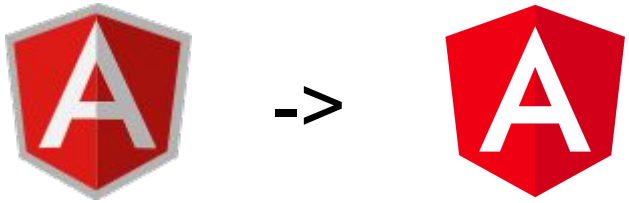


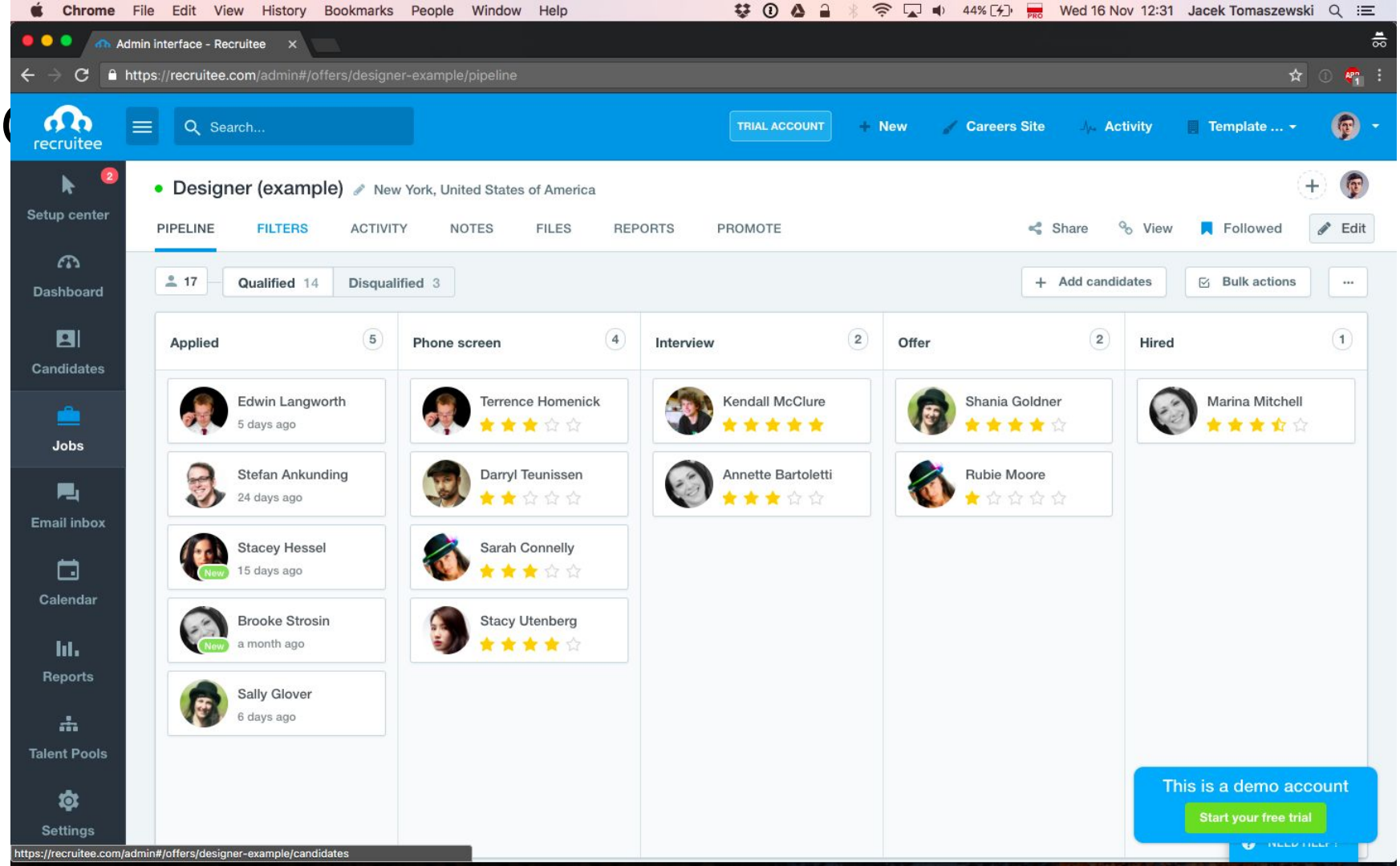
Migration from ng1 to ng2

Jack Tomaszewski (jtom.me)

AngularJS Beers @ Barcelona,
17th November 2016

Case study

Recruitee.com


Chrome File Edit View History Bookmarks People Window Help
Wed 16 Nov 12:31 Jacek Tomaszewski
Admin interface - Recruitee
https://recruitee.com/admin#/offers/designer-example/pipeline
recruitee
Search...
TRIAL ACCOUNT
New
Careers Site
Activity
Template ...
Setup center
Dashboard
Candidates
Jobs
Email inbox
Calendar
Reports
Talent Pools
Settings
Designer (example)
New York, United States of America
PIPELINE
FILTERS
ACTIVITY
NOTES
FILES
REPORTS
PROMOTE
Share
View
Followed
Edit
17
Qualified 14
Disqualified 3
Add candidates
Bulk actions
Applied 5
Edwin Langworth
5 days ago
Stefan Ankunding
24 days ago
Stacey Hessel
15 days ago
Brooke Strosin
a month ago
Sally Glover
6 days ago
Phone screen 4
Terrence Homenick
Darryl Teunissen
Sarah Connelly
Stacy Utenberg
Interview 2
Kendall McClure
Annette Bartoletti
Offer 2
Shania Goldner
Rubie Moore
Hired 1
Marina Mitchell
This is a demo account
Start your free trial
https://recruitee.com/admin#/offers/designer-example/candidates

Angular1 SPA for a SaaS platform

- ~30 different app views
- 130 components
- 83 service & factory objects

- *// Note: We used to have all components written as controllers, but with arrival of angular 1.5, we migrated all of it to `angular.component()` structure (to make the transition to ng2 easier later on)*

Angular1 problems

1. Too low performance in some of the views - digest loop taking too long
2. Tried several fixes, like:
   a. Angular bindonce syntax (`{{ ::isLoaded }}`)
   b. Ng-repeat `track by`
   c. Not rerendering an often changing view (ng-show instead of ng-if)
   d. Not rendering an inactive view (ng-if instead of ng-show)
3. … but still, the loop just takes some time and you can’t really help it.
   We need something better, more stable.

Solution

Let’s switch to a new framework! ;>

“Switching to a new framework” - Schedule

1. Set up a new framework to live at the same time with the current app
2. Slowly rewrite existing views (and implement all the new views) in the new framework
3. When all the views and old system’s parts get replaced with the new framework,
    a. Switch routing from ng1 to be handled by the new framework
    b. Bootstrap the app with the new framework only
    c. Remove the ng1 dependency completely

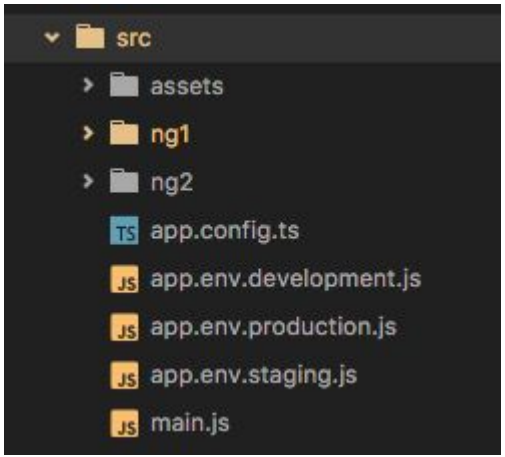
“Switching to a new framework” - but which one?

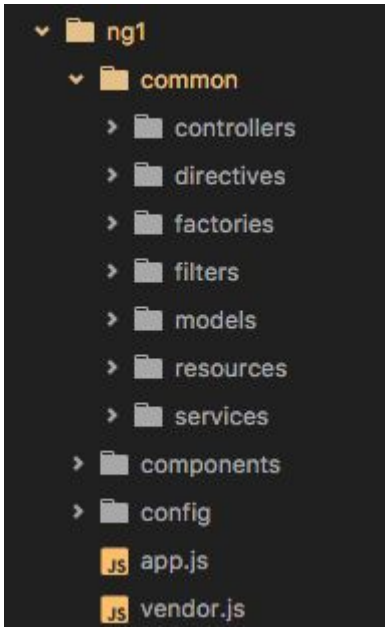
- React + Redux?
  - Great community support atm
  - The problem: how do you structure and maintain such a big SPA app in Redux?
    Splitting everything into reducers, actions and stateless components looks nice on a todo app,
    But in our case, propably looots of code would have to be written
- Angular 2?
  - There’s @angular/upgrade library helping us with the migration
  - We can, but we don’t have to, to use a single Redux/ngrx store for our app’s state
    - *// Our custom solution: use service state objects, instantied by View Components, containing both its state (as RxJS Observable) and actions that update it*

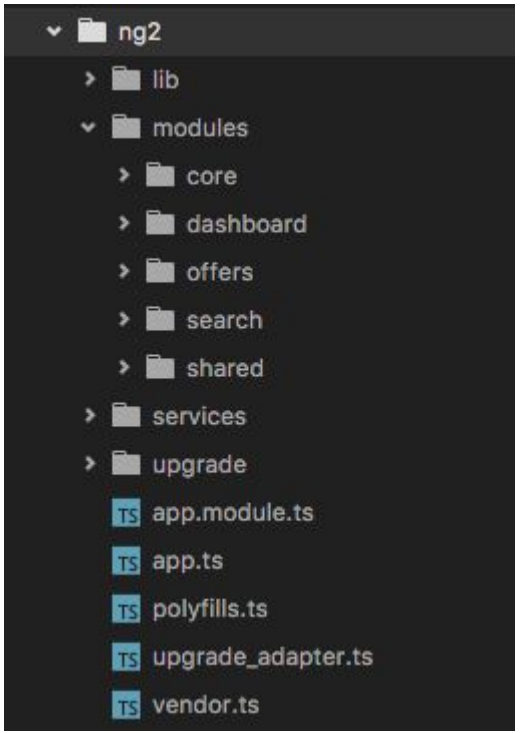
Switching from angular1 to angular2

1. Structure the app, by splitting it into ng1/ and ng2/ directories
2. Switch the app build system to webpack with typescript support (optionally)
3. Bootstrap the app with @angular/upgrade, instead of `angular.bootstrap()`
4. Upgrade some of the ng1 services to ng2
5. Downgrade some of the ng2 services to ng1
6. Create a new component in ng2 and downgrade it to ng1
7. Upgrade a ng1 component to ng2 (optionally)
   // Generally you shouldn't do it, but a lazy programmer is a good programmer ;)
8. Synchronize your data (service objects) between ng1 and ng2 (optionally)

Split the app into ng1/ and ng2/ directories







Switch to webpack with typescript support

- Remove local/bower vendor dependencies; use npm's packages instead
- Inject vendor dependencies using webpack's `require()`, f.e. `require('angular')`
  - Some hacks and polyfills required here and there, like `window.moment = require('moment')`
- Export app ENV variables to globally available `app.env.ts` file
- Example: https://gist.github.com/jtomaszewski/40a6f3e1db85528efd05ad1c83a168d7

Bootstrap the app with @angular/upgrade

- https://gist.github.com/jtomaszewski/40a6f3e1db85528efd05ad1c83a168d7

src--ng2--app.ts

```
1  import { adapter } from './upgrade_adapter';
2  import { AppModule } from './app.module';
3
4  /* tslint:disable */
5  function requireAll(r) { r.keys().forEach(r); }
6  requireAll(require.context('./upgrade/', true, /\.(js|coffee|ts)$/));
7
8  // HACK Required to make upgradeadapter work
9  // (More info in upgrade_adapter.ts)
10 adapter['ng2AppModule'] = AppModule;
11 /* tslint:enable */
12
13 (<any>window).ng2 = adapter.bootstrap(document.body, ['app'], {strictDi: false});
14 (<any>window).ng2.ready(upgradeAdapterRef => {
15   (<any>window).injector = angular.element(document.body).injector();
16 });
```

Upgrade ng1 services to ng2

```
// src/ng2/upgrade/service-upgrades.ts
import { adapter } from 'ng2/upgrade_adapter';

adapter.upgradeNg1Provider('Authorization');
adapter.upgradeNg1Provider('$location');
adapter.upgradeNg1Provider('$modal');
adapter.upgradeNg1Provider('$state');
adapter.upgradeNg1Provider('$rootScope');
```

Upgrade ng1 services to ng2

```
import { Inject, Component, ChangeDetectorRef } from '@angular/core';

@Component({
  selector: 'rt-search-content-view',
  template: (<string>require("./search-content-view.html"))
})
export class SearchContentViewComponent {
  constructor(
    @Inject('$location') private $location,
    @Inject('$rootScope') private $rootScope
  ) { }

  onFiltersChange(): void {
    this.$rootScope.$applyAsync(() => {
      this.$location.search(filters);
    });
  }
}
```

Downgrade ng2 services to ng1

```
import { adapter } from 'ng2/upgrade_adapter';
const app = angular.module('app');

import { Account } from 'ng2/services/account';
import { AppRepo } from 'ng2/services/app-repo';
import { AppStore } from 'ng2/services/app-store';
import { Helpers } from 'ng2/lib/helpers';

app.factory('ng2Account', adapter.downgradeNg2Provider(Account));
app.factory('ng2AppRepo', adapter.downgradeNg2Provider(AppRepo));
app.factory('ng2AppStore', adapter.downgradeNg2Provider(AppStore));
app.constant('ng2Helpers', Helpers);
```

Downgrade ng2 services to ng1

```
// src/ng1/common/services/account.js.es6
class Account {
  constructor($injector, ng2Account) {
    this.refreshCurrentAdmin().then((admin) => {
      ng2Account.setAuthToken(admin.authToken)
      ng2Account.setCurrentCompanyId(admin.companyId)
    });
  }

  refreshCurrentAdmin() {
    // ... return a Promise
  }
}

angular.module('app.common').service("Account", Account);
```

Create a ng2 component

```
// src/ng2/modules/dashboard/dashboard-view.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'rt-dashboard-view',
  template: require('./dashboard-view.html')
})
export class DashboardViewComponent {

}
```

Downgrade a ng2 component to ng1

```
// src/ng2/upgrade/component-downgrades.ts
import { adapter } from 'ng2/upgrade_adapter';
const app = angular.module('app');

import { DashboardViewComponent } from 'ng2/modules/dashboard/dashboard-view/dashboard-view.component';

app.directive('rtDashboardView', adapter.downgradeNg2Component(DashboardViewComponent));
```

Use the ng2 component in the ng1 app

```
diff --git a/admin_app/src/ng1/config/routes.js.coffee b/admin_app/src/ng1/config/routes.js.coffee
index 1721d81..f473bc0 100644
--- a/admin_app/src/ng1/config/routes.js.coffee
+++ b/admin_app/src/ng1/config/routes.js.coffee
@@ -65,7 +65,7 @@ hotAngular.module("app").config "routes", ($stateProvider, $urlRouterProvider) -

  $stateProvider.state "dashboard",
    url: "/dashboard"
    template: [-"<dashboard-view></dashboard-view>"-]{+"<rt-dashboard-view></rt-dashboard-view>"+}
    reloadOnSearch: false

  $stateProvider.state "activities",
```

Upgrade a ng1 component to ng2 (optionally)

```
// src/ng2/upgrade/component-upgrades.ts
import { adapter } from 'ng2/upgrade_adapter';
const app = angular.module('app');

export const ActivitiesListComponent = adapter.upgradeNg1Component('activitiesList');
export const ChartComponent = adapter.upgradeNg1Component('chart');

export const NG1_UPGRADED_COMPONENTS = [
  ActivitiesListComponent,
  ChartComponent
];
```

Upgrade a ng1 component to ng2: why?

- for ‘adapter-like’ components - to migrate old ng1 directives to new angular.component() syntax, and then use them in the ng2 app

- When we’re too lazy to migrate whole ng1 component to ng2, and want to still use some parts of the old code

```
// This basically does the same stuff as angular-chart.js directives,
// but stores it in angular.component().
// So it can be used later by ng2 app.
angular.module('app.components').component('chart', {
  bindings: {
    chartClass: "<",
    chartHeight: "<",
    chartType: '<',
    chartData: '<',
    chartLabels: '<',
    chartOptions: '<',
    chartSeries: '<',
    chartColors: '<'
  },
  templateUrl: `
    <div class="chart-outer-container">
      <canvas
        ng-class="vm.chartClass"
        height="{{vm.chartHeight}}"
        chart-base
        chart-type="vm.chartType"
        chart-data="vm.chartData"
        chart-labels="vm.chartLabels"
        chart-options="vm.chartOptions"
        chart-series="vm.chartSeries"
        chart-colours="vm.chartColors"></canvas>
    </div>
  `,
  controllerAs: 'vm',
  controller: function ChartController() {}
```

Data synchronization between ng1 and ng2:

Manually,
by $rootScope
callbacks

```
// When filters get updated, update current url and search input value
this.ctrl.filters$
  .distinctUntilChanged(null, x => JSON.stringify(x))
  .debounceTime(300)
  .subscribe(filters => {
    this.$rootScope.$applyAsync(() => {
      this.$rootScope.$emit('searchQueryChanged', filters.query);
      this.$location.search(_.pickBy(filters, _.identity));
    });
  });

// Respond to search query input in header-menu component
this.$rootScope.$on('searchQueryEntered', (event, query) => {
  const newFilters = Object.assign({}, this.ctrl.filters, {query});
  this.handleFiltersUpdate(newFilters);
  this.cd.detectChanges();
});
```

Data synchronization between ng1 and ng2:

Automatically (by callbacks, through your service objects)

```
module = hotAngular.module('app.common');
module.service "AdminModel", (Model, ng2AppRepo, ng2AppStore) ->
  new class AdminModel extends Model
    constructor: ->
      super

      # When admins collection gets loaded in ng2 app,
      # let's update the ng1's collection cache as well.
      # (ng1 -> ng2)
      ng2AppStore?.filterEventType(["admins.load_collection.success"])
        .subscribe ({data, bucket}) =>
        angular.copyData(data, @collection(bucket.payload))

    collection: ->
      @getModel(modelKey: "admins", isArray: true)

    loadCollection: ->
      @sendRequest
        modelKey: "admins"
        httpParams:
          method: "GET"
          url: "#{@baseUrl}/admins"
        transformResponse: (data) =>
          data.admins
        success: (data) ->
          # When admins collection gets loaded in ng1 app,
          # let's update the ng2's collection cache as well.
          # (ng1 -> ng2)
          ng2AppRepo?.admins.getCollection().data = angular.copyData(data)
      @collection()
```

Other notes from ng1 -> ng2 migration

- Webpack is slooow and not so easy to configure..
  Take a day off for it; feel free to use our gist code sample:
  https://gist.github.com/jtomaszewski/40a6f3e1db85528efd05ad1c83a168d7
- Use webpack hot reload
  - https://github.com/AngularClass/angular2-hmr didn't work with @angular/upgrade ;(
  - https://github.com/jtomaszewski/hot-app - our alternative -
    hot reload working with both ng2 AND ng1 app!
- RxJS is awesome
- TypeScript typings are cool

Thanks.
Any questions?

Jack Tomaszewski
Fullstack Web Developer
Freelancer / Consultant for hire: jacek@jtom.me / www.jtom.me