Evolution of a web app

or

What to do with Angular 1?

@AndiMarek

Small Improvements

- 3-4 teams
- one product
- one repository
- Daily deployment
- > 700 customers

Tech Stack

- Java Backend (on Google App Engine)
- Wicket Frontend (Java Server Side Rendering)
- Angular (since 2013)

Angular was great

- SPA (faster UX, …)
- Faster development
- Attracted modern frontend developers
- Winner of 1. generation SPA frameworks
- Good testability (but not perfect, e.g. E2E)

Time goes by …

- Two-way data binding is no longer a good idea
- ES6 comes to life (we started to use Babel in 2015)
- Functional programming is gaining attraction
- Virtual DOM

What now?

- Keeping Angular
- Migrating to Angular 2
- Migrating to React

Keeping Angular

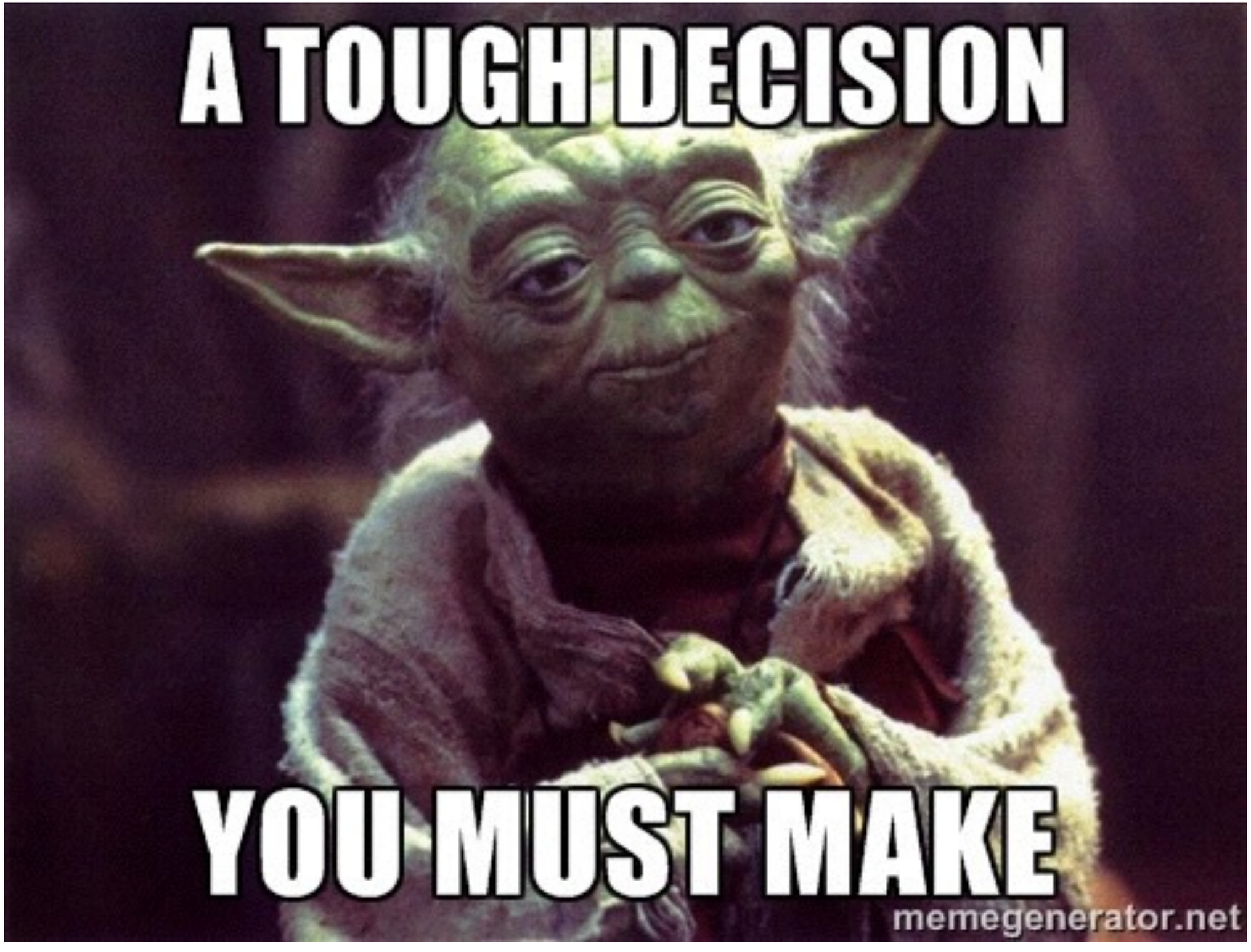
- Pro: Stable
- Pro: We know how to use it
- Contra: Outdated concepts (e.g. custom module system)

Migrating to Angular 2

- Pro: Supported migration path
- Pro: Better Angular in every aspect
- Pro: We could keep a lot of code and concepts
- Contra: Just a better Angular
- Contra: Typescript

Migrating to React

- Contra: Completely different technology
- Contra: Rewriting a lot
- Pro: Completely different technology
- Pro: ES6 - more language, less framework
- Pro: Huge eco system and very positive reviews
- Pro: We already have some experience


A TOUGH DECISION
YOU MUST MAKE
memegenerator.net

React it is

- One team is redesigning a complex Screen
- One team is developing a new feature

React mood barometer

- We love React!
- Simpler and better code
- UI UX people love it too
- We are feeling very confident that it was the right decision

React lessons learned

- Better testing with Mocha/Node instead of Karma/ PhantomJS
- Smaller and more components
- JSX has some minor issues
- Redux is great, but a lot of details are open for discussion
- A lot to learn

Example: Screen redesign

- UX Redesign of a complex screen and implement it in React
- One team over several weeks
- > 60 Redux Actions
- Ca. 80 components

Demo Screen Redesign

Relay

- Also tried to use Relay/GraphQL for Client-Server synchronization
- Great features, but complex and very new
- Integrating with Redux and Angular is not easily done
- Missing best practices because it’s so new and adaption is rather slow
- After a lot experimenting and discussions we decided against Relay (for now)

Challenges ahead

- Scaling development to 3-4 teams
- We are dealing with three frontend technologies in the foreseeable future (Wicket, Angular, React)
- Balancing refactoring, migrating and new features
- Learning all the different technologies takes time
- Progress doesn’t stop and the next big thing might already out there

Thanks!

@AndiMarek