From Backbone to Ember and

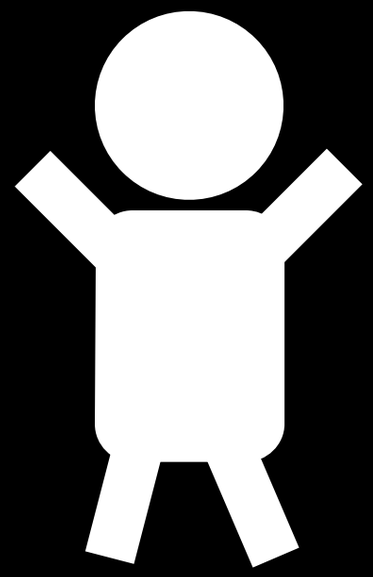
Back*(bone)* Again

Jon Knapp

Hi, I'm Jon.

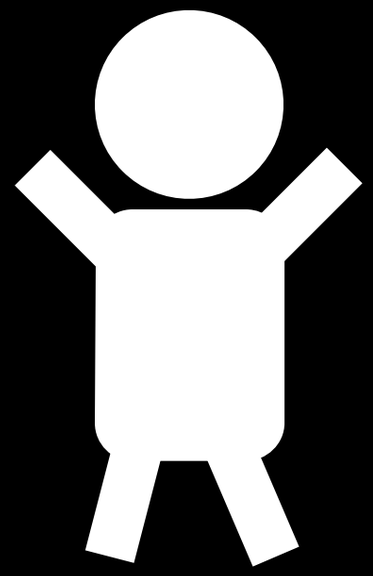
Developer at Coffee & Code.
Lover of tech.

Developer



```
module.exports = Work.create({
    code: function() {
        return 'the solution';
    }
})
```

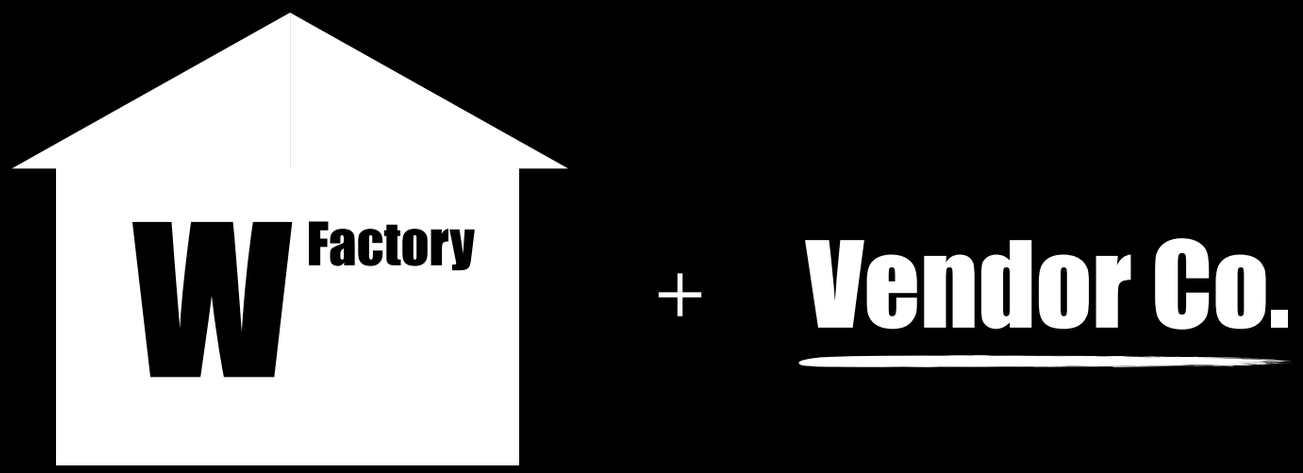
Consultant



```
module.exports = Work.create({
    code: function() {
        return 'the solution';
    }
})
```

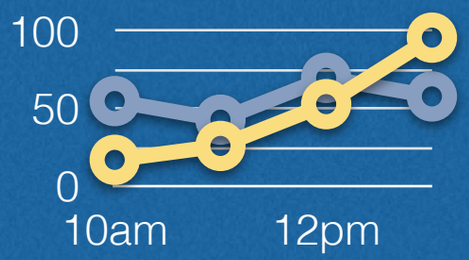
Widget Factory

Widget Factory



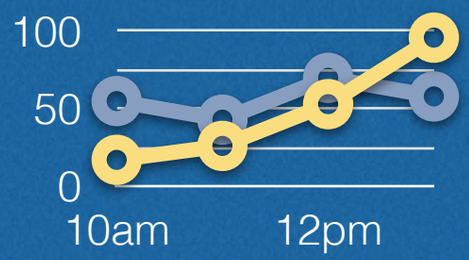
Widget Dashboard

Item 1



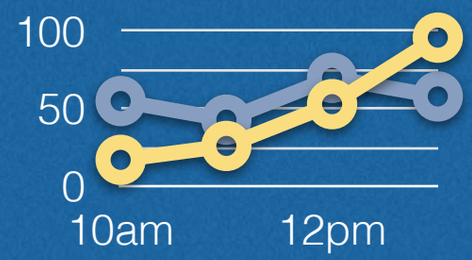
Interest Level	**72%**
Room #	**316**
# of Yawns	**12.4**

Item 2



Hunger Level	**12%**
Room #	**Gym 202**
# of Yawns	**204**

Item 3

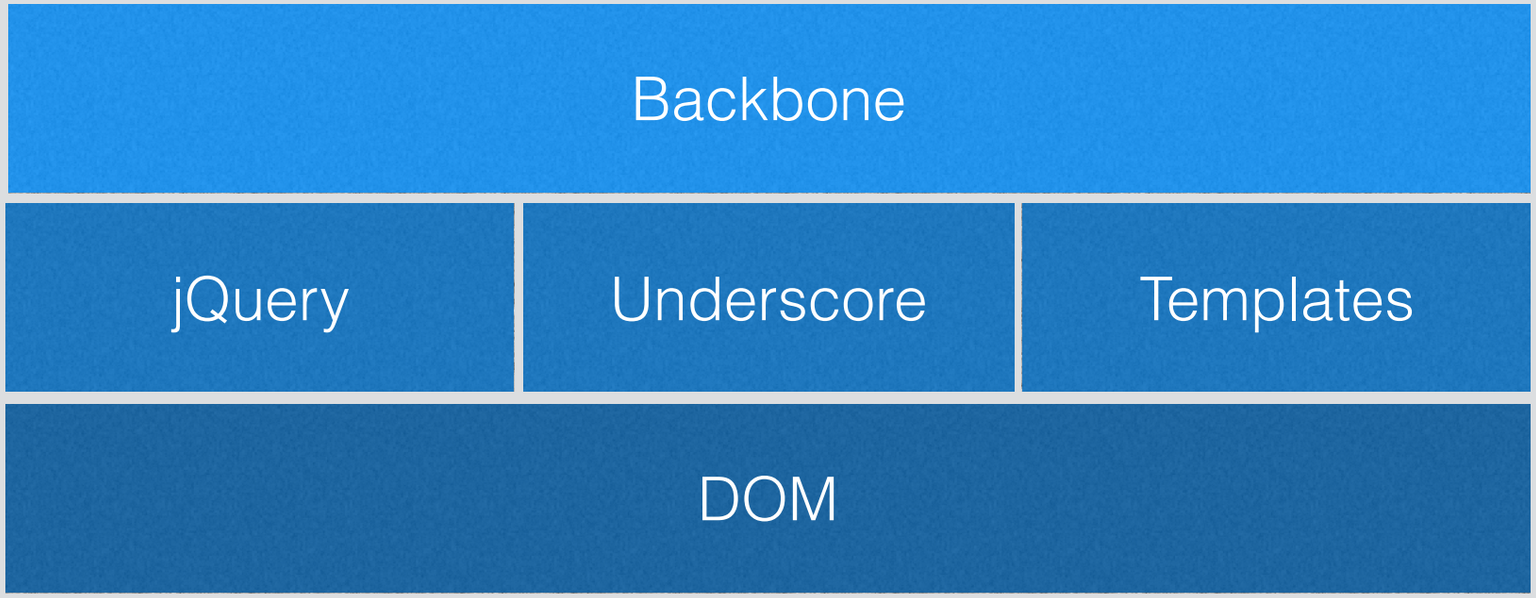


Hunger Level	**45%**
Room #	**106**
# of Yawns	**16**

Widget Dashboard

- single page Javascript app, built in Backbone
- pulls data from API server
- more and more features caused code bloat
- Widget Factory does not know much Javascript


BACKBONE.JS


Backbone
jQuery
Underscore
Templates
DOM

BACKBONE.JS

Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

BACKBONE.JS

Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

BACKBONE.JS

Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

BACKBONE.JS

Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

BACKBONE.JS

Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

BACKBONE.JS

Backbone.js gives structure to web applications by providing models with key-value binding and custom events, collections with a rich API of enumerable functions, views with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

BACKBONE.JS

```
var MyView = Backbone.View.extend({
  el: '#content',
  template: _.template($('#template').text()),
  render: function() {
    this.$el.html(this.template({
      language: 'Backbone'
    }));
    return this;
  }
});

var myView = new MyView();
myView.render();

<div id="content"></div>
<script id="template" type="text/template">
  <h1>Hello from <%- language %>!</h1>
</script>
```

http://codepen.io/coffeeandcode/pen/VYmZjK

BACKBONE.JS: Issues

- library does not do much, never will
- easy to start integration, difficult things around edges
- lack of project architecture leaves a lot up to the developers; “structure” is minimal
- easy to create memory leaks
- difficult to override non-restful API calls; opinionated

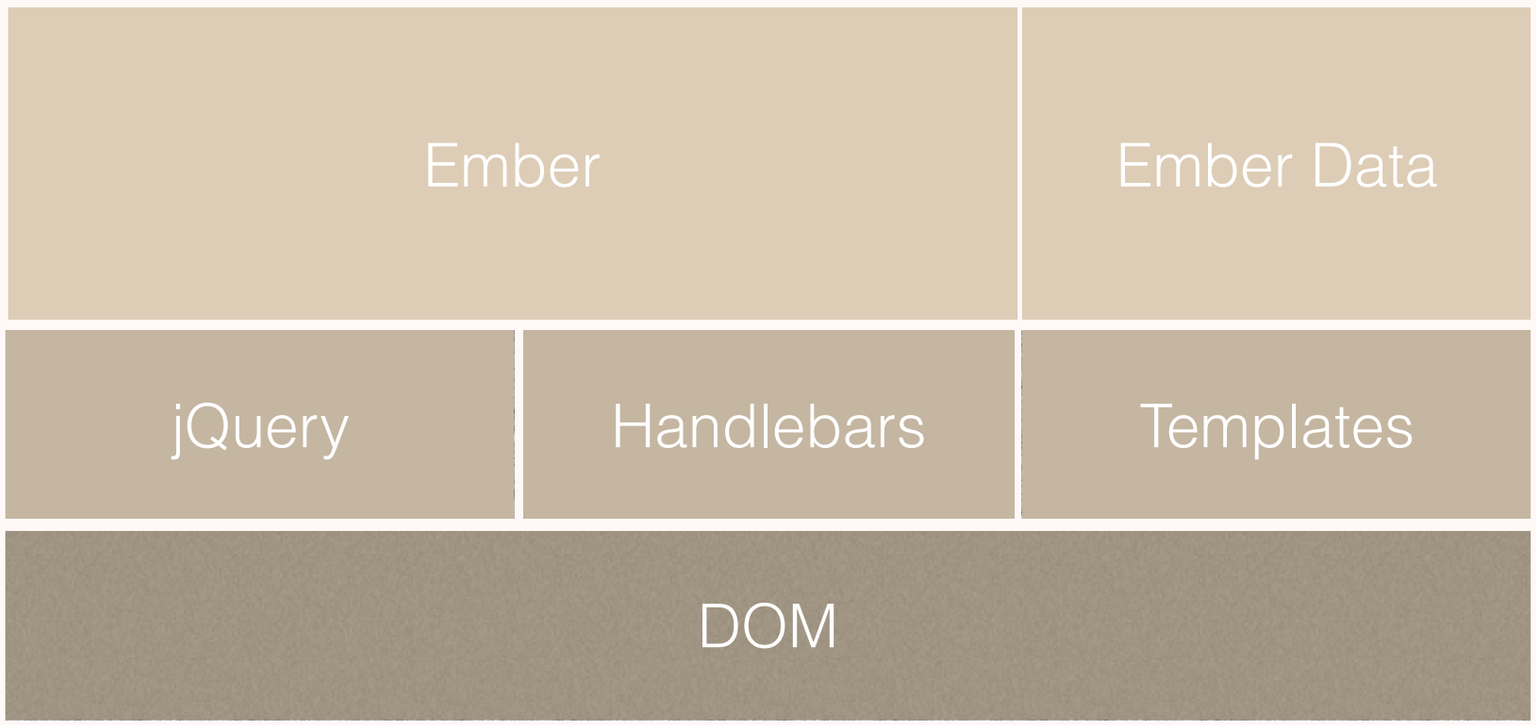
Widget Dashboard 2.0

- app rewritten in Ember.js
- goal was to write less code, easier to teach
- less moving parts == less to screw up
- Widget Factory still does not know much JS

Widget Dashboard 2.0

- app rewritten in Ember.js
- goal was to write less code, easier to teach
- less moving parts == less to screw up
- Widget Factory still does not know much JS


ember


Ember
Ember Data
jQuery
Handlebars
Templates
DOM

A framework for creating ambitious web applications.

A framework for creating ambitious web applications.

A framework for creating ambitious web applications.

Ember.js is an open-source client-side JavaScript web application framework based on the model-view-controller (MVC) software architectural pattern. It allows developers to create scalable single-page applications by incorporating common idioms and best practices into a framework that provides a rich object model, declarative two-way data binding, computed properties, automatically-updating templates powered by Handlebars.js, and a router for managing application state.[1]

1. http://en.wikipedia.org/wiki/Ember.js

Ember.js is an open-source client-side JavaScript web application framework based on the model-view-controller (MVC) software architectural pattern. It allows developers to create scalable single-page applications by incorporating common idioms and best practices into a framework that provides a rich object model, declarative two-way data binding, computed properties, automatically-updating templates powered by Handlebars.js, and a router for managing application state.[1]

1. http://en.wikipedia.org/wiki/Ember.js

Ember.js is an open-source client-side JavaScript web application framework based on the model-view-controller (MVC) software architectural pattern. It allows developers to create scalable single-page applications by incorporating common idioms and best practices into a framework that provides a rich object model, declarative two-way data binding, computed properties, automatically-updating templates powered by Handlebars.js, and a router for managing application state.[1]

1. http://en.wikipedia.org/wiki/Ember.js

Ember.js is an open-source client-side JavaScript web application framework based on the model-view-controller (MVC) software architectural pattern. It allows developers to create scalable single-page applications by incorporating common idioms and best practices into a framework that provides a rich object model, declarative two-way data binding, computed properties, automatically-updating templates powered by Handlebars.js, and a router for managing application state.[1]

1. http://en.wikipedia.org/wiki/Ember.js

Ember.js is an open-source client-side JavaScript web application framework based on the model-view-controller (MVC) software architectural pattern. It allows developers to create scalable single-page applications by incorporating common idioms and best practices into a framework that provides a rich object model, declarative two-way data binding, computed properties, automatically-updating templates powered by Handlebars.js, and a router for managing application state.[1]

1. http://en.wikipedia.org/wiki/Ember.js

Ember.js is an open-source client-side JavaScript web application framework based on the model-view-controller (MVC) software architectural pattern. It allows developers to create scalable single-page applications by incorporating common idioms and best practices into a framework that provides a rich object model, declarative two-way data binding, computed properties, automatically-updating templates powered by Handlebars.js, and a router for managing application state.[1]

1. http://en.wikipedia.org/wiki/Ember.js


```
var App = Ember.Application.create();

App.IndexRoute = Ember.Route.extend({
  model: function() {
    return {
      language: 'Ember'
    };
  }
});
```

```
<div id="content"></div>
<script data-template-name="index" type="text/x-handlebars">
  <h1>Hello from {{language}}!</h1>
</script>
```

http://codepen.io/coffeeandcode/pen/ZYBzad

: Issues

- very opinionated, very large codebase
- does black magic with Handlebars
- assumes you need the big guns
- two-way data binding by default
- easy to shoot yourself in the foot

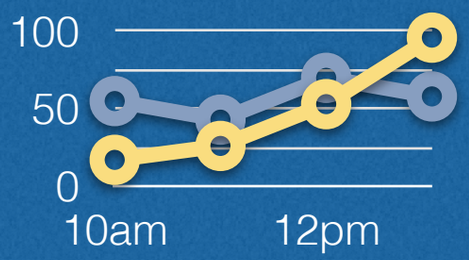
Two Main Problems

#1 Performance

Civic i-VTEC Engine by Kris Carillo: https://flic.kr/p/7VFLeC

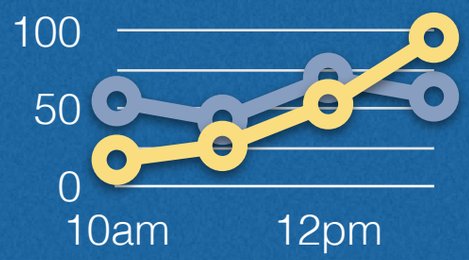
Widget Dashboard

Item 1



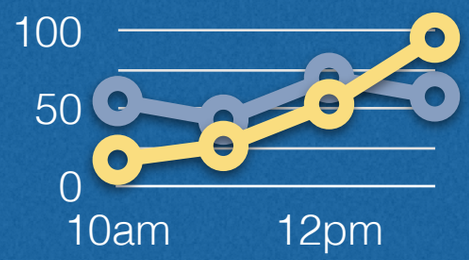
Interest Level	**72%**
Room #	**316**
# of Yawns	**12.4**

Item 2



Hunger Level	**12%**
Room #	**Gym 202**
# of Yawns	**204**

Item 3



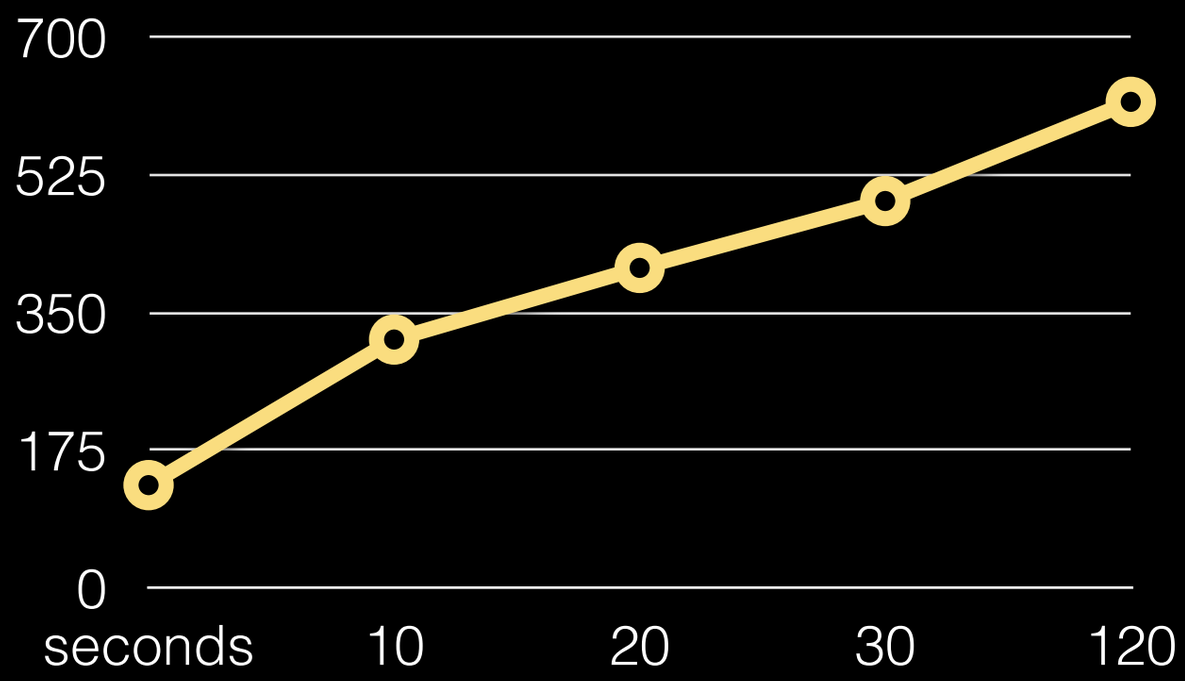
Hunger Level	**45%**
Room #	**106**
# of Yawns	**16**

Widget Dashboard

Poor Man's Testing

Client's Ember App

MB of memory

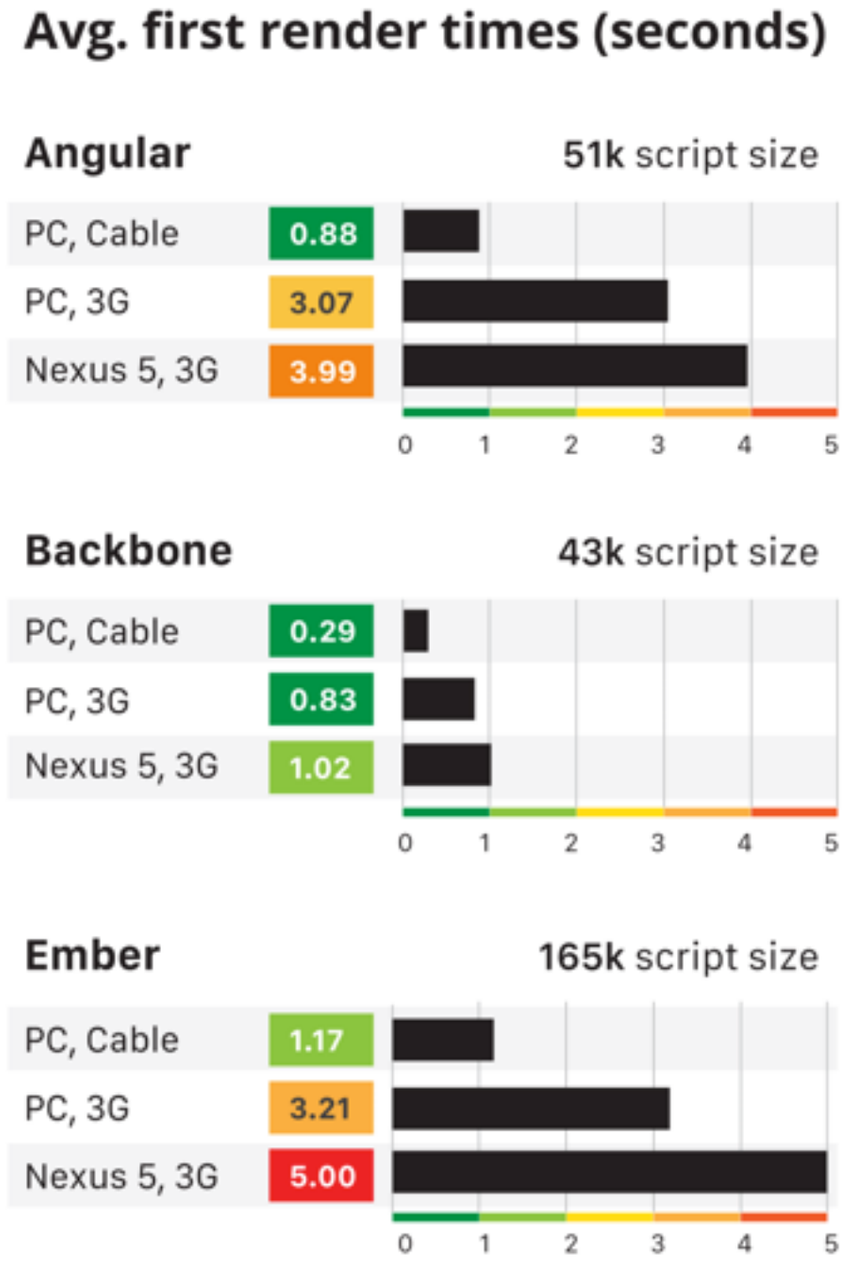


Design for Production Data

- How many widgets will clients have?
- How much data does each need?
- How does that effect our system?
- What does success look like?

TodoMVC

http://www.filamentgroup.com/lab/mv-initial-load-times.html

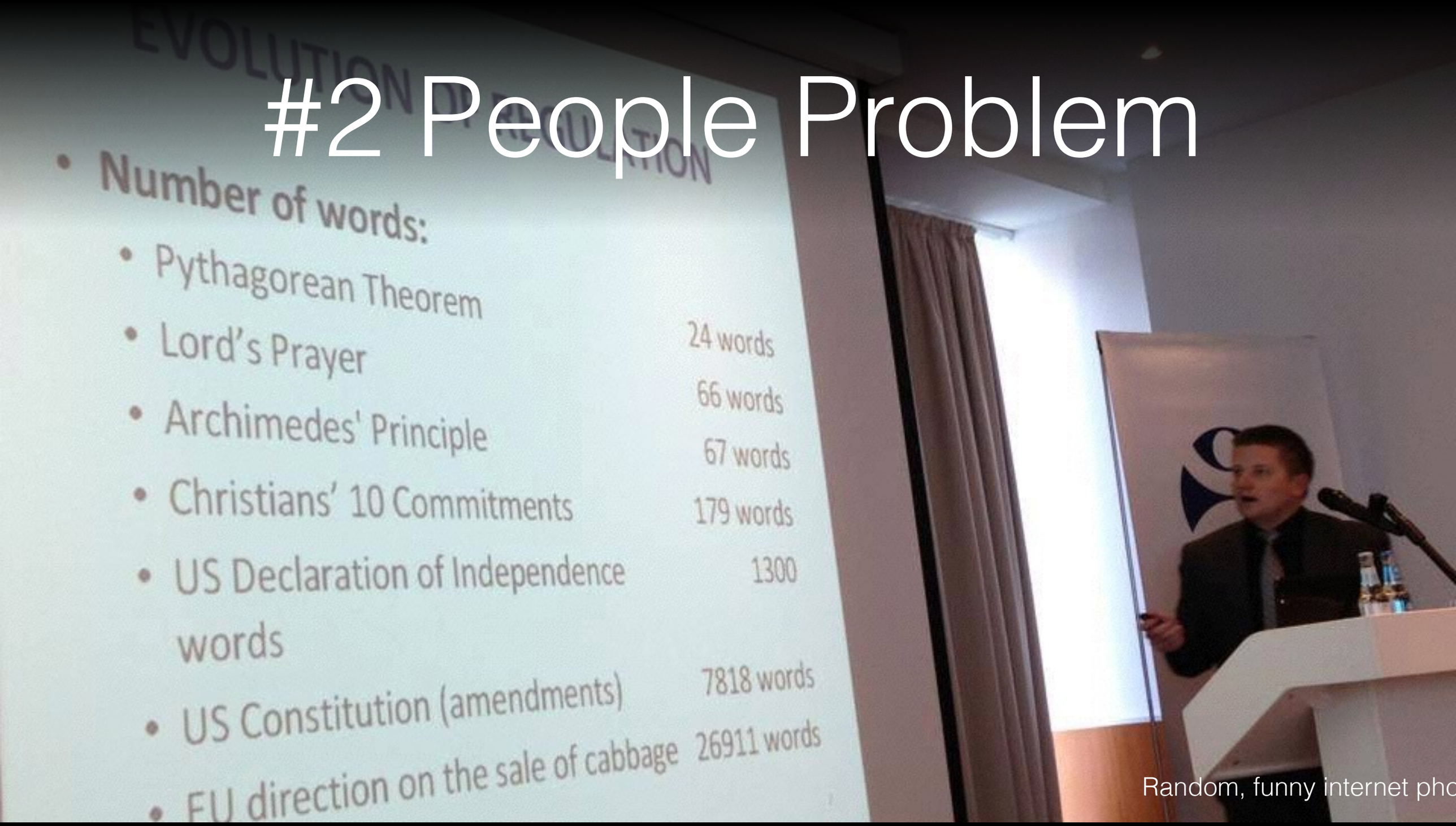


lines	words	chars	
12	36	197	backbone/js/app.js
41	166	1130	backbone/js/collections/todos.js
26	78	536	backbone/js/models/todo.js
26	67	499	backbone/js/routers/router.js
131	428	3703	backbone/js/views/app-view.js
132	496	3888	backbone/js/views/todo-view.js
368	**1271**	**9953**	**total**

```
$ find emberjs -name '*.js' | xargs wc -wcl
```

6	14	159	emberjs/js/app.js
59	165	1486	emberjs/js/controllers/todo_controller.js
51	128	1195	emberjs/js/controllers/todos_controller.js
16	38	432	emberjs/js/controllers/todos_list_controller.js
11	31	262	emberjs/js/helpers/pluralize.js
9	18	158	emberjs/js/models/todo.js
38	78	816	emberjs/js/router.js
12	33	298	emberjs/js/views/todo_input_component.js
202	**505**	**4806**	**total**

#2 People Problem



Random, funny internet pho

Non-Technical Requirements

- lack of knowledge in front-end technologies
- not enough pairing and information sharing
- team skill level was not adequate to pick up where Vendor Co. left off
- amount of documentation was lacking

Our Solution

- focus on knowledge sharing
  - pair programming / technical side projects
- build what the client understands
  - built JS workflow with the client's help
- use technologies with little change and a wealth of documentation
- we went back to Backbone (with Marionette)

Marionette

Backbone

jQuery | Underscore | Templates

DOM

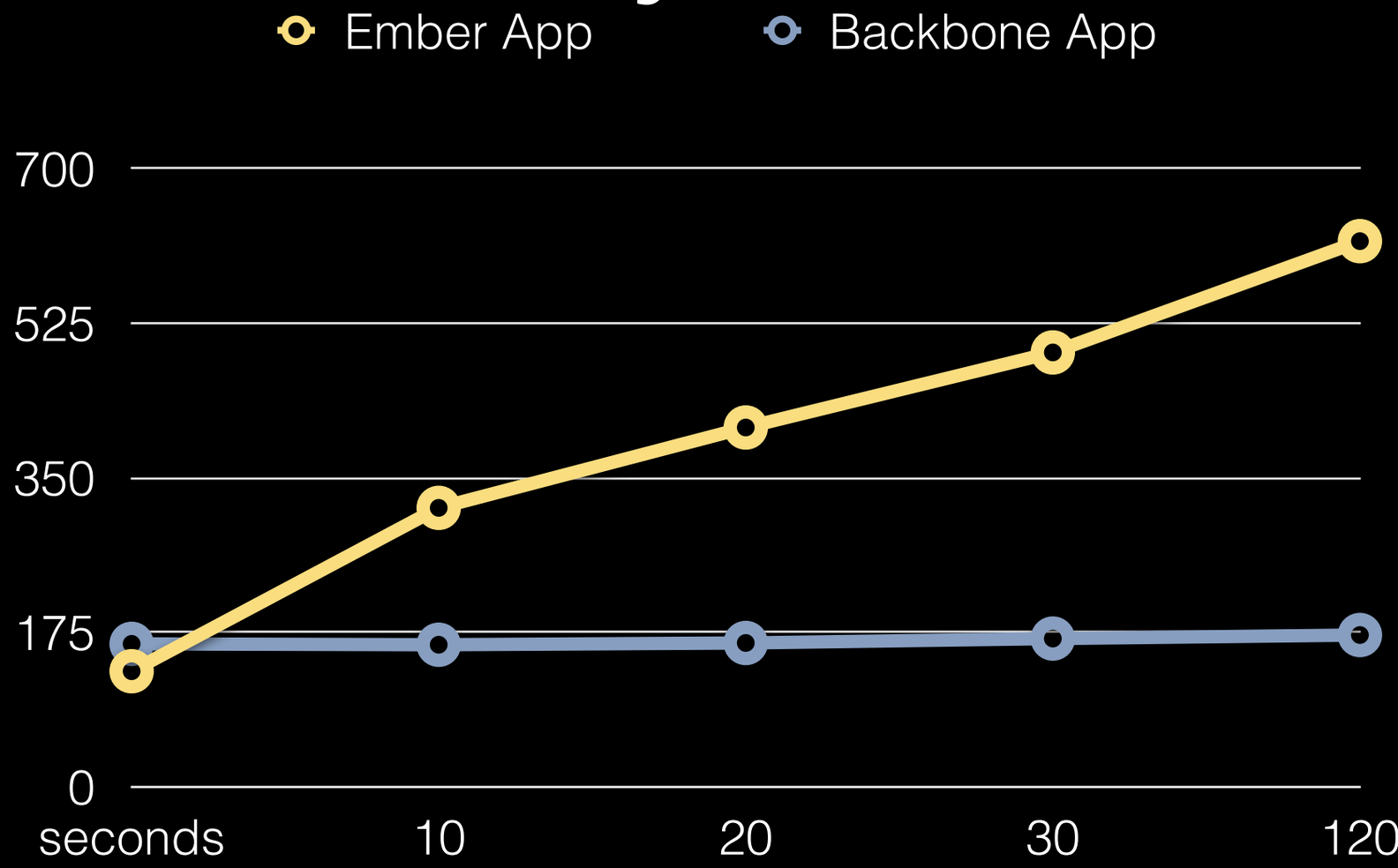

```
var MyView = Marionette.ItemView.extend({
  el: '#content',
  template: '#template',
  templateHelpers: function() {
    return {
      language: 'Marionette'
    };
  }
});

var myView = new MyView();
myView.render();

<div id="content"></div>
<script id="template" type="text/template">
  <h1>Hello from <%- language %>!</h1>
</script>
```

http://codepen.io/coffeeandcode/pen/raWBWE

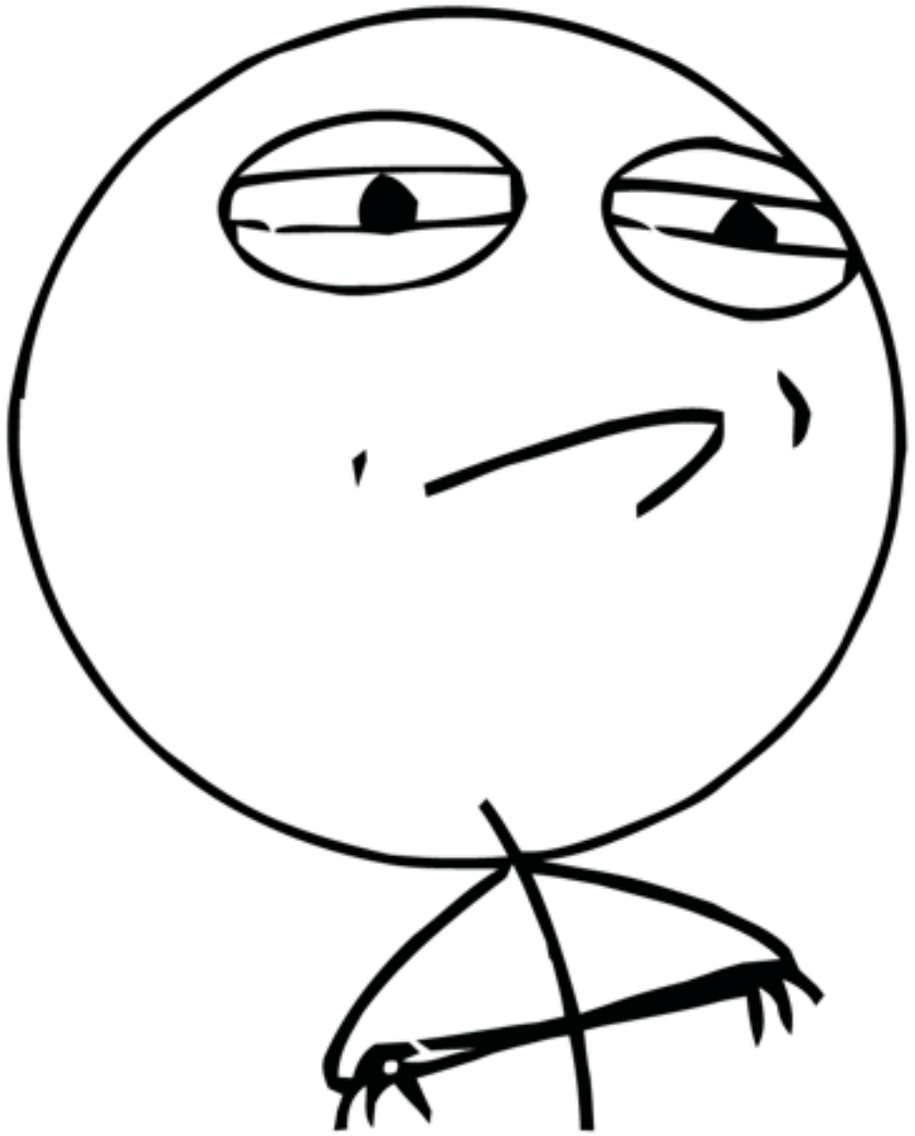
Activity Monitor



My Challenge To You

- build apps with production data in mind
- consider non-technical needs of project
- show love for all technology choices, but be hesitant to adopt new libraries

CHALLENGE ACCEPTED



Thanks

Jonathan Knapp
@CoffeeAndCode

http://coffeeandcode.com

Resources

Slide 1 image: https://flic.kr/p/8Y8T5T

Slide 2 image: Keynote default :)

Slide 35 image: https://flic.kr/p/7VFLeC

Slide 39: TodoMVC performance work by Filament Group: http://www.filamentgroup.com/lab/mv-initial-load-times.html

Slide 39: Reference to Glimmer being fast: http://www.codekitchen.ca/visualizing-glimmer-performance/

Slide 41 image: the internets; could not find source