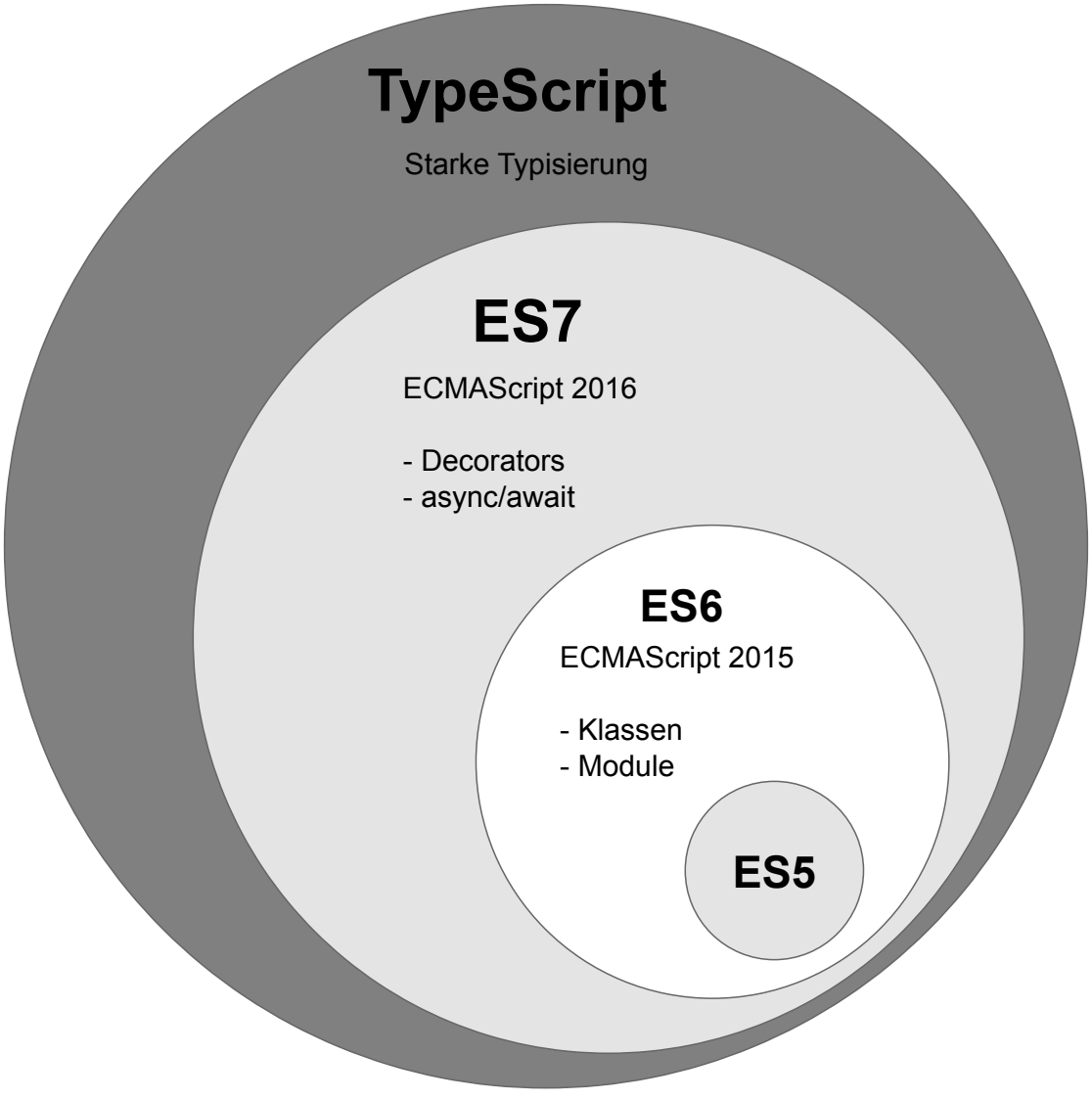
Angular 2.0

Einführung & Schnellstart


TypeScript


TypeScript
Starke Typisierung
ES7
ECMAScript 2016
- Decorators
- async/await
ES6
ECMAScript 2015
- Klassen
- Module
ES5

Setup

```
$ npm install -g typescript@1.8.7 typings@0.7.3
$ typings install es6-shim --ambient --save
$ tsc --watch
```

... oder Angular-CLI

... und/oder Atom mit atom-typescript

Klassen

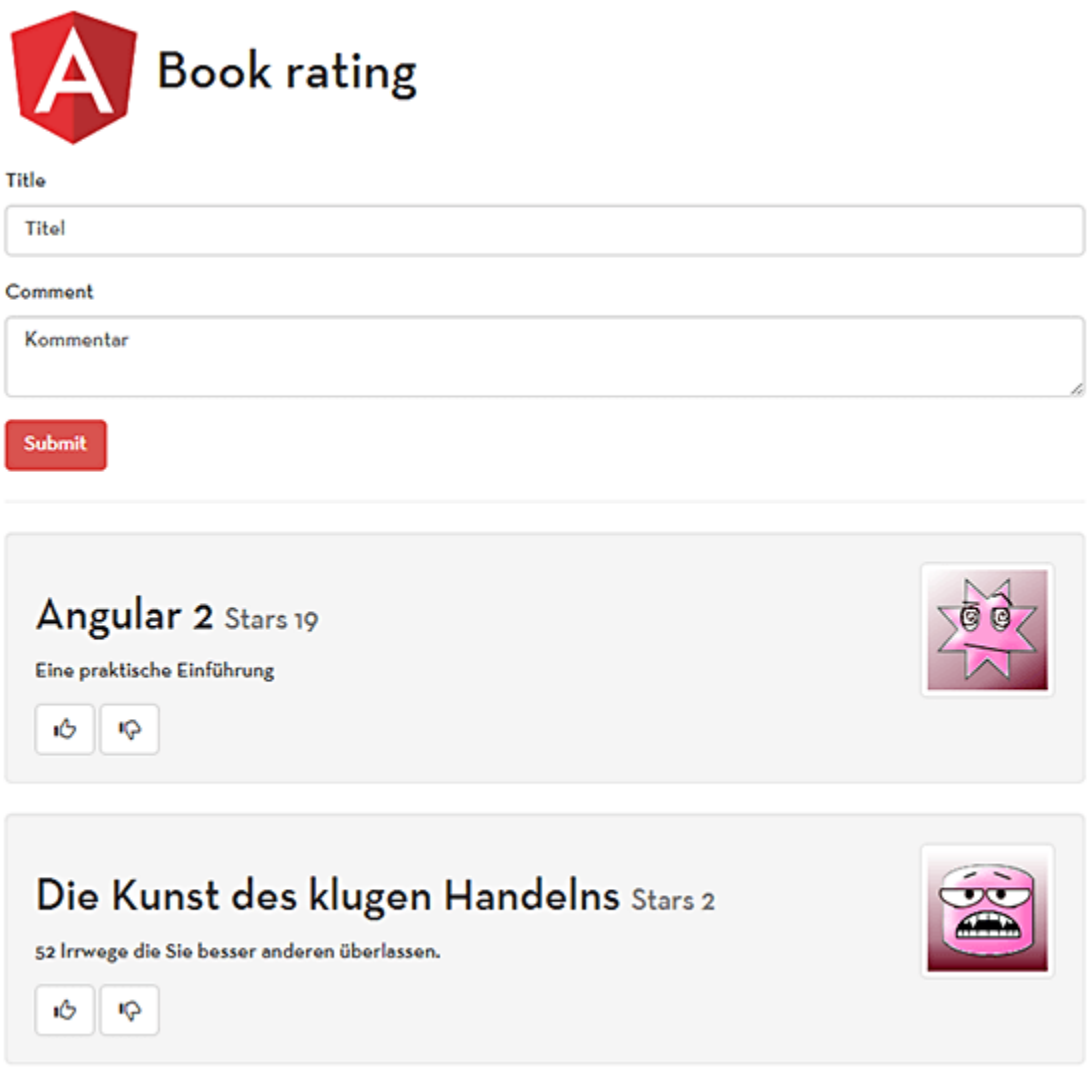
```
class Book {
  title: string;
  comment: string;
  rating: number = 0;

  // TODO: constructor shorthand
  constructor(title: string, comment: string) {
    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();
```

Angular-CLI

Was wir bauen wollen



```
$ npm install -g angular-cli@0.0.24
$ ng help
$ ng new BookRating
$ cd BookRating
$ npm start
```

Bootstrapping

index.html

SystemJS lädt app

```
<book-rating-app>Loading...</book-rating-app>

<script src="vendor/es6-shim/es6-shim.js"></script>
[... + more polyfills]

<script src="vendor/angular2/bundles/angular2.dev.js"></script>
[... + more bundles]

<script>
  System.config({ packages: { app: { }}});
  System.import('app.js').then(null, console.error.bind(console));
</script>
```

Laden der App

bootstrap der Root-Component

```
// app.ts
import {bootstrap} from 'angular2/platform/browser';
import {BookRatingApp} from './app/book-rating';

bootstrap(BookRatingApp);
```

bootstrap() != Bootstrap

Framework Bootstrap hinzufügen

```
$ npm install bootstrap@3.3.6 --save
```

```
//ember-cli-build.js
var Angular2App = require('angular-cli/lib/broccoli/angular2-app');

module.exports = function(defaults) {
  var app = new Angular2App(defaults, {
    vendorNpmFiles: ['bootstrap/dist/**']
  });
  return app.toTree();
}
```

broccoli: asset pipeline anpassen

index.html

mit Bootstrap-Framework

```
<link rel="stylesheet" href="vendor/bootstrap/dist/css/bootstrap.min.

[...]

<body class="container">

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

</body>
```

Dashboard

erste Komponente generieren lassen

```
$ ng generate component Dashboard
```

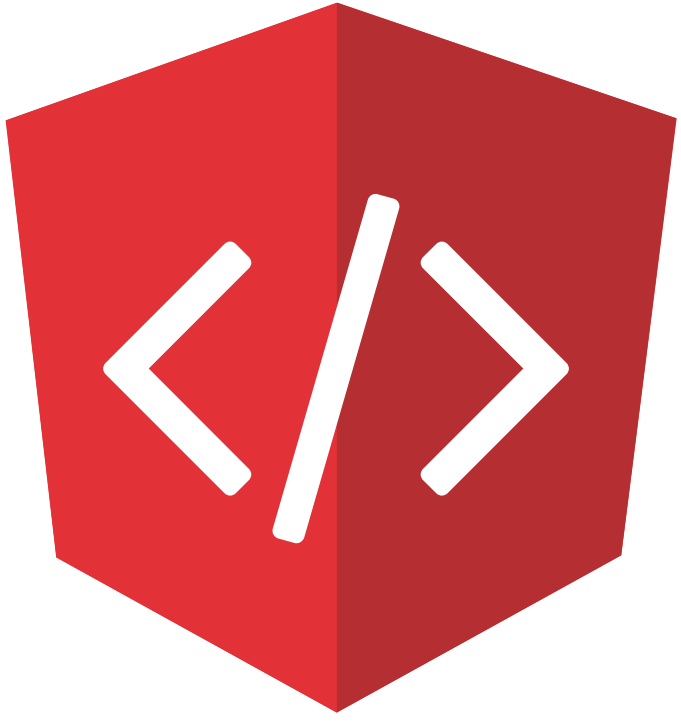
Dashboard

Komponente soll zunächst nur ein `string[]` darstellen

```
// dashboard.ts
import {Component} from 'angular2/core';

@Component({
  selector: 'dashboard',
  directives: [], // later: BookComponent
  template: `
    <h1>Bücher</h1>
    <p>{{ books }}</p>`
})
export class Dashboard {
  books: string[];

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

It works!

On my machine! ™



Components & Views

Bücher-Klasse

TypeScript-Typen verwenden

```
// book.ts
export class Book {
  rating: number = 0;

  constructor(public title: string, public comment: string) { }
}
```

BookComponent

zweite Komponente generieren lassen

```
$ ng generate component BookComponent
```

BookComponent

Komponente soll ein einzelnes Buch anzeigen

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

@Component({
  selector: 'book',
  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 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>
```

Dashboard

Dashboard soll zunächst nur ein einziges Buch anzeigen

```
// dashboard.ts
import {Component} from 'angular2/core';
import {Book} from '../../models/book';
import {BookComponent} from '../book-component/book-component';

@Component({
  selector: 'dashboard',
  directives: [BookComponent],
  template: `
    <h1>Buch</h1>
    <book [book]="book"></book>`
})
export class Dashboard {
  book: Book; // TODO books: Book[];

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

Dashboard importiert **BookComponent**

[Property bindings]

Ziel: **@Input()** der gebundenen Komponente
Property bindings sind durch **[ ]** gekennzeichnet

```
<book [book]="book" *ngFor="#book of books"></book>
```

Forms

Formular

Dashboard erhält zusätzliche Interaktionselemente

```
// dashboard.ts
@Component({
  selector: 'dashboard',
  directives: [BookComponent],
  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 *ngFor="#book of books" [book]="book"></book>`
})
/* [...] */
```

Referenzvariablen

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

```
<input name="title" #title>
<textarea name="comment" #comment></textarea>
```

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

(Event Binding)

Event bindings sind durch **()** gekennzeichnet

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

Interaktion

Komponenten sind die neuen Controller

```
// dashboard.ts

/* [...] */
export class Dashboard {
  books: 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, Input} from 'angular2/core';
import {Book} from '../../models/book';

@View({
  selector: 'book',
  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 class BookComponent {
  @Input() book: Book;

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

Externe Ereignisse

Sender: **@Output()** sendet Event aus Komponente heraus

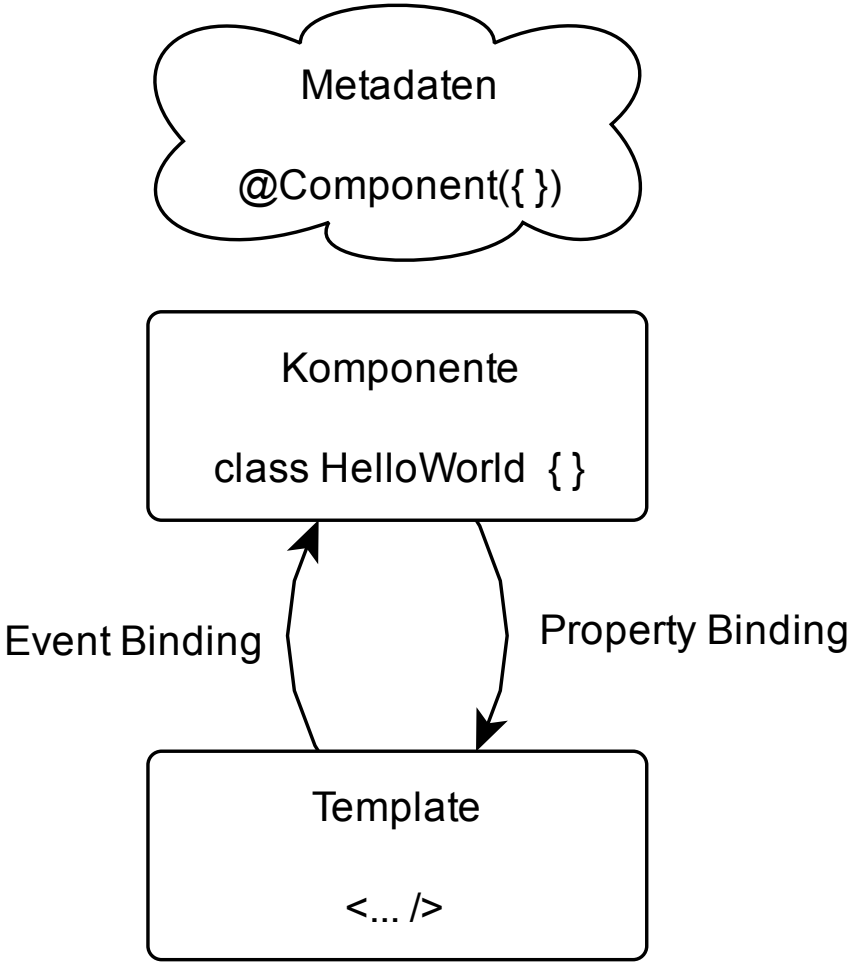
```
// book-component.ts
import {Component, Input, Output, EventEmitter} from 'angular2/core';
import {Book} from '../../models/book';

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

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

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

Auf Event reagieren

Empfänger: **(event binding)**

```
// dashboard.ts
/* [...] */

@Component({
  selector: 'dashboard',
  directives: [BookComponent],
  template: `
     <!-- ... -->
     <book ... (rated)="reorderBooks($event)"></book>
   `
})
export class Dashboard {
  /* [...] */

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

Zusammenfassung



Unit Tests mit

Jasmine

Tests im BDD-Style

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

angular2/testing

immer die gepatchten Test-Methoden verwenden!

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

Los geht's

```
$ ng test --watch
```

Spec

Simpler Test der Klasse BookComponent

```
import {it, describe, expect, inject, beforeEachProviders} from 'angular2/testing';
import {BookComponent} from './book-component';
import {Book} from '../../models/book';

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

  it('should increase book rating on rateUp()', inject([BookComponent], (bookComponent: BookComponent) => {

    bookComponent.book = new Book('Test Title', 'Test Comment');
    bookComponent.rateUp();

    expect(bookComponent.book.rating).toBe(1);
  }));
});
```

Spec

generierter Code zum Testen einer Komponente

```
describe('BookComponent', () => {

  it('should ...', injectAsync([TestComponentBuilder],
      (tcb:TestComponentBuilder) => {
    return tcb.createAsync(BookComponent).then((fixture) => {
      fixture.detectChanges();
    });
  }));
});
```

etwas aufräumen

beforeEach

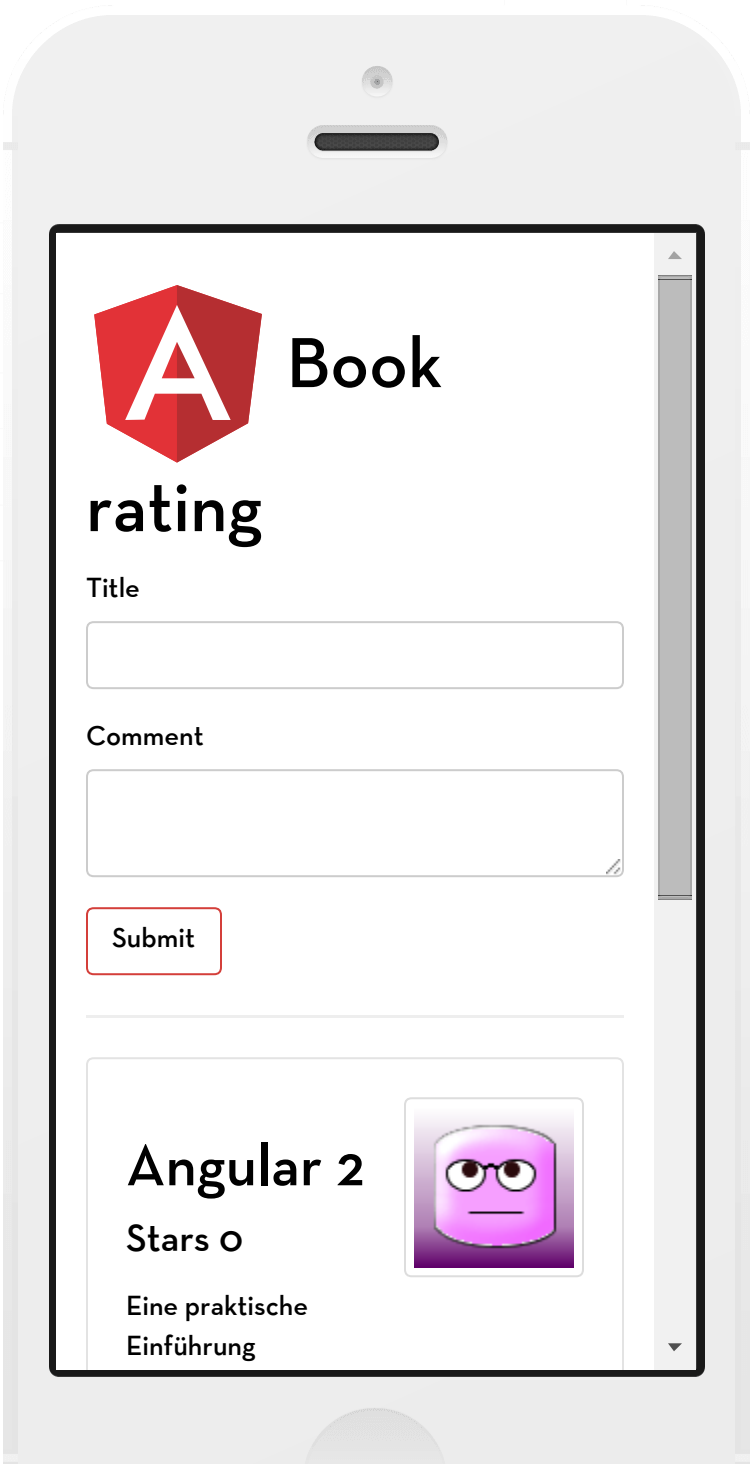
aufgeräumter Code zum Testen einer Komponente

```
describe('BookComponent', () => {

  var compBuilder: TestComponentBuilder;
  beforeEach(inject([TestComponentBuilder], _compBuilder_ => {
    compBuilder = _compBuilder_;
  }));

  it('should ...', injectAsync([], () => {
    return compBuilder
      .createAsync(BookComponent)
      .then((fixture) => {

        fixture.detectChanges();
      });
  }));
});
```

Spec

Fortgeschrittener Test: Initialisierte Komponente und Spy

```
it('should fire rated-event on rateUp click', injectAsync([], () => {
    return compBuilder
      .createAsync(BookComponent)
      .then((fixture) => {

        // given a component instance with an initialized book input
        var bookCmp = fixture.componentInstance;
        bookCmp.book = new Book('Test Title', 'Test Comment');

        // we fake the event emitter with a spy
        spyOn(bookCmp.rated, 'emit');

        // when we click on rateUp button
        var button = fixture.nativeElement.querySelector('button:first-of-type');
        button.dispatchEvent(new Event('click'));

        // we trigger the change detection
        fixture.detectChanges();

        // then the event emitter should have fired an event
        expect(bookCmp.rated.emit).toHaveBeenCalled();
      });
  }));
```


Book
rating
Title
Comment
Submit
Angular 2
Stars 0
Eine praktische
Einführung

Vielen Dank!

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