thecodecampus</>

# Angular & TypeScript

## Grundlagen

TypeScript

# Überblick

**TypeScript**

<> Typen & Grundlagen
Klassen & Interfaces
Dekoratoren
Module </>

**Tooling**

<> Modul Systeme & Module Loader
TypeScript Compiler & Linting
Build-System: Angular CLI </>

**Angular**

<> Komponenten (Grundlagen)
Bootstrapping
Data-Binding & Direktiven
Inputs, Outputs & Events
Template Syntax
Services & Dependency Injection
Pipes
Asynchrone & Reactive Programming
Routing
Formulare & Validierung
Komponenten Interaktion & Datenfluss
Kommunikation mit dem Backend </>

**Testen**

<> Unit-Tests mit Karma und Jasmine </>

# Über uns

**theCodeCampus.de - Weiter.Entwickeln.**

<> Gegründet 2013
Schulungen (seit 2007)
Projekt-Kickoffs
Unterstützung im Projekt </>

**w11k GmbH - the Web Engineers**

<> Gegründet 2000
Entwicklung / Consulting
Web / Java
Esslingen / Siegburg </>

# Ziele der Schulung

<> Grundverständnis für moderne Web-Entwicklung vermitteln

<> Alle wichtigen Angular Konzepte verstehen und anwenden können

<> Schritt für Schritt eine kleine Demo-Anwendung entwickeln

<> In jedem Schritt ein Konzept kennen lernen und dann anwenden

# Vorkenntnisse

<> **EcmaScript 5**
- Variablen, Dynamische Typisierung, Datentypen
- Kontrollstrukturen (if/else, while, for, try/catch, ...)
- Funktionen, Funktionsreferenzen
- Prototyping

<> **HTML**
- Aufbau eines Dokumentes, Tags und Attribute
- Grundlegende Elemente wie `div`, `span`, `ul`, `li`, `table`, `form`, ...

<> **CSS**
- Grundsätzlicher Aufbau & Funktionsweise, Syntax
- Selektoren mit Elementen und Klassen

# Benötigte Software

<> **Moderner Browser mit guten Developer Tools**
- Google Chrome

<> **IDE mit guter HTML und TypeScript Unterstützung**
- JetBrain WebStorm oder IntelliJ Ultimate

<> **Node.JS |** `node`
- Mindestens Version 6 (mit NPM Version >= 3)
- Mac OS: `brew install node`
- Windows: Installer von nodejs.org runter laden

<> **Globale NPM Pakete**
- `npm install -g @angular/cli`

<> **Optional: Versionsverwaltung**
- Für Zwischenstände der Übungsaufgaben
- z.B. Git via GitKraken

# Komponenten - Grundlagen

# Komponenten

<> **Grundlegendes Konzept in Angular**

<> **Gesamtes UI ist aus Baum von Komponenten aufgebaut**
- "Anwendung" ist Top-Level-Komponente
- Komponenten können geschachtelt werden

<> **Komponenten bestehen aus**
- Template / View (HTML)
- Klasse
- Decorator an Klasse

<> **Achtung: Doppelbelegung des Begriffs Komponente**
- In Angular Teil des UI
- Allgemein eine Software-Komponente

# Komponenten schreiben

<> **Komponenten werden im TypeScript Code definiert**

- Annotierte Klasse (Best Practice: PascalCase-Name + Component)
- Annotation Angular spezifisch
- Klasse möglichst Framework unabhängig
- Best Practice: Module mit kebab-case-Name
- Best Practice: Selektor mit kebab-case-Name

```
1  import { Component } from "@angular/core";
2
3  @Component({
4    // css selector to find the component's usage
5    selector: "music-app",
6    template: `
7      <h1>Music App</h1>
8    `
9  })
10 export class MusicAppComponent {}
```

# Komponenten verwenden

<> **Komponenten werden immer im HTML verwendet**

- außer in Tests
- Verwendung muss genau Selektor entsprechen

```
<body>
  <h1>Some static text not managed by Angular</h1>
  <!-- matched by defined selector -->
  <music-app></music-app>
  <!-- not matched by defined selector -->
  <div class="music-app"></div>
  <div id="music-app"></div>
  <div music-app></div>
</body>
```

# Komponenten mit Data-Binding

<> **Template nicht statisch sondern dynamisch**

<> **Komponenten-Klasse stellt Daten bereit**

- Felder / Instanz Variablen

<> **Template zeigt Daten an**

- Interpolation mit `{{ expression }}`
- Gibt Ergebnis der Expression als String aus
- Zugriff nur auf Instanz-Variablen der eigenen Klasse
- Template-Variablen

# Komponenten mit Data-Binding

```
1  import { Track } from "./../tracks/track.model";
2  @Component({
3    selector: "music-app",
4    template: `
5      <h1>Music App</h1>
6      Music App has {{tracks.length}} tracks to play
7    `
8  })
9  export class MusicAppComponent {
10   public tracks: Track[];
11   constructor() {
12     this.tracks = [
13       new Track("Sound of Silence"), new Track("I Wish"),
14       new Track("Over My Shoulder"), new Track("Take Five")
15     ];
16   }
17 }
```

# Kommunikation mit dem Backend

# Kommunikation mit dem Backend

<> Saubere Trennung Frontend und Backend

<> Backend stellt API für Daten: REST, WebSockets, ...

<> Frei in Technologie-Wahl für Server

<> Kommunikation läuft (meist) über HTTP

<> Daten sind (meist) JSON

# Demo Backend

## <> Installation

- Zip-Datei *server-js.zip* entpacken
- Konsole öffnen und in Server Verzeichnis wechseln
- Installieren per `npm install`
- Starten über `node server.js`
- Backend läuft unter `localhost:3000`
- `dev.proxy.json`: `{ "/api": { "target": "http://localhost:3000" } }`
- Dev-Server mit Proxy starten (CORS) `ng serve --proxy dev.proxy.json`

## <> API

- `GET api/training` gibt alles Trainings zurück
- `POST api/training` fügt ein neues Training zur Collection hinzu
- `GET api/training/:id` gibt ein einzelnes Training aus der Collection zurück
- `PUT api/training/:id` aktualisiert ein Training innerhalb der Collection
- `DELETE api/training/:id` löscht ein Training aus der Collection

# Http Service

<> Methoden für Http-Verben: get, put, post, delete, ...

<> Gibt Observable statt Promise zurück

```
getAll(): Observable<Release[]> {
  return this.http.get('api/release')
                  .map(this.extractData)
                  .catch(this.handleError);
}

private extractData(res: Response) {
  // fetch API
  let body = res.json();
  // security: wrapped to object with data property
  return body.data;
}
```

# Http Service

<> `Http` **stark konfigurierbar**

- Alles was HTTP-Aufruf hergibt
- Body, Headers, Options
- `BaseRequestOptions` für Default-Werte

```
add(release: Release): Observable<Release> {
    let body = JSON.stringify(release);
    let headers = new Headers({ 'Content-Type': 'application/json' });
    let options = new RequestOptions({ headers: headers });

    return this.http.post('api/release', body, options)
                    .map(this.extractData)
                    .catch(this.handleError);
  }
```

info@thecodecampus.de
@thecodecampus

www.w11k.de
www.thecodecampus.de

# Copyright