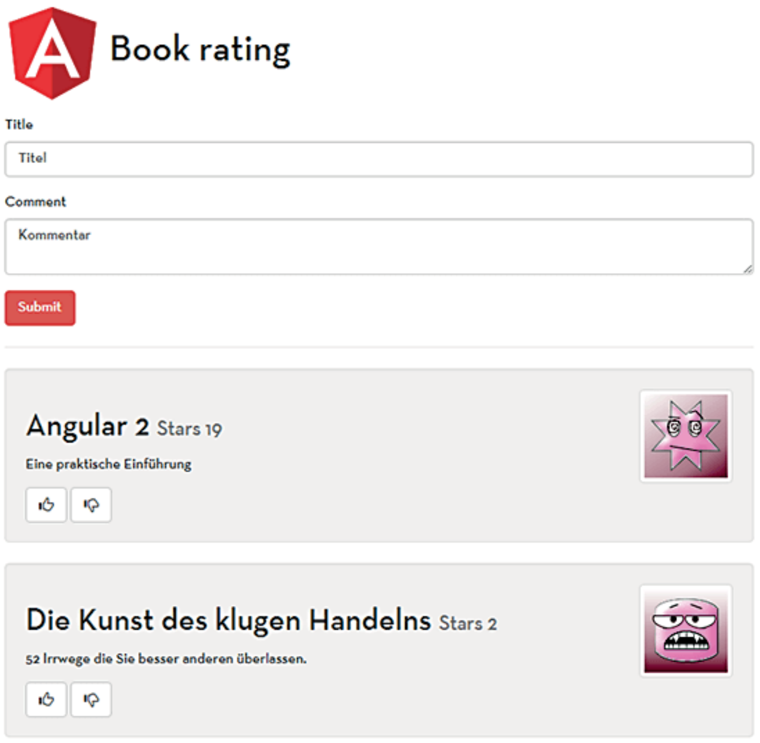
Angular 2.0

Einführung & Schnellstart

Was wir bauen wollen


Book rating

Title

Titel

Comment

Kommentar

Submit

Angular 2 Stars 19

Eine praktische Einführung

Die Kunst des klugen Handelns Stars 2

52 Irrwege die Sie besser anderen überlassen.

TypeScript

Setup

```
$ npm install -g typescript tsd
$ tsd install angular2/ es6-shim
$ tsc --watch
_
```

… oder Atom

Klassen

```
class Book {
  title: string;
  comment: string;
  rating: number = 0;

  constructor(title, comment) {
    this.title = title;
    this.comment = comment;
  }

  rateUp() {
      this.rating++;
  }
}
```

Module

```
import { Book } from 'book'

var book = new Book('Angular 2', 'Bald ist es soweit!');
book.rateUp();
```

Bootstrapping

index.html

SystemJS lädt app

```html
<h1>
  <img src="./img/angular2-logo.svg"> Book rating
</h1>

<book-rating></book-rating>

<script>
  System.config({ packages: { 'app': { }}});
  System.import('./app/app')
</script>
```

BookRating

Komponente soll zunächst nur ein `Array<string>` darstellen

```
// book-rating.ts
import {Component, View} from 'angular2/angular2';

@Component({
  selector: 'book-rating'
})
@View({
  directives: [], // later: BookComponent, NgFor
  template: `
    <h1>Bücher</h1>
    <p>{{ books }}</p>
   `
})
export default class BookRating {
  books: Array<string>;

  constructor() {
    this.books = ['Angular 2', 'Aurelia'];
  }
}
```

Neu: Dekoratoren

Laden der App

bootstrap der ersten Start-Komponente

```
// app.ts
import { bootstrap } from 'angular2/angular2';
import BookRating from './components/book-rating';

bootstrap(BookRating);
```

Components & Views

Bücher-Klasse

TypeScript-Typen verwenden

```
// book.ts
export default class Book {
  title: string;
  comment: string;
  rating: number = 0;

  constructor(title, comment) {
    this.title = title;
    this.comment = comment;
  }
}
```

BookComponent

Komponente soll ein einzelnes Buch anzeigen

```
// book-component.ts
import { Component, View, Input } from 'angular2/angular2';
import Book from '../models/book';

@Component({
  selector: 'book'
})
@View({
  template: `
    <div class="well">
      <div class="thumbnail pull-right">
        <img src="//gravatar.com/avatar/__BEWERTUNG__?s=80&default=wavatar"/>
      </div>
      <h2>__TITEL__ <small>Stars __BEWERTUNG__</small></h2>
      <p>__KOMMENTAR__</p>

    </div>
  `
})
export default class BookComponent {
  @Input() book: Book;
}
```

Template Binding

Interpolation: Daten können via `{{ }}` gebunden werden

```
<!-- book-component.ts -->
<img src="//gravatar.com/avatar/{{ book.rating }}?s=80&default=wavata
<h2>{{ book.title }} <small>Stars {{ book.rating }}</small></h2>
<p>{{ book.comment }}</p>
```

BookRating

Komponente soll zunächst nur ein einziges Buch anzeigen

```
// book-rating.ts
import {Component, View, NgFor} from 'angular2/angular2';
import Book from '../models/book';
import BookComponent from './book-component';

@Component({
  selector: 'book-rating'
})
@View({
  directives: [BookComponent, NgFor],
  template: `
     <h1>Buch</h1>
     <book [book]="book"></book>
   `
})
export default class BookRating {
  //books: Array<Book>;
  book: Book;

  constructor() {
    this.book = new Book('Angular 2', 'Eine praktische Einführung');
  }
}
```

BookRating importiert BookComponent

[Property bindings]

Ziel: inputs der gebundenen Komponente
Property bindings sind durch [] gekennzeichnet

```
<book [book]="book" *ng-for="#book of books"></book>
```

Forms

Formular

BookRating erhält zusätzliche Interaktionselemente

```
// book-rating.ts
@View({
  directives: [BookComponent, NgFor],
  template: `
     <div class="form">
       <div class="form-group">
         <div><label for="title">Title</label></div>
         <div><input class="form-control" name="title" #title></div>
       </div>
       <div class="form-group">
         <div><label for="link">Comment</label></div>
         <div><textarea class="form-control" name="comment" #comment></textarea></div>
       </div>
       <div class="form-group">
        <button (click)="add(title, comment)" class="btn btn-danger">Submit</button>
       </div>
     </div>

     <hr>
     <book *ng-for="#book of books" [book]="book"></book>
   `
})
```

Referenzvariablen

Mit einer **#** kann man eine Referenz initialisieren

```
<div><input name="title" #title></div>
```

Vergleichbar mit `ng-model` aus AngularJS 1.x

(Event Binding)

Event bindings sind durch **()** gekennzeichnet

```
 <button (click)="add(title, link)">Submit</button>
```

Interaktion

Klassen sind die neuen Controller

```
// book-rating.ts
export default class BookRating {
  books: Array<Book>;

  constructor() {
    this.books = [
      new Book('Angular 2', 'Eine praktische Einführung')
    ];
  }

  add(title, comment) {
    var newBook = new Book(title.value, comment.value);
    this.books.push(newBook);

    title.value = '';
    comment.value = '';
  }
```

Data Flow

Interne Ereignisse

Bücher bewerten

```
// book-component.ts
import { Component, View, Input } from 'angular2/angular2';
import Book from '../models/book';

@Component({ /* ... */ })
@View({
  template: `
    <div class="well">
      <!-- ... -->

      <button (click)="rateUp()" class="btn btn-default glyphicon glyphicon-thumbs-up"></button>
      <button (click)="rateDown()" class="btn btn-default glyphicon glyphicon-thumbs-down"></button>
    </div>
  `
})
export default class BookComponent {
  @Input() book: Book;

  rateUp() {
    this.book.rating++;
  }

  rateDown() {
    this.book.rating--;
  }
}
```

Externe Ereignisse

Sender: outputs senden Event aus Komponente heraus

```
// book-component.ts
import { Component, View, EventEmitter } from 'angular2/angular2';
import { Input, Output } from 'angular2/angular2';
import Book from '../models/book';

@Component({
  /* ... */
})
@View({
  /* ... */
})
export default class BookComponent {
  @Input() book: Book;
  @Output() rated: EventEmitter = new EventEmitter();;

  rateUp() {
    this.book.rating++;
    this.rated.next(this.book);
  }

  rateDown() {
    this.book.rating--;
    this.rated.next(this.book);
  }
}
```

Auf Event reagieren

Empfänger: **(event binding)**

```
// book-rating.ts
import {Component, View, NgFor} from 'angular2/angular2';
import Book from '../models/book';
import BookComponent from './book-component';

@Component({
  selector: 'book-rating'
})
@View({
  directives: [BookComponent, NgFor],
  template: `
     <!-- ... -->
     <book ... (rated)="reorderBooks(book)"></book>
   `
})
export default class BookRating {
  books: Array<Book>;

  /* ... */

  reorderBooks(book: Book) {
    this.books.sort((a, b) => b.rating - a.rating);
  }
}
```

Unit Tests mit

KARMA

Jasmine

Tests im BDD-Style

```
describe("A suite", function() {
  it("contains spec with an expectation", function() {
    expect(true).toBe(true);
  });
});
```

angular2/testing

import {} verwenden!

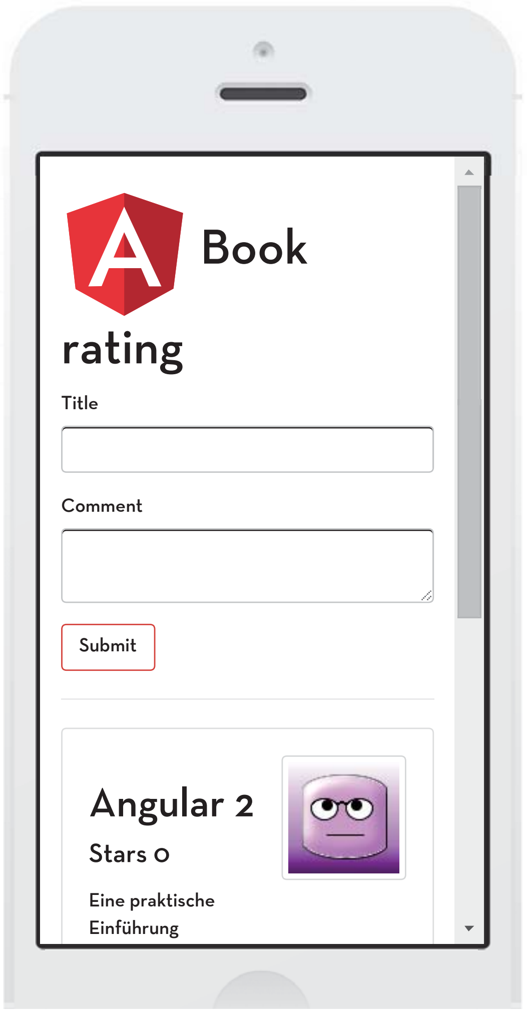
```
import {
    it,
    describe,
    expect,
    inject,
    beforeEachProviders
} from 'angular2/testing';
```

Spec

```
import { it, describe, expect, inject, beforeEachProviders } from 'angular2/testing';
import BookRating from '../../app/components/book-rating';

describe('BookRating component', () => {
  beforeEachProviders(() => [BookRating]);

  it('should have a predefined list of 2 books', inject([BookRating], (bookRating: BookRating) => {
    expect(bookRating.books.length).toBe(2);
  }));
});
```



Wir sind happy

Slides: bit.do/dos-angular2
Demo: bit.do/dos-angular2-demo

Danke, wir freuen uns auf eure Anregungen!