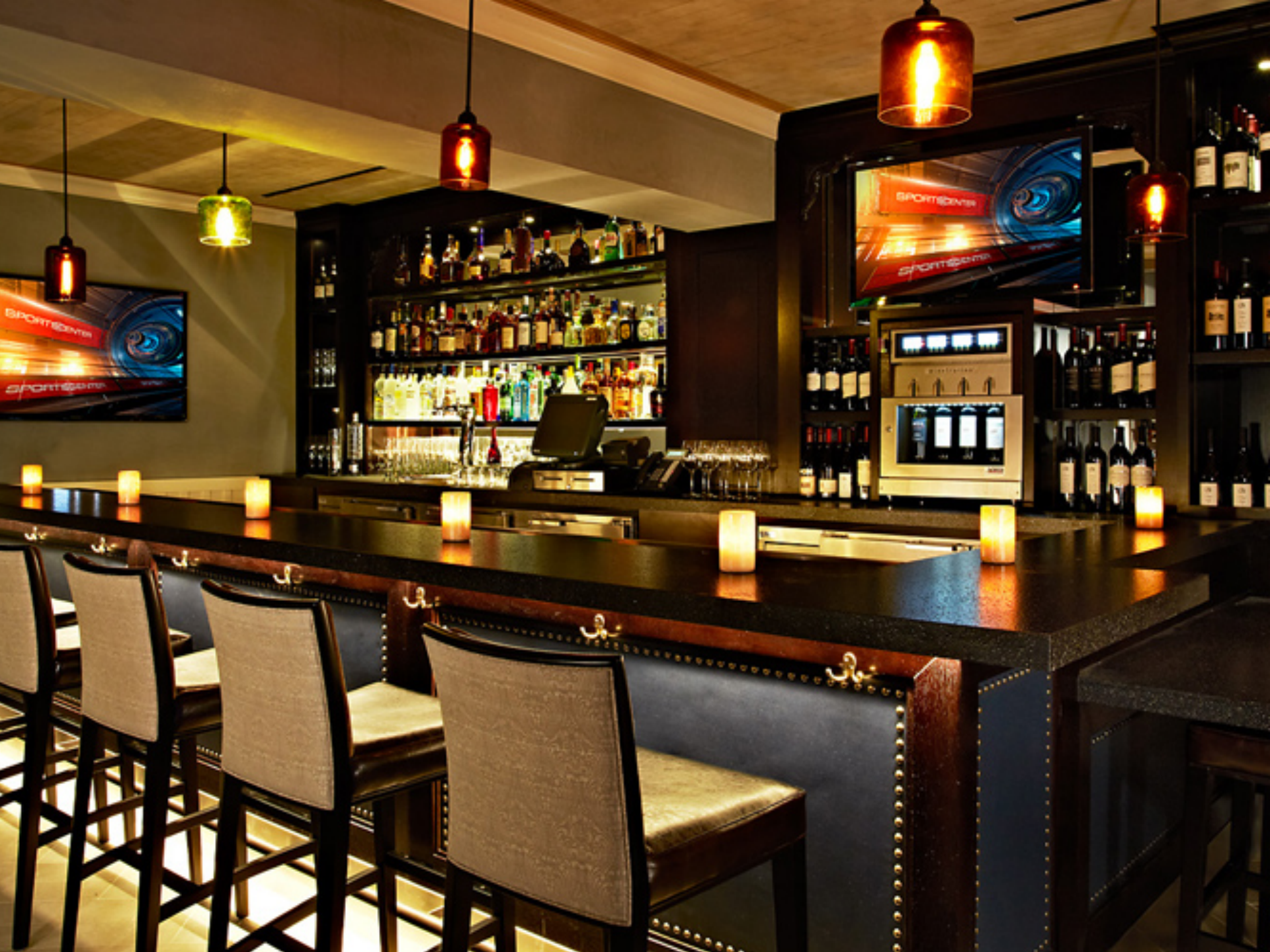
SPORTSCENTER
SPORTSCENTER
SPORTSCENTER

AngularJS & TypeScript

Stopping Frontend Development in Rails?

By Diego Steiner & Lukas Elmer

2015-06-24, Rails Höck, Wallisellen

renuo

Agenda

- Dialogue in brief
- Usability
- How we started with AngularJS
- Why not …?
- JS & Apps
- AngularJS
- TypeScript
- Recommendations & Conclusion

Dialogue in brief

New project in december, **multiple 1’000 reqs/day**!	How many dynos did you have to pay for?
None, deployed directly to cloudfront, decided to use **new technology**: Angularjs	JavaScript? Why when you can do **everything you need with ruby and rails**? Ruby is much better than JavaScript!
But you use CoffeeScript, too? Have you tried **TypeScript**? It’s very cool: you have optional type checking, a lot of new features from es6 and everything compiles down to plain JavaScript!	So what? What’s the use when you don’t have Gems? There’s a **Gem for everything** and usually very well tested!
There’s **NPM and bower** with thousends of good libraries aswell!	Ok, but why learn something new? Rails has been around **for years** and those JS-Frameworks seem to come and go every minute

Dialogue in brief

True, but JavaScript has been around even longer! Use **tools you already know** (Chrome dev console, browser)

That's the nice thing; We don't need it in the frontend. When you split it up the **backend is** even **swappable**!

Separation of concerns! Have you never had clean ruby code, but totally **untested mess** of JS code inside a rails project?

But with a separate frontend you can! There are tools that provide **tests as easy as rspec**. And you still can test the API **separately** and it's much cleaner!

But do you have good tools for the DB as well? Like **migrations**?

I like to have everything **in one place**!

Yes, but **you cannot test the JavaScript**, it's just too messy!

But why should I implement an API when it's not needed? **YAGNI**?!?

Dialogue in brief

Would you rather **programm ruby** or write HTML?
And have you ever seen a pure rails project **without JavaScript**?
But you **already do that** with your current JavaScript/CoffeeScript. You'll have to write some only once!
You don't have to. You can **split up the project in your team**!

Ruby of course!
No, but if I split it up, I will end up writing a lot of **duplicated code**!
Damn you're right! But I still **don't like JavaScript** or the likes of it!
Well, that doesn't sound so bad after all. Maybe rails is not doomed yet!

Usability

- Classic Rails
  - Disadvantage built-in
- **Need** JS to “fix” usability
- Offline

Technology can have huge impact on usability!

How we started

Babysteps

From our sewing-box :)

- May 2014: JS-only widget / app for worldcup
  - http://tamedia.renuo.ch/stats/
  - CoffeeScript
  - jQuery, **no** framework 

  - Deployed: pure frontend 

  - No JS Pipeline (e.g. Grunt, Gulp, NPM, Bower)
  - Generated with ruby tools (~Backend)
  - Offline 

  - Millions of pageviews

From our sewing-box :)

- Dec 2014: AngularJS app for food stats
  - Rails BE (no HTML, Assets Pipeline)
  - CoffeeScript
  - AngularJS 

  - Seperately tested 

  - Grunt pipeline, karma tests 

  - No integration tests (yet)

From our sewing-box :)

- Feb 2015: AngularJS app for elections
  - External BE 

  - CoffeeScript
  - AngularJS
  - Millions of pageviews
    - **we** had no problem :)
      - but helped fixing it :D

From our sewing-box :)

- Mar 2015: private project
  - JavaScript
  - Gulp 

  - Ionic 

From our sewing-box :)

- May 2015: internal chrome plugin
  - TypeScript 

  - Grunt

From our sewing-box :)

- Jun 2015: AngularJS app
  - Rails BE
  - TypeScript
  - AngularJS

Babysteps!

Why not…?

- Frameworks
  - Rails Frontend + jQuery
  - React
  - Backbone + Handlebars
  - Ember.js
- Languages
  - JavaScript
  - CoffeeScript
  - Dart
  - Opal

JavaScript

- Everywhere
- Everyone
- Tooling & Libs
- Browser Support (IE9+ or 11+)

JS Based Frontend Apps

- Very scalable & cacheable
- Architecture & software engineering in FE
- Frontend tests
- Can be offline
- Shared code between web, mobile, desktop, server
- Clear interface BE/FE
- Fast tests
- Simpler error resolution

AngularJS

- 2009
- “mature”, “stable”
- Active, Community

AngularJS

- Nothing is missing
- Dependency injection & tests
- Pushes towards good architecture

Typescript - Organizational

- Invented by C# inventor at Microsoft
- Driven by MS (and Google?)
- Open Source
- First MS Project on Github
- Active community

Typescript

- JS is TS already
- ES6
  - Classes
- Optional typing
- Refactoring
- External Type Definitions
  - Already implemented for a lot of libraries

```
class Point {
    x: number;
    y: number;
    constructor(x: number, y: number) {
        this.x = x;
        this.y = y;
    }
    getDist() {
        return Math.sqrt(this.x * this.x +
        this.y * this.y);
    }
}
var p = new Point(3,4);
var dist = p.getDst();
alert("Hypotenuse is: " + dist);
```

Recommendations

- Avoid putting everything in one app
  - Different jobs - different tools
  - Makes big app slow
- Babysteps
  - Use tools & languages you know
  - Change one thing at a time

Conclusion

- Rails will not die (yet)
  - It will just **play another part**
  - It will just **feel better** this way
- Embrace the new “competitors”
  - They’ve got **the right premises** to do the job better
  - They’re not here to **replace** rails
  - They’re here to **complement** it
- The rails-core-team already knows that
  - Rails::API will be part of Rails 5: https://github.com/rails/rails/pull/19832#event-328922869

Stop patching your FE.

Start using AngularJS & TypeScript.