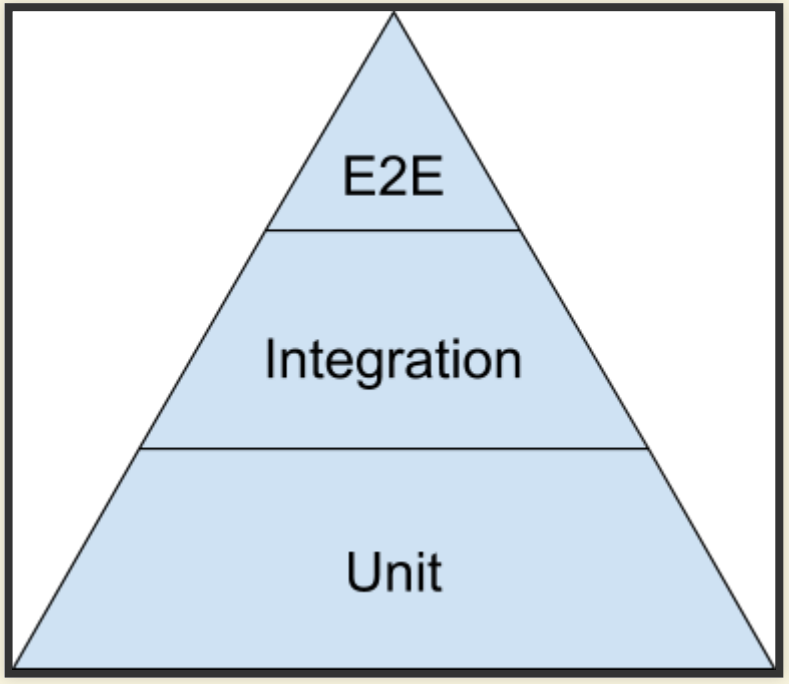
TESTEN VON VUE-KOMPONENTEN

LESSONS LEARNED

André Tebart - Modell Aachen

BISHERIGER WERDEGANG MIT VUE

- Start: Mitte April 2016
- Umsetzung vereinzelter Komponenten mit Vue
- Aktuell:
  - Aufbau einer Standard Komponenten Bibliothek
  - Umsetzung eines größeren Frontend Projekts inklusive Vuex
  - Ca. 180 Unit-Tests im Frontend



[Google Testing Blog: Just say no to more end-to-end tests]

UNIT TESTS

DAS WICHTIGSTE

BEKANNTES BEST PRACTICE

F.I.R.S.T.

- Fast
- Independent
- Repeatable
- Self-Validating
- Thorough

EINFACHE AUSFÜHRUNG

1. Repo klonen
2. yarn && yarn test

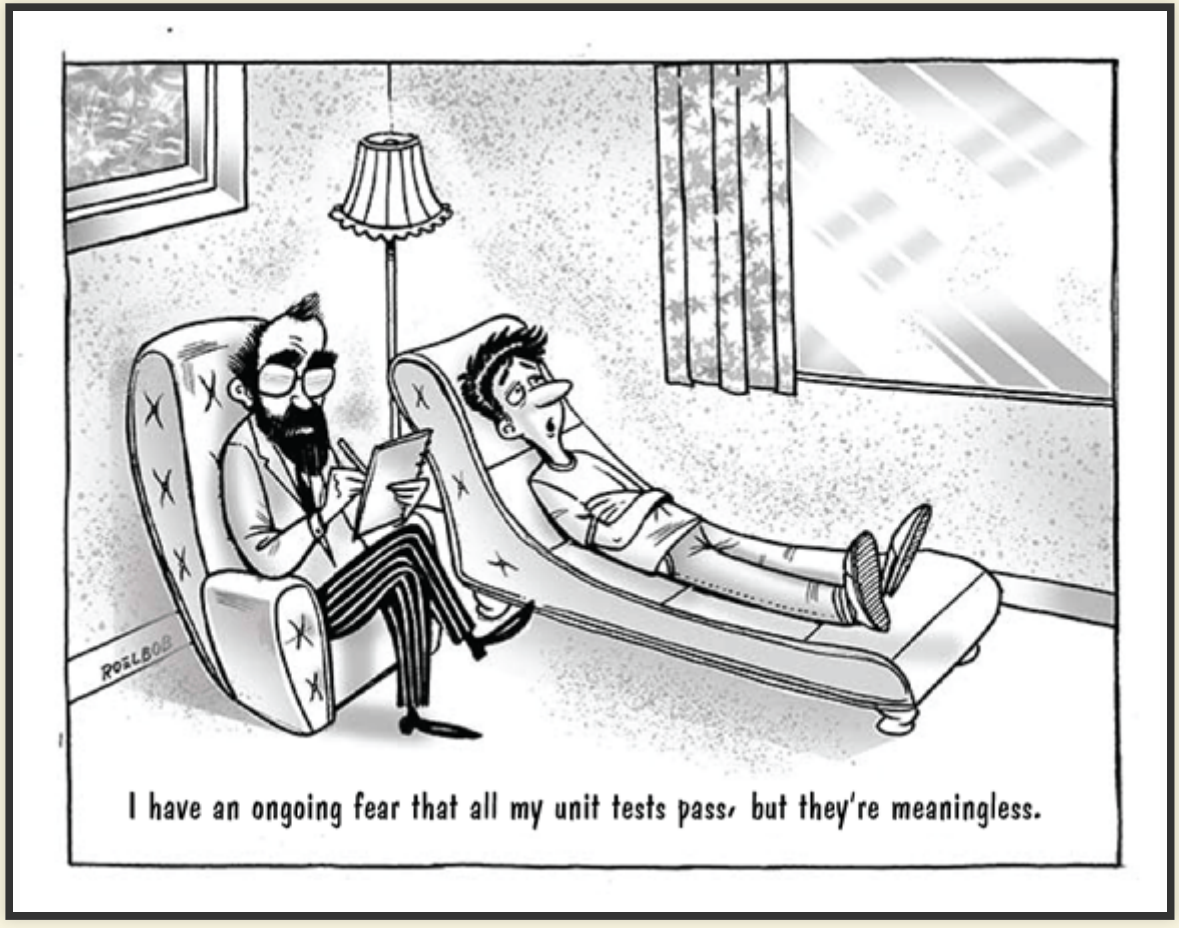
SCHNELLE AUSFÜHRUNG

- Automatische Ausführung nach jedem Speichern
- Sofortiges Feedback beim Entwickeln

LEICHT ZU VERSTEHEN

- Ein Entwickler braucht maximal eine halbe Minute, um einen Test Case zu verstehen

GETESTET WERDEN ANFORDERUNGEN, NICHT DIE IMPLEMENTIERUNG



- Für Refactorings müssen keine Tests angepasst werden
- Tests müssen nur angepasst werden, wenn sich die Anforderungen ändern
- **Tests dürfen nicht an die Implementierung gekoppelt sein!**

PRODUCTION CODE MUSS TESTBAR SEIN

- Nur guter Code ist auch gut testbar
- Schlechtes Design macht das Testen schwierig

TECHNOLOGIEN

- Karma
- Jsdom
- Jasmine
- Vue Test Utils

JASMINE

```
describe('My component suite', () => {
        beforeEach(() => {
                //Setup before each test case
        });

        afterEach(() => {
                //Teardown after each test case
        });

        it('has a component spec', () => {
                const a = true;
                expect(a).toBe(true);
        });
});
```

VUE TEST UTILS

```
import { mount } from '@vue/test-utils';
import MyComponent from './MyComponent.vue';

describe('My component', () => {
        let myComponent;

        beforeEach(() => {
                //mount returns a wrapper object
                myComponent = mount(MyComponent);
        });

        afterEach(() => {
                myComponent.destroy();
        });

        it('emits a destroy event when the destroy button is clicked', () => {
                const buttonWrapper =
                        myComponent.find(".destroy-button");

                buttonWrapper.trigger('click');

                expect(myComponent.emitted()['destroy']).toBeTruthy();
        });
});
```

CASE STUDY

TESTEN VON STANDARD KOMPONENTEN

- Test einer Tabellen Komponente

First Name ▲	Last Name ⇕	E-Mail	Gender ⇕
Elinor	Doone	edoone9@a8.net	Female
Jerry	Kindley	jkindley7@umich.edu	Male
Julissa	Nestoruk	jnestoruk2@joomla.org	Female
Karel	Sprackling	ksprackling3@marketwatch.com	Female
Mohammed	Beamish	mbeamish0@nyu.edu	Male

TEST STRATEGIE

GETESTET WIRD DAS INTERFACE DER KOMPONENTE

- Props
- Slots
- Events
- DOM (mit Vorsicht!)

DOM TESTEN

- Semantische Selektoren verwenden

```
//👎
wrapper.find(".container .bottom-row button:first-of-type");
//👍
wrapper.find(".add-new-item-button");
```

- Alternativ: Vue refs verwenden

```
wrapper.find({ref: "add-new-item-button"});
```

NICHT GETESTET WERDEN

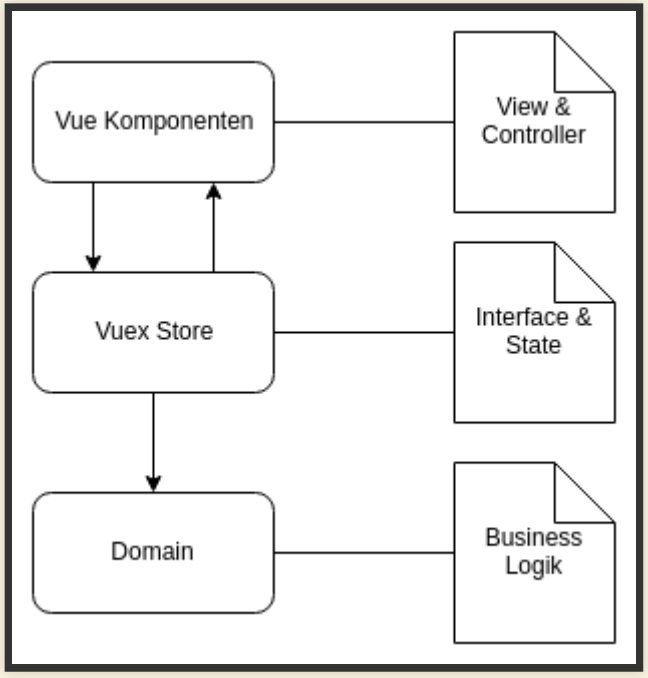
- Data Properties
- Computed Properties
- Interne Komponenten
- (Methoden)

CASE STUDY

TESTEN VON SPAS

- Vuex Store
- Business Logik im Frontend

ARCHITEKTUR



TEST STRATEGIEN

GETESTET WERDEN ANFORDERUNGEN ÜBER DAS STORE INTERFACE

- Mutations
- Actions
- Getters
- State

NICHT GETESTET WERDEN

- Implementierungsdetails der Domänenlogik

WAS IST MIT DEN VUE KOMPONENTEN?

- Testen wir aktuell kaum
  - Wir bauen auf unseren Standardkomponenten auf, die schon getestet sind
  - Gerade während der Neuentwicklung ändert sich der Aufbau der Komponenten regelmäßig
  - Fehler in den Vue Test Utils machen es schwierig komplexer strukturierte Komponenten zu testen

WAS MAN MITNEHMEN SOLLTE

- Tests prüfen Anforderungen und keine Implementierungsdetails
- Tests sind von der Implementierung entkoppelt
- Tests sind einfach zu lesen und schnell zu verstehen