Maximilian Berghoff

Angular von 0 auf 100

Micro Hackathon

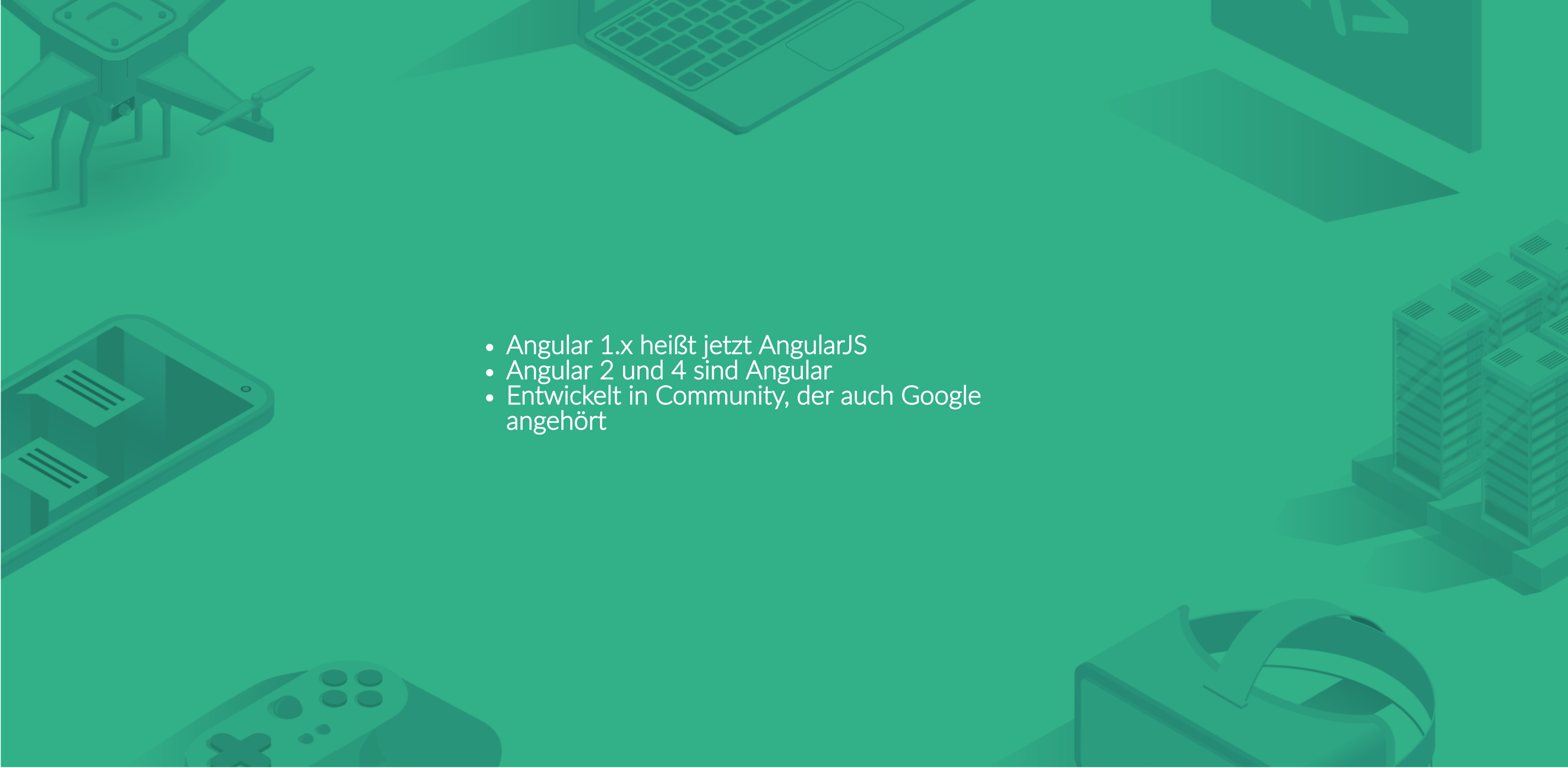
code.talks

- Maximilian Berghoff
- @ElectricMaxxx
- github.com/electricmaxxx
- Maximilian.Berghoff@mayflower.de
- Mayflower GmbH - Würzburg

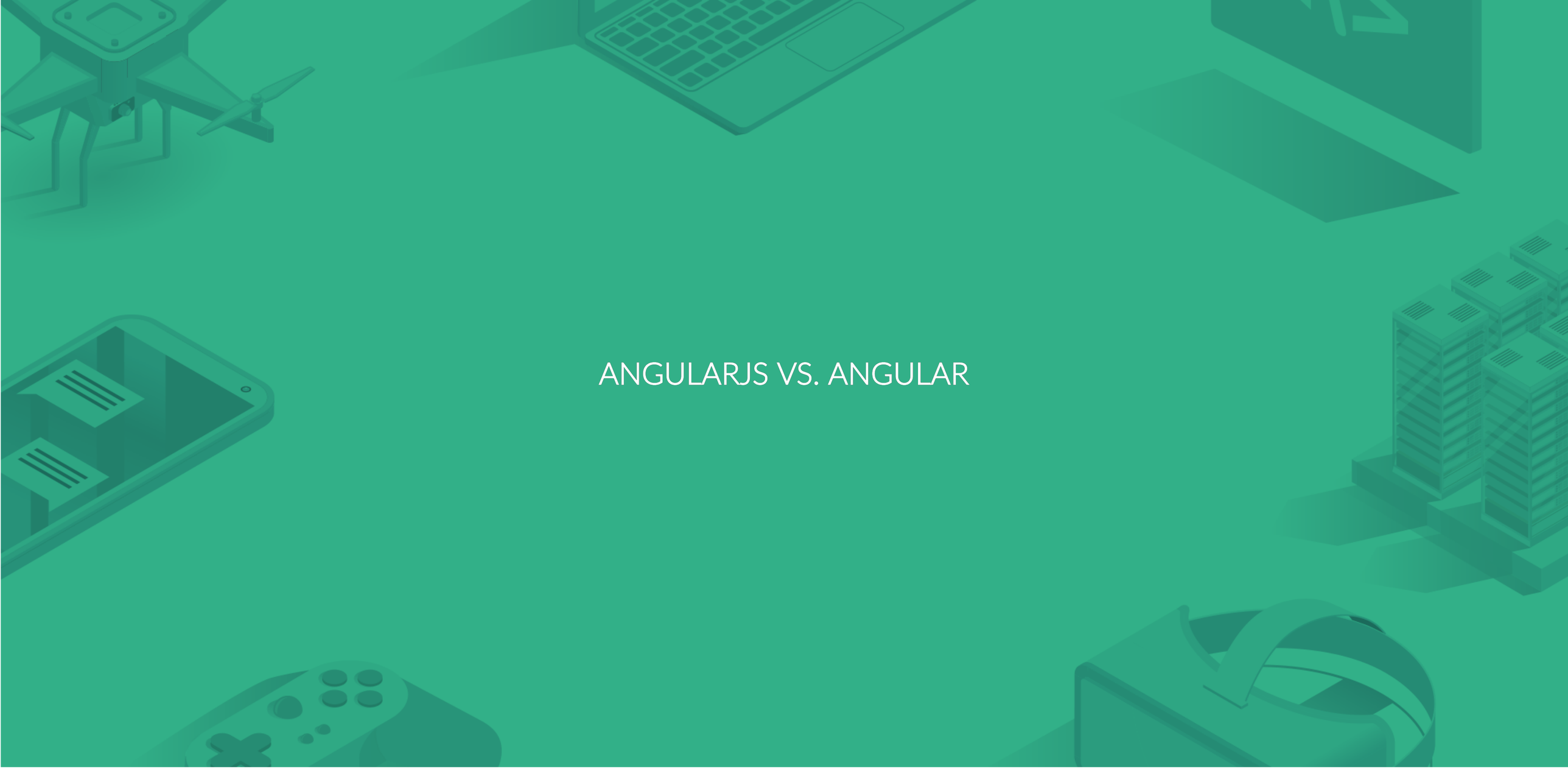
AUSBLICK

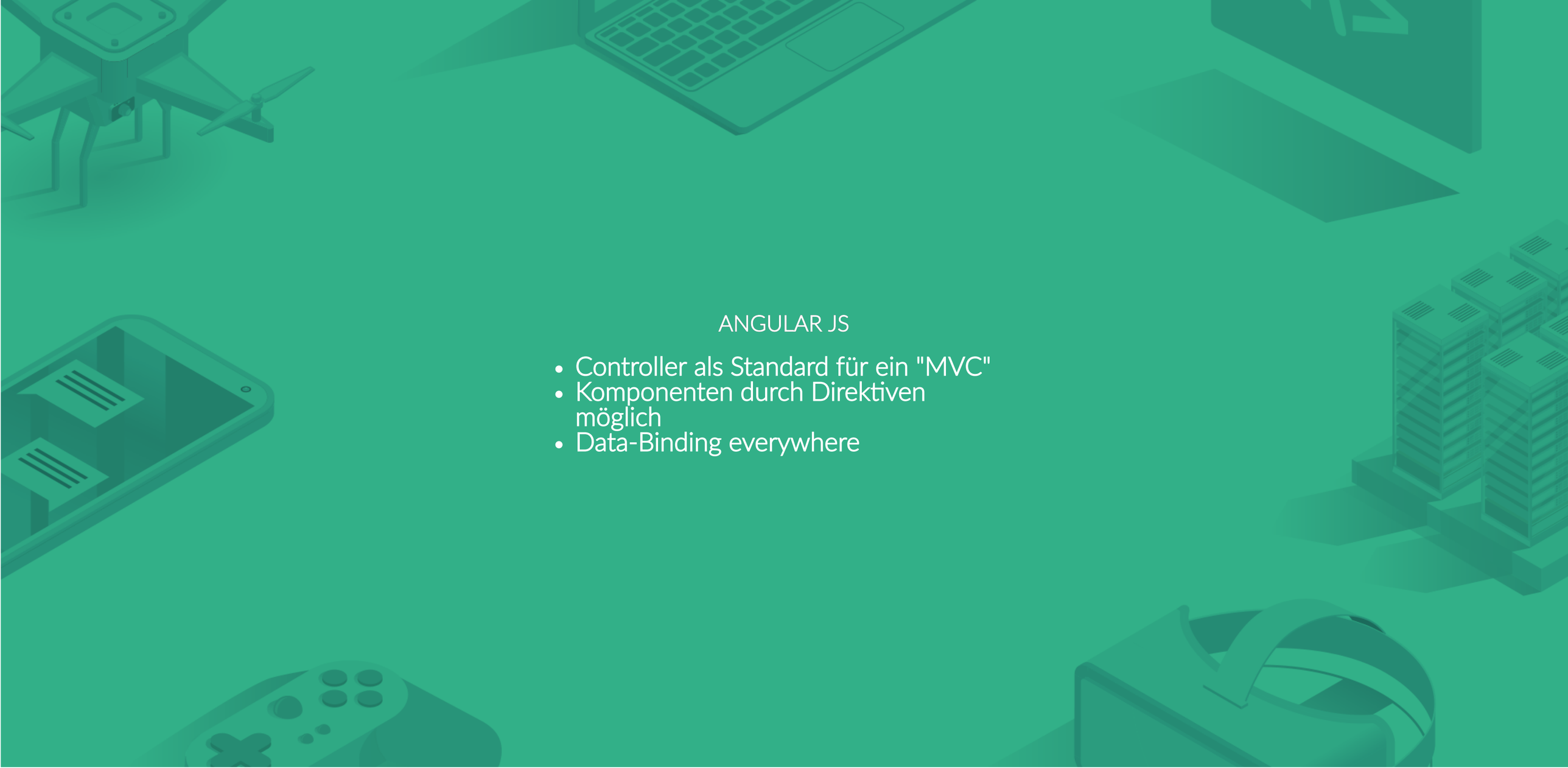
- Basics: Templatating, Komponenten Services
- Advanced: Forms vs. Reactive Forms

HISTORY



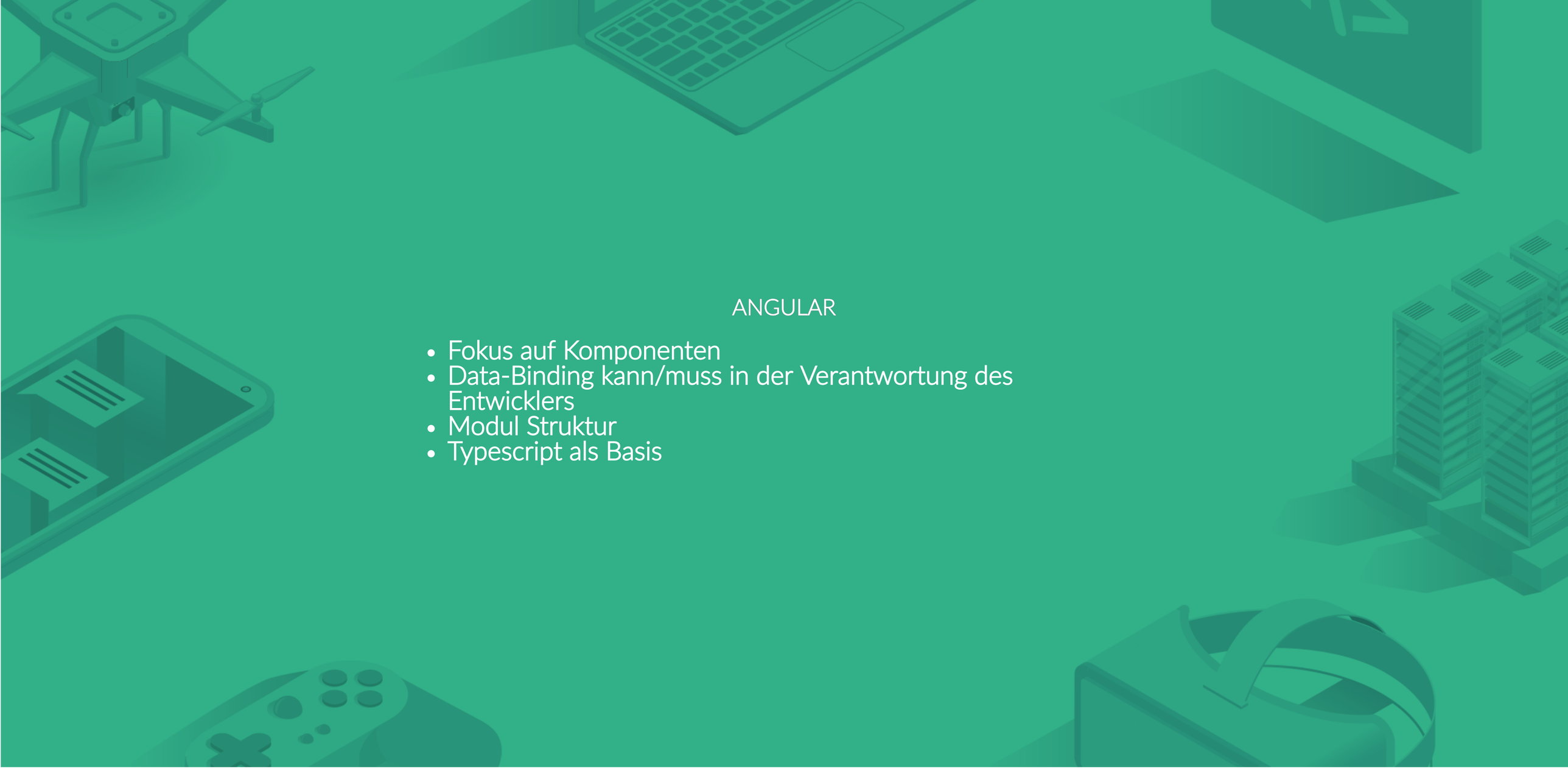
- Angular 1.x heißt jetzt AngularJS
- Angular 2 und 4 sind Angular
- Entwickelt in Community, der auch Google angehört


ANGULARJS VS. ANGULAR



ANGULAR JS

- Controller als Standard für ein "MVC"
- Komponenten durch Direktiven möglich
- Data-Binding everywhere



ANGULAR

- Fokus auf Komponenten
- Data-Binding kann/muss in der Verantwortung des Entwicklers
- Modul Struktur
- Typescript als Basis

- faktisch keine Migration von AngularJS auf Angular möglich (außer Rewrite)
- Angular 2 lässt sich einfach auf 6 migrieren

AUSBLICK - THEMEN

- Bootstraping - Installation einer Skeleton App mit dem CLI
- Basics:
- - Templating, Komponenten, App-Struktur, DI
- Advanced:
- - Forms - Data-Binding oder Reactive
- - Events - Inter-Komponenten-Kommunikation
- - Async - Kommunikation mit einem Server


LOS GEHT'S

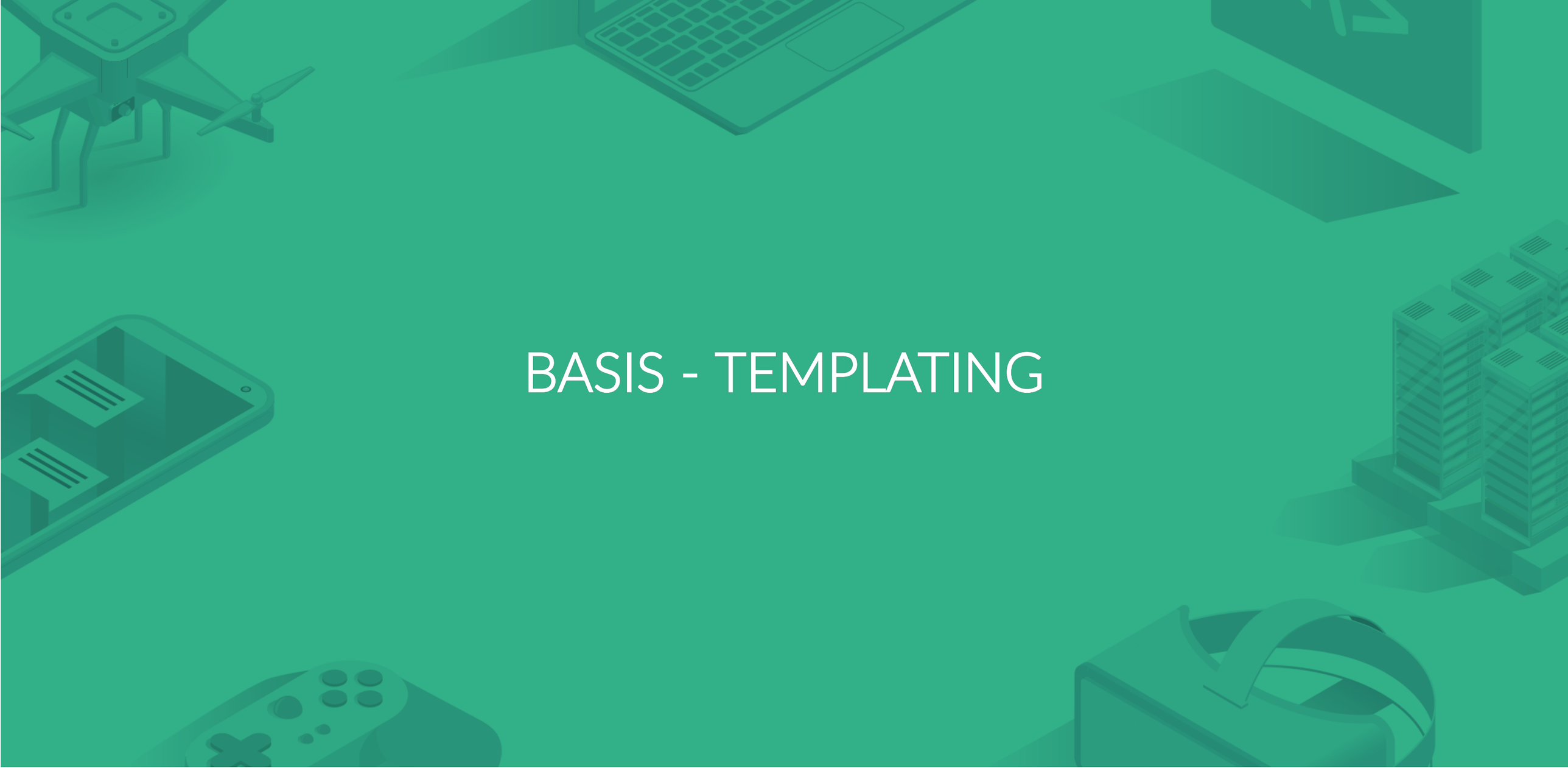
TASK 1: BOOTSTRAPING

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

# Create new skeleton app
ng new example-app
cd example-app

# run it
ng serve -o
```

BASICS - STRUKTUR



BASIS - TEMPLATING

TEMPLATING - INTERPOLATION

```
{{ 'ich bin ein string'}}
<!-- String -->

{{ ichBinEineVariable}}
<!-- public property in component -->

{{ 'ich bin ein string' + ichBinEineVariable}}
<!-- Concatenation -->

{{ gibString() }}
<!-- Method Call -->
```

```
export class AppComponent {
  ichBinEineVariable = 'app';
  gibString(): string {
    return 'Ich bin ein String';
  }
}
```

TEMPLATING - EXPRESSIONS

```
[property]="expression"
```

TEMPLATING - EXPRESSIONS

```
    <span [hidden]="isUnchanged">changed</span>
    <!-- Sichtbarkeit von Span Element -->

    <div *ngFor="let task of tasks">{{task.name}}</div>
    <!-- Angular interne Direktive -->
```

```
export class AppComponent {
  isUnchanged: bool = true;
  tasks: Task[] = [];
}
```

TEMPLATING - STATEMENTS

```
    <button (click)="onClick()">Drück mich</button>
    <!-- Event Handler -->
```

```
export class AppComponent {
  onClick(): void {
    alert('ich wurde gedrückt');
  }
}
```

TEMPLATES - DATA BINDING

Data direction	Syntax	Type
One-Way from data source to view template	{{expression}},[target]="expression"	Interpolation, Property, Attribute, Class, Style
One-way from view target to data source	(target)="statement"	Event
Two-Way	[(target)]="expression"	Two-Way

BASICS - TESTING

BASICS - DEPENDENCY INJECTION

DEPENDENCY INJECTION

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

import { HEROES }     from './mock-heroes';

@Injectable()
export class HeroService {
  getHeroes() { return HEROES; }
}
```

DEPENDENCY INJECTION

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

import { HEROES }     from './mock-heroes';

const heroesPromise = Promise.resolve(HEROES);

@Injectable()
export class HeroService {
  getHeroes(): Promise<Hero[]> { return heroesPromise; }
}
```

DEPENDENCY INJECTION

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

import { HeroService }        from './hero.service';

@Component({
  selector: 'my-heroes',
  providers: [HeroService],
  template: `
  <h2>Heroes</h2>
  <hero-list></hero-list>
  `
})
export class HeroesComponent {
    constructor (private service: HeroService) {}
 }
```

ADVANCED

ADVANCED - EVENTS

EVENTS - INNER COMPONENT

```
import {Component, EventEmitter, Input, Output} from "@angular/core";
@Component({
    selector: 'inner-component',
    template: '<h1>{{title}}</h1>'
})
export class InnerComponent {
    @Input() title: string = '';
}
```

EVENTS -- INNER COMPONENT - SUBMIT EVENT

```
import {Component, EventEmitter, Input, Output} from "@angular/core";
@Component({
    selector: 'inner-component',
    template: `
        <h1>{{title}}</h1>
        <p><button (click)="onClick()">Klick mich</button></p>
        `
})
export class InnerComponent {
    @Input() title: string = '';
    @Output() onClick: EventEmitter<void> = new EventEmitter();
    onClick() {
        this.onClick.emit();
    }
}
```

EVENTS - OUTER COMPONENT

```
import {Component} from "@angular/core";
@Component({
    selector: 'outer-component',
    template: '<inner-component [title]="title"></inner-component>'
})
export class OuterComponent {
    title: string = 'Title';
    onInnerClick(): void {
        alert('Innen wurde gedrückt');
    }
}
```

EVENTS - OUTER COMPONENT - CATCHING

```
import {Component} from "@angular/core";
@Component({
    selector: 'app',
    template: `
        <outer-component (onClick)="onInnerClick()">
        </outer-component>`
})
export class AppComponent {
}
```

ADVANCED - FORMS

FORMS

```
import {Component} from "@angular/core";
@Component({
    selector: 'app',
    template: `
        <form>
            <input
                type="text"
                [(ngModel)]="name"
                name="name"
                id="name" />
        </form>
    `
})
export class AppComponent {
    name: string = 'Max';
}
```

FORMS - REACTIVE FORMS

FORMS - REACTIVE FORMS

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

import { AppComponent }        from './app.component';
@NgModule({
  imports: [
    BrowserModule,
    ReactiveFormsModule
  ],
  declarations: [
    AppComponent
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

FORMS - REACTIVE FORMS - TEMPLATE

```
<h2>Hero Detail</h2>
<h3><i>FormControl in a FormGroup</i></h3>
<form [formGroup]="heroForm" novalidate>
 <div class="form-group">
   <label class="center-block">Name:
     <input class="form-control" formControlName="name">
   </label>
 </div>
</form>
```

FORMS - REACTIVE FORMS - COMPONENT

```
import { Component }             from '@angular/core';
import { FormBuilder, FormGroup } from '@angular/forms';
@Component({
    selector: 'hero-detail',
    templateUrl: 'hero-detail.html'
})
export class HeroDetailComponent {
  heroForm: FormGroup;

  constructor(private fb: FormBuilder) { this.createForm()}

  createForm() {
    this.heroForm = this.fb.group({name: ''});
  }
}
```

FORMS - REACTIVE FORMS

```
<p>Form value: {{ heroForm.value | json }}</p>
<p>Form status: {{ heroForm.status | json }}</p>
```

DIE APP - CFP FOR CODETALKS SPEAKERS

1

As a Speaker i would like to enter my speaker data once, to propose my talk abstracts.

2

As a Speaker i would like enter my talk related data to get a speaker slot at codetalks

3

As a organizer of codetalks i would like to see all proposed talks in a list.

4

As a curator of a codetalks tracks i would like to vote on an abstract, to get it into my track or not

CLONE REPOSITORY

```
# repository and angular cli
$ git clone git@gitlab.com:ElectricMaxxx/code-talks-angular-hack.git
$ cd code-talks-anagular-hack
## use local node stack
$ npm install
$ npm install @angular/cli
$ ng serve -o
## OR use docker-compose
$ docker-compose up -d
```

OR

```
# container usage
$ mkdir app
$ docker run -p 4201:4200 -d -v /absolute/path/to/app/:/usr/src/app/ registry.gitlab.com/electricmaxxx/code-talks-angular-hack:latest
```

QUESTIONS

- Ask Now!
- Twitter: @ElectricMaxxx
- Mail: Maximilian.Berghoff@mayflower.de


THANK YOU!