ANGULAR BASICS

CLI, project structure, development environment, data binding

ANGULAR INSTALLATION

- Prerequisite: Node LTS / Yarn
- Installation: `npm install -g @angular/cli`
- Check version: `ng version`
- In case, set `Path` environment variable: `set Path=%Path%;%appdata%\npm`

CREATE NEW ANGULAR APPLICATION

```
ng new my-angular-app
```

Option `--skip-install` can be used to bypass installation of Node modules.

Skip routing for now!

Node modules can be restored using `npm install` within the project folder.

RUN ANGULAR APPLICATION

Within the project folder, run `ng server` (oder `npm start`)

Tip: Use command prompt instead of PowerShell. Withing a PowerShell, open command prompt by typing `cmd` + return key.

Angular app runs in watch mode on `http://localhost:4200`.

VISUAL STUDIO CODE

We are using Visual Studio Code for Angular development (also for PLFs).

Recommended extensions:

- Angular Language Service
- Angular Snippets
- REST Client

PROJECT STRUCTURE

- `index.html` … entry point
- `main.ts` … main module
- app folder … main component
- `package.json` … Node module dependencies
- `.gitignore` … skip files from version control

ANGULAR TEMPLATE & DATA BINDING

- **Important:** Import `FormsModule`

app-module.ts:

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

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

@NgModule({
   declarations: [AppComponent],
   imports: [BrowserModule, FormsModule],
   providers: [],
   bootstrap: [AppComponent],
})
export class AppModule {}
```

ONE-WAY INTERPOLATION

app-component.ts:

```
export class AppComponent {
   title = 'My Angular App';
}
```

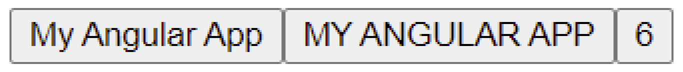
app-component.html:

```
<h1>{{ title }}</h1>
<img src="anything.png" alt="{{ title }}" />
```

ONE-WAY PROPERTY BINDING

app-component.html:

```
<input type="button" [value]="title" />
<input type="button" [value]="title.toUpperCase()" />
<input type="button" [value]="1 + 2 + 3" />
```

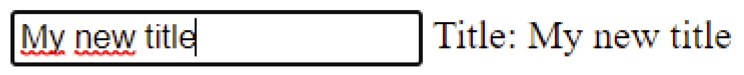


TWO-WAY DATA BINDING

[( )] ... "banana in a box"

app-component.html:

```
<input type="text" [(ngModel)]="title" /> Title: {{title}}
```


My new title Title: My new title

ONE-WAY EVENT BINDING

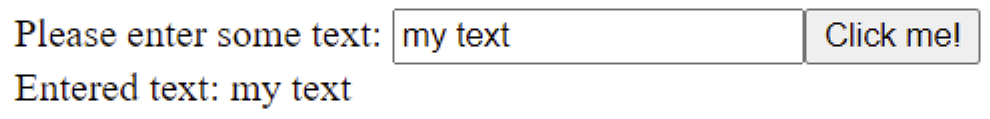
```
export class AppComponent {
  title = 'My Angular App';
  text = '';
  enteredText = '';

  onClick(value: any): void {
    this.enteredText = value;
  }
}
```

app-component.html:

```
Please enter some text:
<input #reference type="text" />
<button (click)="onClick(reference.value)">Click me!</button>

<div>Entered text: {{ enteredText }}</div>
```


Please enter some text: my text Click me!
Entered text: my text

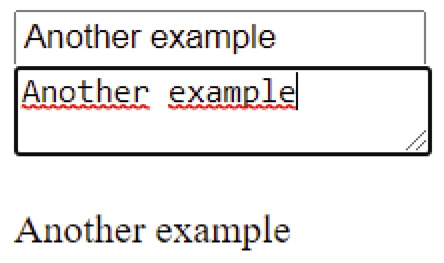
text / textarea

app-component.ts:

```
export class AppComponent {
   title = 'My Angular App';
   text = '';
}
```

app-component.html:

```
<div><input type="text" [(ngModel)]="text" /></div>
<div><textarea [(ngModel)]="text"></textarea></div>
<p>{{ text }}</p>
```



Another example

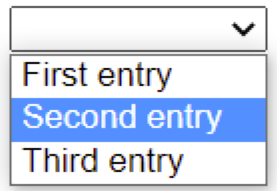
Another example

Another example

`select` (COMBOBOX)

app-component.html:

```
<select [(ngModel)]="text">
   <option value="First entry">First entry</option>
   <option value="Second entry">Second entry</option>
   <option value="Third entry">Third entry</option>
</select>
```



ngFor (REPEATER DIRECTIVE)

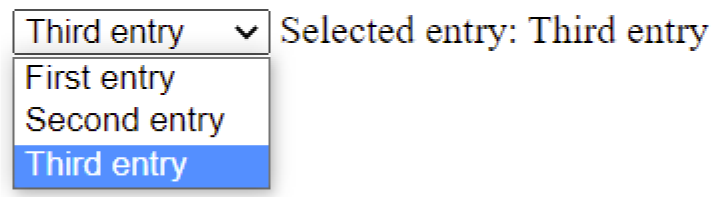
app-component.ts:

```
export class AppComponent {
   title = 'My Angular App';
   text = '';
   entries = ['First entry', 'Second entry', 'Third entry'];
}
```

app-component.html:

```
<select [(ngModel)]="text">
   <option *ngFor="let entry of entries" [value]="entry">{{ entry }}</option>
</select>

Selected entry: {{ text }}
```



Selected entry: Third entry

ngIf (CONDITIONAL DISPLAY) + TEMPLATE VARIABLES

app-component.ts:

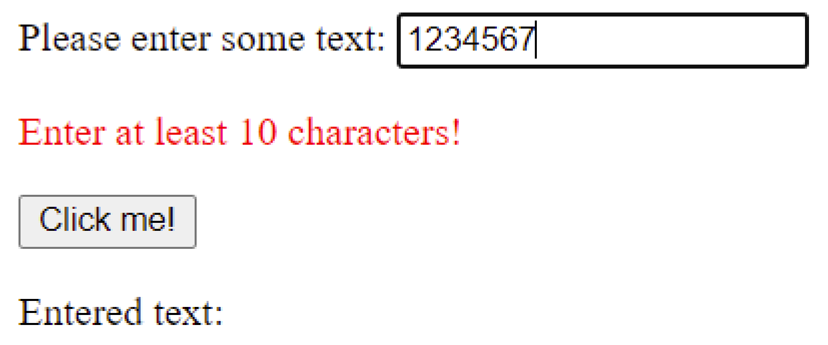
```
export class AppComponent {
  title = 'My Angular App';
  text = '';
  enteredText = '';

  onClick(value: any): void {
    this.enteredText = value;
  }
}
```

app-component.html:

```
Please enter some text:
<input #reference type="text" [(ngModel)]="text" />
<p *ngIf="text.length < 10" style="color: red">Enter at least 10 characters!</p>
<button (click)="onClick(reference.value)">Click me!</button>

<p>Entered text: {{ enteredText }}</p>
```


Please enter some text: 1234567

Enter at least 10 characters!

Entered text:

[hidden]

app-component.ts:

```
export class AppComponent {
  title = 'My Angular App';
  text = '';
  enteredText = '';

  onClick(value: any): void {
    this.enteredText = value;
  }

  isInputValid(): boolean {
    return this.text.length >= 10;
  }
}
```

app-component.html:

```
Please enter some text:
<input #reference type="text" [(ngModel)]="text" />
<p [hidden]="isInputValid()" style="color: red">
  Enter at least 10 characters!
</p>
<button (click)="onClick(reference.value)">Click me!</button>

<p>Entered text: {{ enteredText }}</p>
```

ngValue

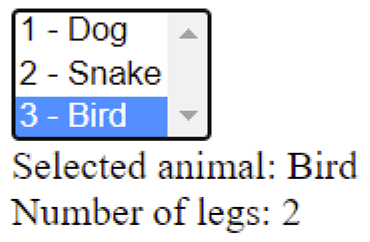
app-component.ts:

```
Animal { name: string; legs: number; } ... export class
AppComponent { title = 'My Angular App'; text = ''; entries = ['First entry',
'Second entry', 'Third entry']; animals: Animal[] = [ { name: 'Dog', legs: 4 },
{ name: 'Snake', legs: 0 }, { name: 'Bird', legs: 2 }, ]; selectedAnimal: Animal
= this.animals[0]; }
```

app-component.html:

```
<select [(ngModel)]="selectedAnimal" size="3">
   <option *ngFor="let animal of animals; let index = index" [ngValue]="animal">
      {{ index + 1 }} - {{ animal.name }}
   </option>
</select>

<div>Selected animal: {{ selectedAnimal.name }}</div>
<div>Number of legs: {{ selectedAnimal.legs }}</div>
```


```
1 - Dog
2 - Snake
3 - Bird
Selected animal: Bird
Number of legs: 2
```

CLASSES & STYLES

Syntax	Description
`[class]="errorClass"`	Replacement class binding
`[class.error]="hasError()"`	Toggling class binding
`[style.color]="hasError() ? 'red' : 'green'"`	Style binding

`[ngClass]`

- Define multiple CSS classes at once:
  - *Variant 1:* Using a JSON Map containing the classes and a condition (`true` or `false`).

app-component.css:

```css
.highlight { background-color: yellow; }
.mark { font-weight: bold; color: red; }
```

app-component.ts:

```typescript
doHighlight(text: string): boolean {
  return text.toLowerCase().indexOf(this.searchText) >= 0;
}
```

app-component.html:

```html
<div [ngClass]="{ highlight:doHighlight(), mark:person.age > 40}">{{ person.firstName }}</div>
```

[ngClass] (CONT.)

- Define multiple CSS classes at once:
  - *Variant 2:* Using a comma separated list of classes.

app-component.css:

```
.highlight { background-color: yellow; }
.mark { font-weight: bold; color: red; }
```

app-component.ts:

```
setAgeStyles(): void {
  this.ageCss = [];
  if (this.selectedAge < 25) this.ageCss.push('highlight');
  if (this.selectedAge < 20) this.ageCss.push('mark');
}
```

app-component.html:

```
<div [ngClass]="ageCss">{{ person.firstName }}</div>
```

[ngStyles] (CONT.)

app-component.ts:

```
countryStyle(country: string): void {
  var style: any = {};
  style['font-style'] = 'italic';
  if (country === 'Austria') style.color = 'red';
  return style;
}
```

app-component.html:

```
<div [ngStyle]="countryStyle(person.country)">{{ person.firstName }}</div>
```