/core';

@Directive({
  selector: '[appTextUpperLower]'
})
export class TextUpperLowerDirective {
  constructor(private elementRef : ElementRef){}
  @HostListener('blur') onBlur(){
    console.log(this.elementRef);
  }
}
```

```
▼ ElementRef {nativeElement: input}
  ▼ nativeElement: input
      accept: ""
      accessKey: ""
      align: ""
      alt: ""
      assignedSlot: null
    ▶ attributes: NamedNodeMap {0: _ngconte
      autocapitalize: "sentences"
      autocomplete: ""
      autofocus: false
      baseURI: "http://localhost:4200/"
      checked: false
      childElementCount: 0
```

- ElementRef
  - nativeElement ist das eigentliche InputElement

Text in Großbuchstaben umwandeln

```
@HostListener('blur') onBlur(){
  let text : string =this.elementRef.nativeElement.value;
  this.elementRef.nativeElement.value=text.toLocaleUpperCase();
}
```



HTL LEONDING

Parameter übergeben upper/lower

app.component.html

```
1  <div>
2    <input type="text" appTextUpperLower [format]="'lower'" >
3  </div>
```

```
@Input() format : string;

@HostListener('blur') onBlur(){
  let text : string =this.elementRef.nativeElement.value;
  if(this.format=='upper'){
    this.elementRef.nativeElement.value=text.toLocaleUpperCase();
  }
  else{
    this.elementRef.nativeElement.value=text.toLocaleLowerCase();
  }
}
```

Nur 1 Parameter ➔ Directive als Property

app.component.html

```
1  <div>
2    <input type="text" [appTextUpperLower] ="'lower'" >
3  </div>
```

```
@Input() appTextUpperLower : string;

@HostListener('blur') onBlur(){
  let text : string =this.elementRef.nativeElement.value;
  if(this.appTextUpperLower=='upper'){
    this.elementRef.nativeElement.value=text.toLocaleUpperCase();
  }
  else{
    this.elementRef.nativeElement.value=text.toLocaleLowerCase();
  }
}
```