

# LARGE ANGULAR PROJECT POSTMORTEM



@wbucksoft  
willbuck @ github

@zlegein  
zlegein @ github

# INTENDED AUDIENCE

- About to start / in thick of large Angular project
- Know the basics but want more guidance

*Caveat: We're not experts, just experienced*

*Still don't have all the answers*

*We're assuming we know what you want to know, you know better,  
so...*

# ASK QUESTIONS!

# A LITTLE HISTORY

- Built original 'prototype' with no knowledge of how it'd be used
- Closest Model (EMRs) didn't quite fit
- Replacement to current and original application
- We learned something about the domain model
- Only need to support 1 browser (firefox)
- Minimal users of the application

# ANGULAR AT VIRTUWELL



- Started in 2012 building a Backbone app
- Ended up with a lot of code & inconsistent 2-way binding
- Chose AngularJS for our next big project
- Team fell in love
- Re-wrote much of Backbone app in Angular in short matter of months

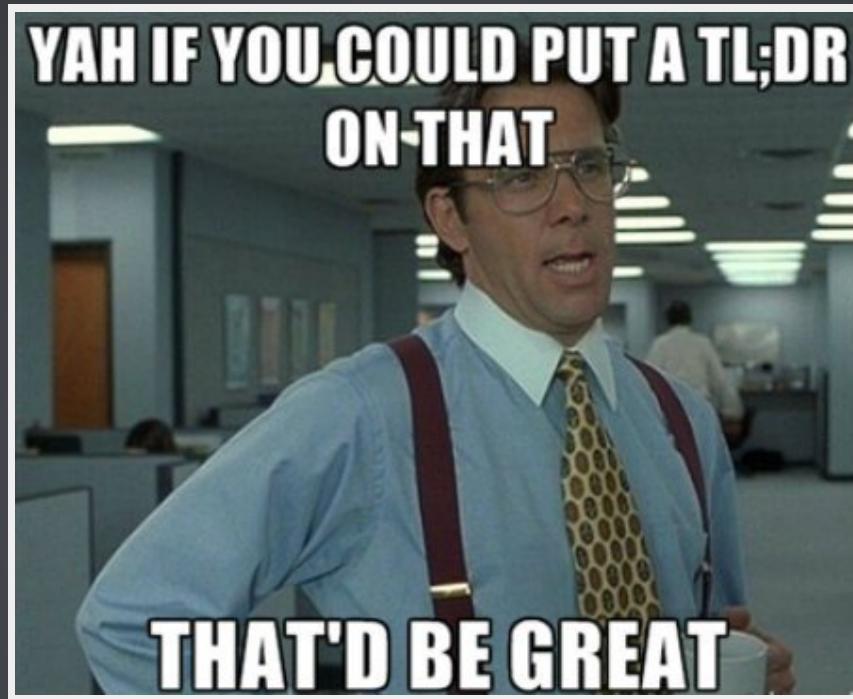
# WHY ANGULAR?

- Easy 2-way binding
- Grails-like MVC(S)
- Light, fast, productive
- Lots of activity & adoption
- TESTABILITY

# THE STACK

- Grails + Angular
- Grunt
- Coffeescript
- Jasmine & Karma
- Bower
- SASS (& Compass)

# TAKEAWAYS (TL;DR)



# READ A STYLE GUIDE

- John Papa (My Personal Favorite)
- Todd Motto (Similar, Encourage both)
- Google Best Practices for Structure (LIFT)

# GOOD RESOURCES

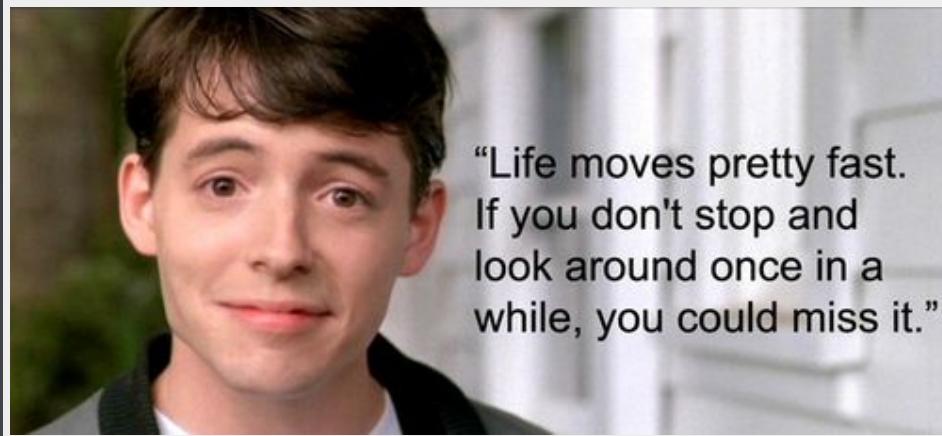
- Egghead bite-sized video chunks
- Plunker small examples
- Thinkster Cool tutorial app, not all best practices
- NG-Book Comprehensive & constantly up to date
- NG-Conf Videos, esp DoubleClick Team
- Year of Moo (Matias' Blog)

# USE TOOLS / LIBRARIES / STRATEGIES YOU FIND USEFUL

For us, this was

- Angular-UI (Router & Bootstrap)
- Restangular
- Yeoman to get started
- A bit of moment.js and underscore.js
- TDD Unit Tests w/ jasmine
- Git & gitflow branching model
- "Use what works" & "just ship it"

# JS LIFE MOVES FAST



"Life moves pretty fast.  
If you don't stop and  
look around once in a  
while, you could miss it."

Stay in touch w/ current happenings

Will help a lot with finding documentation / avoiding odd bugs

# ANGULARJS? THERE'S A MEETUP FOR THAT!



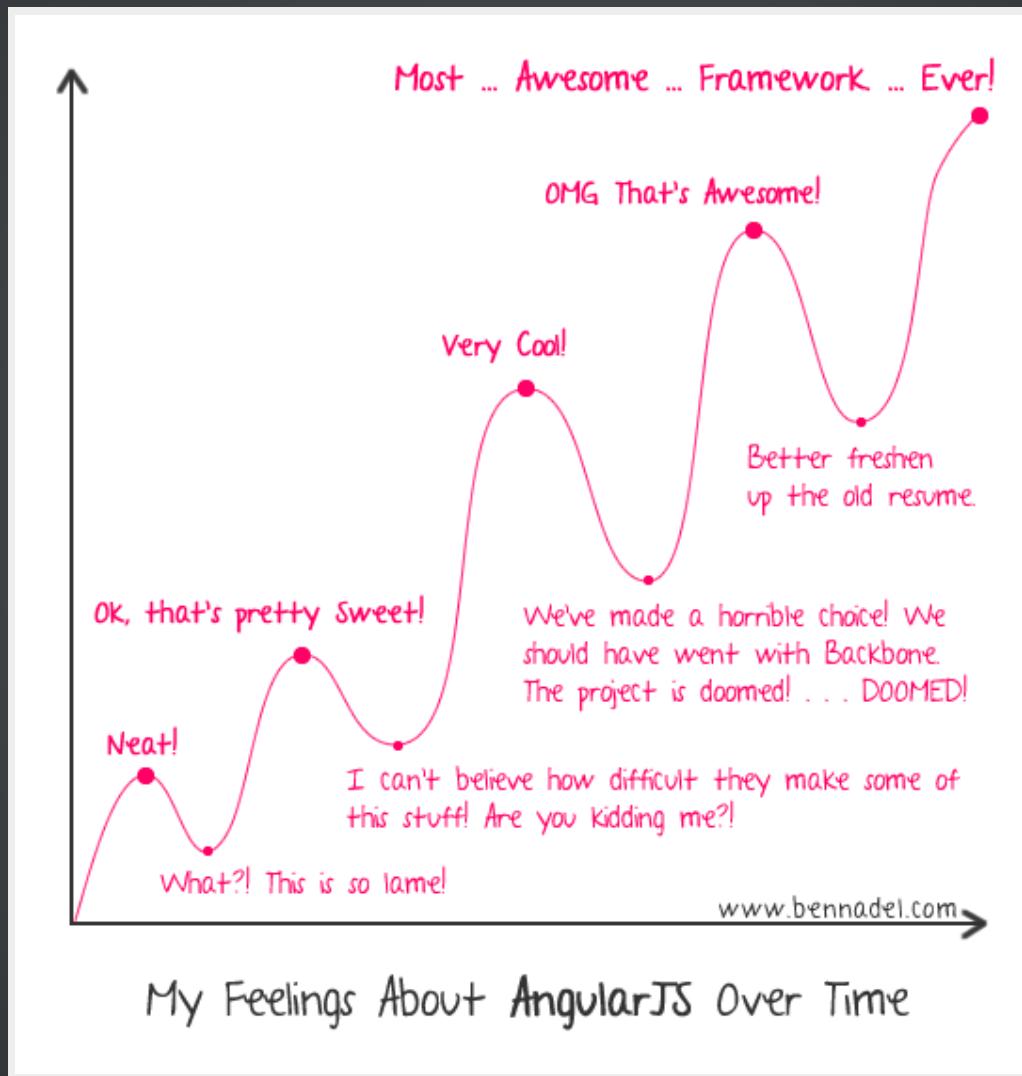
AngularMN meets First Wednesday of Every month  
At Virtuwell Headquarters  
Follow us @AngularMN

# LEARNING PROCESS

A.K.A. Toughest Hurdles Getting Started



# THE LEARNING CURVE



Courtesy @bennadel

# UNLEARNING JQUERY MINDSETS

- Angular is more framework than library
- Angular is more declarative than imperative
- Two-way data binding vs manual DOM management
- Dependency Injection & modularity vs. `$.ready()`
- Design your models first, not your page
- Having Backbone knowledge was helpful here

# FIRST PASS: CONTROLLER ALL THE THINGS

- Intros make a controller, so everything is a controller, amirite?
- Need to coordinate between controllers though...
- ... So use `rootScope.$emit / $on`, listen for model changes?

# FIRST PASS: QUICKLY BECOMES UNWIELDY

- Avoid "scope soup"
- controllerAs syntax helps
- Share data in a .factory (or .value perhaps)
- Magic controller inheritance makes very odd-looking code
- If you're going to do it, at least give it some structure

# SCOPES

Beware of bind by value vs bind by reference

```
$scope.myBoolean = SomeService.thatBoolean  
// Asking for trouble  
SomeService.thatBoolean = false  
// Will save you headaches  
SomeService.thatBoolean = {value: false}
```

And for arrays...

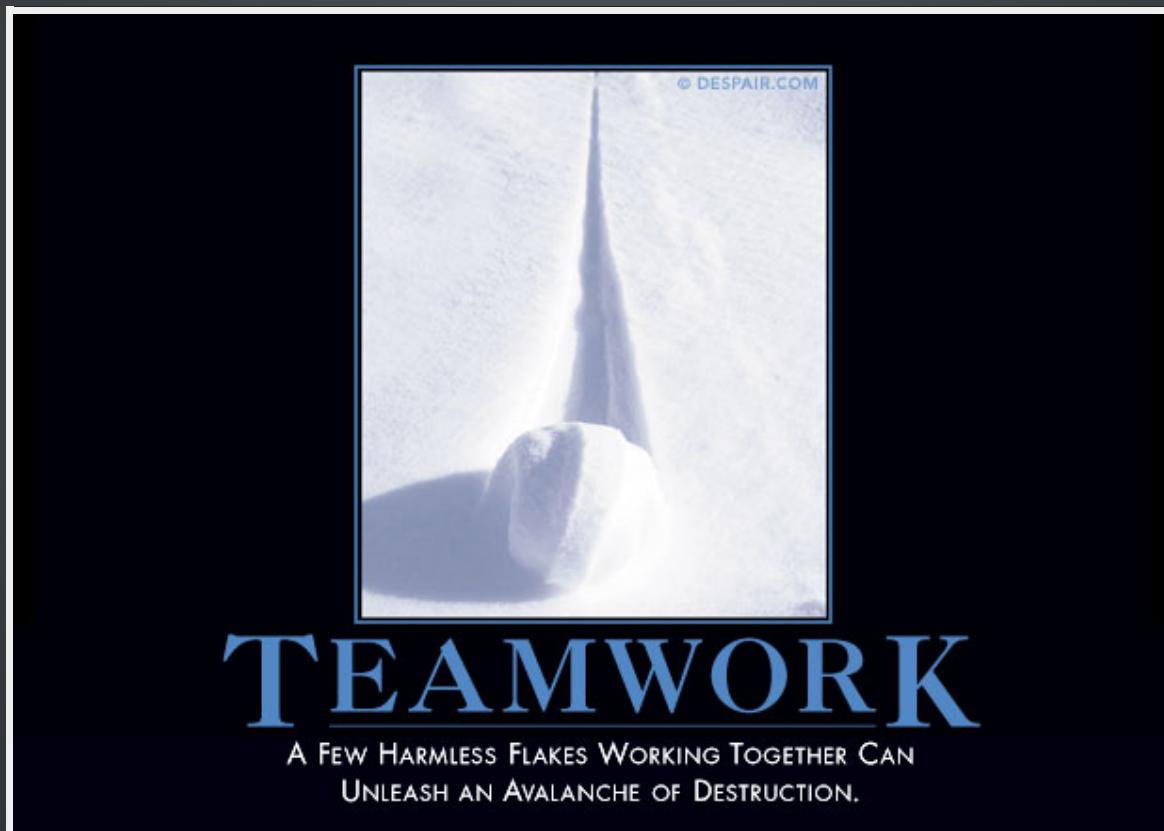
```
_._mixin({  
    bindArray: (origArray, newArray) ->  
        if newArray and origArray  
            origArray.length = 0  
            origArray.push(val) for val in newArray  
})
```

# ANGULAR SERVICES

- Make these fat (little / no logic in Controllers)
- Just use .factory, clearer than .service and not much difference
- Give these their own module for sharing

# THE JOURNEY TO DELIVERY

Things we encountered as a team that helped us towards  
**SHIPPING**



# SOME QUICK TIPS

- Controller vs Ctrl
- use prefix for directives
- wrap scope values in object
- use \$scope.\$digest vs \$scope.apply()

# MODULES

- how to organize
- directives, filters (and services?) are common
- naming, be consistent

```
angular.module('profile.notes', ['ui.router', 'ui.bootstrap', 'profile.path']
  .controller('ProfileNotesController',
    function($scope, $stateParams, $location, $modal, ProfileNotesService
```

```
▼ └─ angular
    ▼ └─ scripts
        ► └─ app
        ► └─ billing
        ► └─ case
        ► └─ cma
        ► └─ common
        ► └─ events
        ► └─ followup
        ► └─ header
        ► └─ interview
        ▼ └─ profile
            ► └─ controllers
            ► └─ dialogs
            └─ _history.html
            └─ _notes.html
            └─ _patient.html
            └─ profileModule.coffee
            └─ profileTabMain.html
    ▼ └─ queue
        ► └─ controllers
            └─ _followUp.html
            └─ _metrics.html
            └─ _rcb.html
            └─ _search.html
            └─ _treatments.html
            └─ queueModule.coffee
            └─ queuePageMain.html
        ► └─ rcb
        ► └─ search
        ► └─ treatment
            └─ app.coffee
        ► └─ styles
            └─ bootstrap
```

# PACKAGE DEPENDENCY

- npm used to install grunt and bower and yeoman
- version equivalent settings in bower and node

```
"dependencies": {  
    "angular-mocks": "~1.2.16",  
    "angular-scenario": "~1.2.16",  
    "es5-shim": "~2.0.8",  
    "angular-resource": "~1.2.16",  
    "angular-cookies": "~1.2.16",  
    "angular-sanitize": "~1.2.16",  
    "angular-bootstrap": "~0.10.0",  
    "jquery": "2.0.3",  
    ...  
}
```

# DEM TILDES



## introduce grunt clean task

- delete node modules
- delete bower components
- npm cache clean
- npm install
- bower cache clean
- bower install

# TESTING

- breaking code base into modules was the biggest challenge to testing
- unit test needed every module that code used under test
- wrote custom matcher for deep comparison testing
- introducing restangular added new challenge for comparison
- wrote unit test for filters, directives, controllers, factories
- \$httpBackend.flush() to ensure all async request are done
- rootScope.\$apply() triggers \$digest cycle to ensure all changes take effect

```
angular.equals(obj1, obj2)
```

## karma.conf.js

```
files: [
  'web-app/vendors/**/*.js',
  'web-app/js/**/app.js',
  'web-app/js/**/*Module.js',// need to load all the modules first
  'web-app/js/**/*.js',
  'test/**/*.coffee'
],  
  
// enable / disable watching file and executing tests whenever any file changes
autoWatch: true,  
  
// Continuous Integration mode
// if true, it capture browsers, run tests and exit
singleRun: false
```

# ROUTES

- Started with ngRoute, quickly moved to ui.router
- ngRoute is based on routes in your application
- ui.router is based on what state your application is in
- ui.router allows for nested states and views
- ui.sref directive that binds anchor tags to states



```
.state 'case',
  abstract: true
  url: '/case'
  templateUrl: 'case.html'
.state 'case.tabs',
  views: {
    "tabs@case" : {templateUrl: 'tabs.html'}
    "customHeader@case" : { templateUrl: 'custom_header1.html' }
  }
.state 'case.tabs.profile',
  url: '/profile/:profileId'
  templateUrl: 'profile/main.html'
  resolve:
    patientProfile: ($stateParams, ProfileService) ->
      ProfileService.fetch $stateParams.profileId
```

```
<a ui-sref="case.tabs.profile({profileId:profile.id})">Profile</a>
```

```
$state.go( "case.tabs.{tab}" , routeParams)
```

# UI BOOTSTRAP

- yeah, get rid of jquery
- bootstrap upgrade from 2.x to 3.x, lag on ui.bootstrap to upgrade
- angularStrap updated right away. ui.bootstrap team took forever.
- blocked us from upgrading to use the bootstrap 3.0 css

# VERSIONS

- Started project at angular 1.0.5 release
- Made conscience effort to keep up with the latest releases
- Docs were behind the curve. Comments were very helpful

- response interceptors
- animations support in 1.1.4
- broke up angular core into modules
- \$http introduced support for request/response promise chaining
- nglf was added
- controllers can now be aliased
- promises have .catch() and .finally() more like traditional try/catch
- ngSelect added 'track by' to allow more robust binding capabilities
- ngBindHtmlUnsafe to ngBindHtml need to have ng-sanitize module as a dependency
- Isolate scope only exposed to directives with scope property

# WHAT TO LOOK OUT FOR

- Good stuff in 1.3
- one-time binding <h1>{{ ::vm.title }}</h1>
- Dropping IE8
- ngMessages for forms (/i18n?)

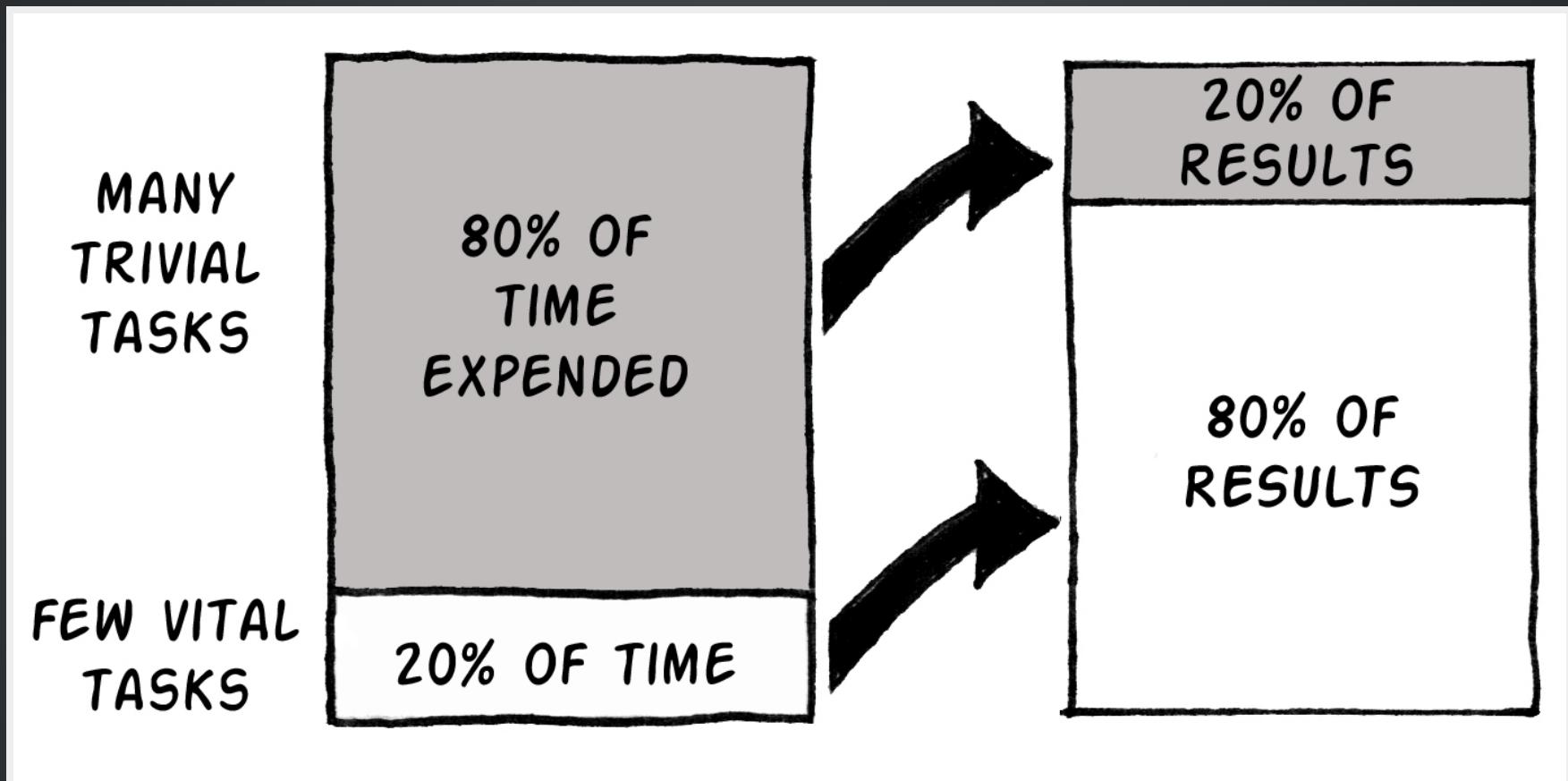
and soon...

# ANGULAR 2.0

(ZOMG)



# POLISHING UP: GOTCHAS



# GOTCHAS: CONTROLLER / SERVICE LIFECYCLE

- Started with putting state on our services
- Moved to having stateful services and stateless service
- Ran into data bleed issues. When to reset the data?

# WHERE WE WOULD LIKE TO GO

- could use an angular value to store data
- create domain models for our data
- inherit a resettable model that resets models on certain events

# GOTCHAS: UI-ROUTER NESTED STATES

```
.state 'followUpNote.tabs.treatment',
  url: '/treatment/:encounterId/:workUnitId'
  templateUrl: '/angular/scripts/treatment/readable.html'
  controller: 'ReadableTreatmentTabController'
  resolve:
    patientProfile: ($stateParams, PatientProfileService) ->
      PatientProfileService.fetch $stateParams.encounterId
.state 'followUpNote.tabs.treatment.plan',
  views: {
    "content@followUpNote.tabs.treatment": {templateUrl: '/angular/scripts/'}
}
.state 'followUpNote.tabs.treatment.note',
  views: {
    "content@followUpNote.tabs.treatment": {templateUrl: '/angular/scripts/'}
}
```

```
$scope.goToProperActivityRoute = (treatmentActivity, useRelative) ->
  relativePathPrepend = ''
  if useRelative then relativePathPrepend = '^' # TODO this kinda sucks ho
  if TreatmentActivityService.isActivityTreatmentPlan(treatmentActivity)
    $state.go(relativePathPrepend + '.plan', {encounterId: $stateParams.e
  else if TreatmentActivityService.isActivityCaseNote(treatmentActivity)
    $state.go(relativePathPrepend + '.note', {encounterId: $stateParams.e}
```

# GOTCHAS: SECURITY ROLES & PERMISSIONS

---

```
core.config ($provide) ->
  profile = angular.copy(window.myActiveProfile)
  $provide.constant('activeProfile', profile)
```

# MAYBE STILL NEED E2E TESTING?

- Best when you have a good handle on how things will be used
- That came late for us, we'd benefit from some now though
- Protractor is pretty awesome though [@ramonvictor's slides on why](#)

# GOTCHAS: OTHER JANITORIAL TASKS

- Need to move to controllerAs syntax
- Consider moving shared data to .value(), separate from .factory()
  - s
- Use of route resolver functions to pre-load more data
- **Apparently ng-min was deprecated, move to ng-annotate**
- Continue cleaning up file structure to be consistent
- move to angular 1.3

# THANK YOU FOR LISTENING!

Questions? Comments? Please give us feedback!

<https://www.surveymonkey.com/s/8WVCBK>