

Motivation

Ausgangsbasis

- SPA nur mit lesenden Back-End-Anfragen
- Ausschließlich dynamische Ausgabe von Daten

Wunsch

- Dynamisch neue Ressourcen erstellen
- Daten bearbeiten und neu erfassen
- Vermeiden, inkorrekte Daten zum Server senden

Lernziele

Formulare

- Sie benennen zwei Möglichkeiten für Erstellung von Formularen
- Sie erklären die Bestandteile des Form Moduls von Angular
- Sie beschreiben das Konzept von Two-Way-Binding

Validierung

- Sie beschreiben, wie Eingabeformulare in Angular validiert werden.
- Sie benennen Bestandteile von Angular, welche für Validierung wichtig sind
- Sie erklären, wie der Zustand eines Formulars geprüft werden kann

Umsetzung von Formularen

- Zwei Möglichkeiten Formulare zu implementieren
 - **Template-driven**, traditionelle Methode mittels Eventlisteners und callbacks
 - **Reactive**, modernere Methode basierend auf Eventstreams, Subjects, Observables und Operatorfunktionen (Rx.js)

Umsetzung von Formularen

■ Reactive

- Fördert Funktionales Programmieren
- Macht Datenverarbeitungsprozess gut sichtbar ("stream processing")
- Einheitliche, zentrale Fehlerbehandlung
- Kontrollobjekte in Komponenten-Klasse gebunden an HTML-Elemente
- Einfachere Unit-Tests

■ Template Driven Forms

- Insgesamt einfacher (und mehr vertraut)
- Formular vollständig im HTML-Template umgesetzt
- Vergleichbar mit AngularJS (Angular 1)
- Validierungsregeln im Template

Einbinden von Template-Driven Forms

- Im zugehörigen Module (hier app.module.ts) das Forms-Module importieren

```
import { FormsModule } from '@angular/forms';
```

- Import im NgModule-Dekorator ergänzen

```
@NgModule({  
  ...  
  imports: [FormsModule, ...],  
  ...  
})  
export class AppModule { }
```

NgModel

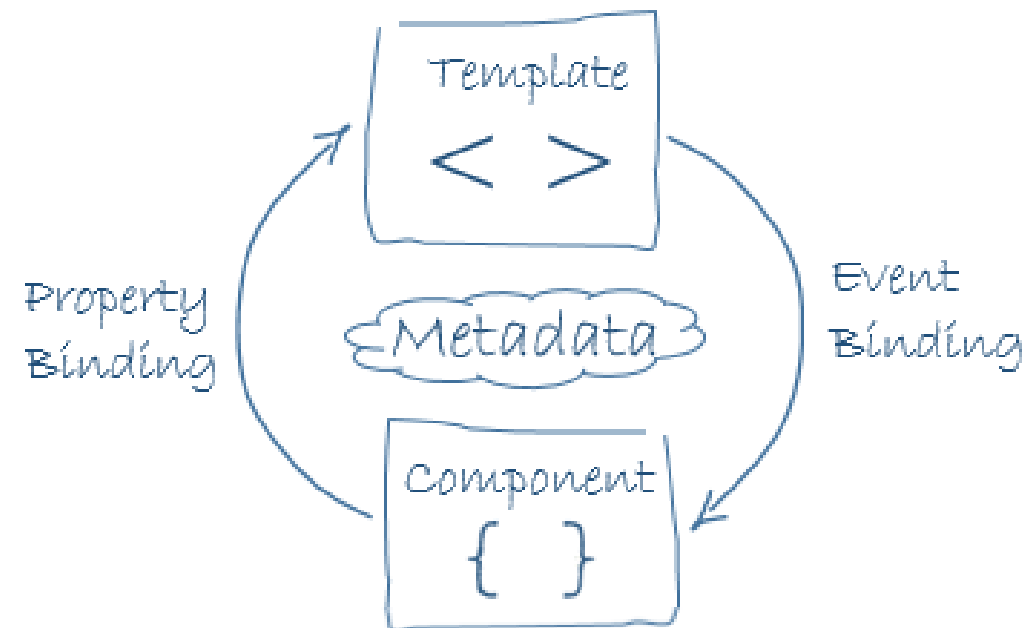
- Mehrere Aufgaben
 - Überwacht den Wert und seine Änderungen
 - Verfolgt Benutzerinteraktionen mit Eingabefeld
 - Pflegt Informationen über Validierungsstatus
 - Synchronisiert Aus- und Eingabe im Template mit Variablenwert aus Komponentenklasse
- **Zusätzlich:** Automatische Anmeldung im Formular als Eingabe
- Bestandteile: ngModel-Direktive und HTML-Attribut „name“

```
<input type="text" id="hero-name"  
      [(ngModel)]="model.name"  
      name="name">
```

Exkurs – Two-Way-Binding

- Syntaktischer Zucker für Property- und Event-Binding
- „Banana into box“-Syntax
- Durch ngModel → Reduzierung von Boiler Plate Code

```
@Component({  
  selector: 'test',  
  template: `  
    <input [(ngModel)]="test"/>`  
})  
export class TestComponent {  
  test = 'nur ein String';  
}
```



NgForm

- Durch Import des Form-Moduls wird die Direktive automatisch hinzugefügt
- Direktiven-Selektor beinhaltet u. a. „form“
- Aggregiert und kontrolliert Eingaben (mit ngModel-Direktive) eines Formulars
 - Werte auslesen
 - Formular validieren

NgForm

Komponenten-Klasse

```
export class CreateHeroComponent {  
  public model: Hero = new Hero();  
  // Initialisierungscode etc.  
  
  onSubmit() { }  
}
```

Template

```
<form (ngSubmit)="onSubmit()">  
  <div>  
    <label for="hero-name">  
      Name  
    </label>  
    <input type="text" id="hero-name"  
      [(ngModel)]="model.name"  
      name="name">  
  </div>  
  <!-- Andere Eingaben -->  
  <button type="submit">  
    Save  
  </button>  
</form>
```

ngSubmit

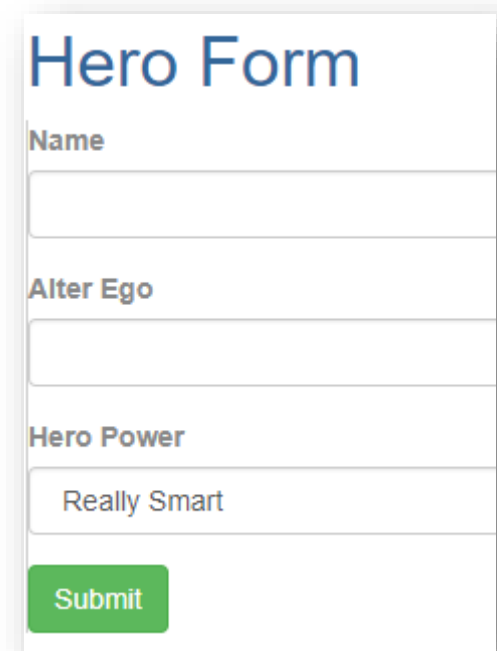
Formular wird per Klick auf Submit-Schaltfläche abgesendet
(`type="submit"`)

- Binden Sie Event-Property *onSubmit* an die Komponentenfunktion

```
<form (ngSubmit)="onSubmit()">
```

- Zugehörige Methode in Komponentenklasse

```
export class HeroFormComponent {  
  ...  
  submitted = false;  
  ...  
  onSubmit() { this.submitted = true; }  
}
```



The image shows a web form titled "Hero Form". It contains three input fields: "Name", "Alter Ego", and "Hero Power". The "Hero Power" field has the text "Really Smart" entered. Below the input fields is a green "Submit" button.

Validierung

Phone number

Phone number

Phone number is required.

Template Driven Form – Validierung

- Definition von Validierungsregeln im Template
- Erfolgt über Direktiven aus Forms-Module
 - required – Erforderliche Eingabe
 - minlength – Mindestlänge der Eingabe
 - pattern – Eingabe muss einen definierten Regex erfüllen

Template Driven Form – Validierung

■ Beispiel:

```
<form (ngSubmit)="onSubmit()">
  <div>
    <label for="phone-number">Phone number</label>
    <input type="text" id="phone-number"
      [(ngModel)]="model.phoneNumber"
      name="phoneNumber"
      required minlength="5">
  </div>
  <!-- Andere Eingaben -->
  <button type="submit">Save</button>
</form>
```

Zustände eines NgModels

- Drei Zustände möglich (Wichtig für Validierung)

Zustand	CSS-Klasse, wenn „true“	CSS-Klasse, wenn „false“
Eingabefeld besucht (durch einen Klick)	ng-touched	ng-untouched
Wert im Eingabefeld verändert	ng-dirty	ng-pristine
Validierungsregeln sind erfüllt	ng-valid	ng-invalid

Zustände eines NgModels

■ Beispiel

<form>

<input type="text" id="phone-number" required

[(ngModel)]="model.phoneNumber" name="phoneNumber" #spy>

Zustände des Eingabefelds: {{spy.className}}

</form>

ng-pristine

ng-valid

ng-touched

ng-dirty

ng-invalid

ng-untouched

Exkurs: Template Reference Variables

- Bestandteil von Angular-Template-Syntax
- Ermöglichen es, eine Reference auf DOM-Elemente in HTML zu definieren

- Beispiel: (#phone entspricht HTMLInputElement)

```
<input #phone value="0123456789">
```

```
<button (click)="callPhone(phone.value)">Call</button>
```

- Direktiven erweitern häufig die Funktionalität von HTML-Elementen beispielsweise:

```
<form #heroForm="ngForm" class="form-group">
```

...

```
<button type="submit" class="btn btn-success,,  
    [disabled]="!heroForm.form.valid">Submit</button>
```

```
</form>
```


Validierung von Eingaben in HTML

- Über Ergebnis der Validierung benachrichtigen

```
<label for="phoneNumber">Phone number</label>
<input type="text" id="phoneNumber" [(ngModel)]="model.phoneNumber"
      name="phoneNumber" required #phoneNumber="ngModel">
<div *ngIf="phoneNumber.invalid &&
      (phoneNumber.dirty || phoneNumber.touched)">
<div *ngIf="phoneNumber.errors.required">Phone number required</div>
</div>
```

Phone number

Phone number required

- Definierte Template Reference Variable: `#phoneNumber="ngModel"`

Zusammenfassung - Formulare

Um was ging es in diesem Modul?

- Bestandteile des form-Modules
- Two-Way-Binding in Angular

Wozu brauche ich das? Was will ich damit machen?

- Nutzung von üblichen HTML-Bausteinen, um mit Hilfe von Angular Daten zu erfassen
- SPA-Zustand von Änderungen bei Werten im Template und Komponentenklassen verfolgen

Zusammenfassung – Validierung

Um was ging es in diesem Modul?

- Validierung von Formularen in Angular
- Syntax und Nutzen von Template-Reference-Variables

Wozu brauche ich das? Was will ich damit machen?

- Validierung von Benutzereingaben, bevor diese zum Back-End gesendet werden (Usability)
- Benutzer vor falschen Eingaben schützen
- Dynamische Referenzen auf HTML-Elemente erzeugen, um deren Werte auszulesen

Kontrollfragen

Formulare

- Welche Möglichkeiten bietet Angular neben Template-driven Forms, um Formulare zu definieren?
- Kann Two-Way-Binding auch ohne NgModel umgesetzt werden?
- Wofür wird der Location-Header eingesetzt?
Gibt es Alternativen?



Kontrollfragen

Validierung

- In welchen Zuständen kann sich ein NgModel befinden?
- Wie werden die Zustände bei der Validierung eingesetzt?
- Wozu können Template-Reference-Variables eingesetzt werden?

