AngularJS 2 + ~~DDD~~
vs.
JavaScript-Frickelei

Ein kleiner Rant

Who?

- @ReneKriest
- Dev @ AOE
  - Tagsüber: Frontend
  - Nachts: Java/Android
- Relaunch für Congstar.de

Frame of Reference

- Kein Angriff auf Frickler
- Kein Angriff auf jQuery
- Ziel: Etwas provozieren, um anzuregen

Story

- Vor ca. 6 Jahren hatte ich noch typische jQuery-Pages entwickelt mit ca. 100-200 Zeilen JavaScript-Code
  - Works on different Browsers?
- Gegenwärtig entwickle ich JavaScript-Apps, die in der Woche auf 100-200 Zeilen kommen und ständig wachsen
  - Kann man das in einem Jahr noch erweitern ohne neuzuschreiben?

Frage

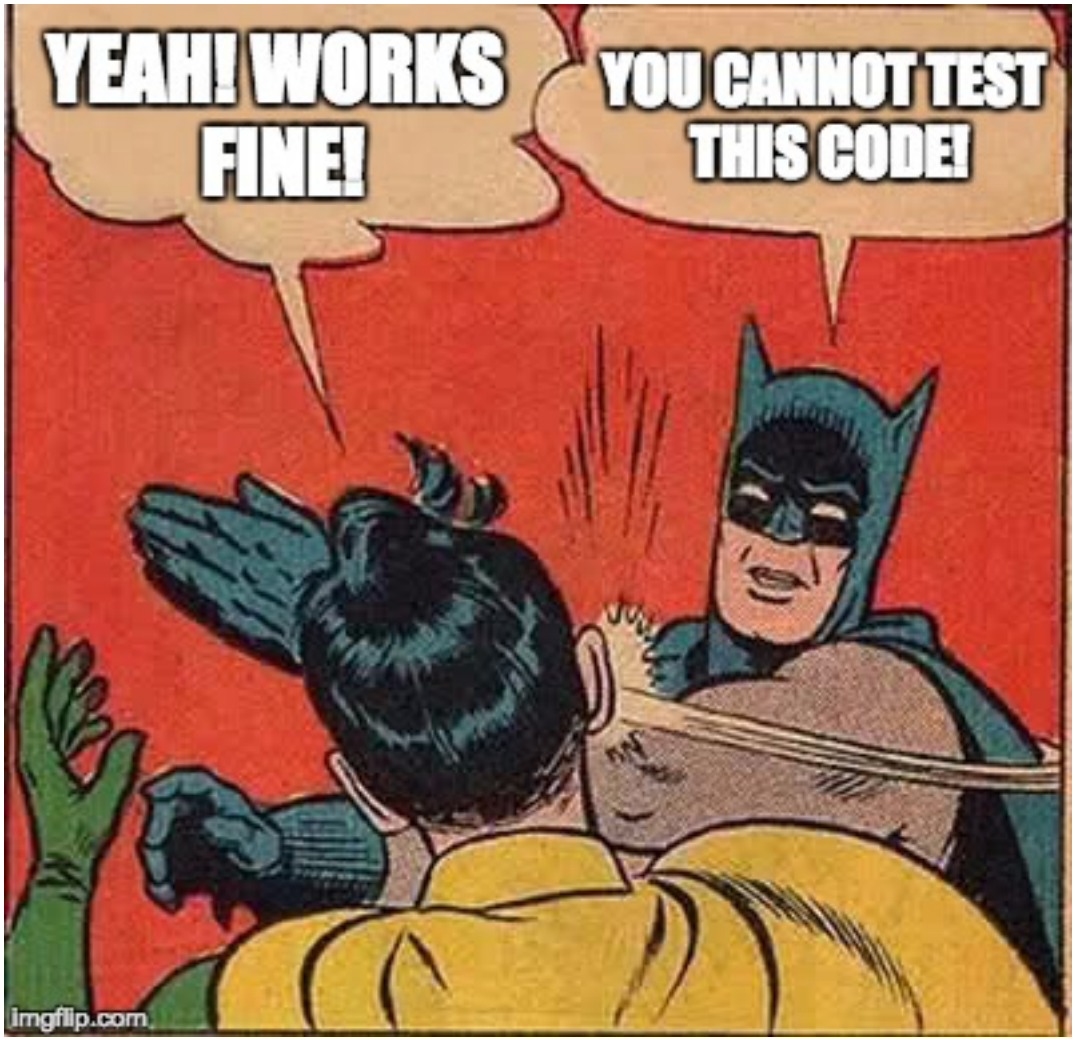
- Weiterfrickeln, oder anpassen, wenn ja wie?
- Oder anders formuliert: Was muss ich tun, um in 3, 4 Jahren noch einen Job als Frontend-Entwickler zu haben?

Was ist Frickelei?

- Boilerplate-Code/Copy-Paste-Code
- jQuery-Code/Vanilla Code
- Viel Code, wenig Ergebnis
  - Text im Browser ausgeben
- Wenig Tooling, viel Handarbeit
- Wiederholung
- Smell: Team-Größe liegt bei 1 Person

```
//<greetings></greetings>

$(document).ready(function() {
    $('greetings').on('click', function () {
        var greetingsText = $(this).text();
        window.alert(greetingsText);
    });
});
```


YEAH! WORKS FINE!
YOU CANNOT TEST THIS CODE!
imgflip.com

```
$(document).ready(function() {
    $('greetings').on('click', function () {
        showAlert.call(this);
    });
});

function showAlert() {
    var greetingsText = $(this).text();
    window.alert(greetingsText);
}
```


CONTROLLER HAS BEEN SEPARATED!
STFU! GLOBALS!
imgflip.com

```
// Global var! IIFE to the rescue
(function() {
    function showAlert() {
        var greetingsText = $(this).text();
        window.alert(greetingsText);
    }
})();


// Wait?! We need to expose this stuff
// to a global object, or use RequireJS!
// ...
```

Was ist cooler Code?

- AngularJS1/2 Code, ggf. React
- Muster/Pattern
- Team-Größe > 3 Entwickler
- Product Owner
  - Fachliche Vorgaben

```
import {Component} from "angular2/core";
import {View} from "angular2/core";

@Component({
    selector: 'greetings',
})

@View({
    template: '<greetings (click)="showAlert()">{{text}}</greetings>'
})

export class Example {
    text: string = 'Hello World!';

    public showAlert() {
        window.alert(this.text);
    }
}
```


HM... ANGULAR
YOU SAY.
WHY SO COOL?
imgflip.com

Why so cool:

- Modules
- Classes
- Components
- Views
- Statische Typisierung
- Data Binding
- MVC

```
import {Component} from "angular2/core";
import {View} from "angular2/core";

@Component({
    selector: 'greetings',
})

@View({
    template: '<greetings (click)="showAlert()">{{text}}</greetings>'
})

export class Example {
    text: string = 'Hello World!';

    public showAlert() {
        window.alert(this.text);
    }
}
```

Was bringt uns das?

- Fokus auf fachliche Anforderungen
- Weniger Code durch Abstraktion
- Weniger Fehler
- Geschwindigkeit
- Beherrschbare Komplexität
- Kann im Team bearbeitet werden, weil eher frei von persönlichen Präferenzen

Vorteile

- Modules
  - Kapselung, modularer Code
- Classes
  - Intuitivere und übersichtlichere Code-Organisation
- Components
  - Fasst zusammen, was zusammengehört und deklariert das Ergebnis im HTML
- Views
  - Keine Mischung mit Code: Trennung von Code und HTML

Vorteile

- Statische Typisierung
  - Refactoring
  - Duck Typing nach Bedarf
  - Ruby/Python schwenkten schon über auf optionale statische Typisierung, React/FB folgt mit Flow
- Data Binding
  - Änderungen werden automatisch propagiert
- MVC
  - Weils halt ein cooler Begriff ist ;)

TL; DR

- Bei JS-Apps: Komplexität bewältigen
- JavaScript hat ähnliche Herausforderungen wie vor 15 Jahren das Backend
- 2 Welten Theorie: JavaScript und **Enterprise-JavaScript**
  - Enterprise benötigt einen neuen JS-Stack
  - Entwickler müssen sich darauf einstellen

TL; DR II

- Backend hat gewonnen: Java hat mit ECMA6 JavaScript übernommen, nicht umgekehrt
- Künftig: Teamwork statt Einzelkämpfer
- Frontend-Zukunft ist rosig: Mehr Flexibilität, bessere Werkzeuge dank ECMA6/Angular