FRONTEND REFACTOR

Created by Marco A. Pajares / gitHub

INDEX

- Backbone
- Style (Sass)
- Jasmine
- Grunt
- Structure
- Creating a new app
- Creating a new style file
- Creating a new test file
- Creating new Doc
- Grunt task

WHY USE BACKBONE?

- We really need MVC for the front end. Traditional methods leave us with code that's too coupled, messy and incredibly hard to maintain.
- Storing data and state in the DOM is a bad idea. This started making more sense after creating apps that needed different parts of the app to be updated with the same data.
- Fat models and skinny controllers are the way to go. Workflow is simplified when business logic is taken care of by models.
- Templating is an absolute necessity. Putting HTML inside your JavaScript gives you bad karma.

see more at: tutplus or backbone tutorials

1 - INTRO TO BACKBONE

Backbone supplies structure to JavaScript-heavy 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 application over a RESTful JSON interface.

1.1 - BACKBONE ESSENTIALS

Backbone's core consists of four major classes:

- Model
- Collection
- View
- Controller

1.1.1 - MODEL

In Backbone, a model represents a singular entity -- a record in a database if you will. But there are no hard and fast rules here. From the Backbone website: Models are the heart of any JavaScript application, containing the interactive data as well as a large part of the logic surrounding it: conversions, validations, computed properties, and access control. The model merely gives you a way to read and write arbitrary properties or attributes on a data set.

1.1.2 - COLLECTION

Collections in Backbone are essentially just a collection of models. Going with our database analogy from earlier, collections are the results of a query where the results consists of a number of records [models].

1.1.3 - VIEW

A view handles two duties fundamentally:

- Listen to events thrown by the DOM and models/collections.
- Represent the application's state and data model to the user.

Creating a new app

WHY USE SASS?

Sass is a preprocessor a meta-language on top of CSS that both provides a simpler, more elegant syntax for CSS and implements various features that are useful for creating manageable stylesheets.

see more at: zivtech or Sass

Creating a new style file

FEATURES

- Syntax
- Compass
- Variables
- Mixins
- Extends

WHY USE JASMINE

Jasmine is a behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that you can easily write tests

Testing with Jasmine see more at: Jasmine

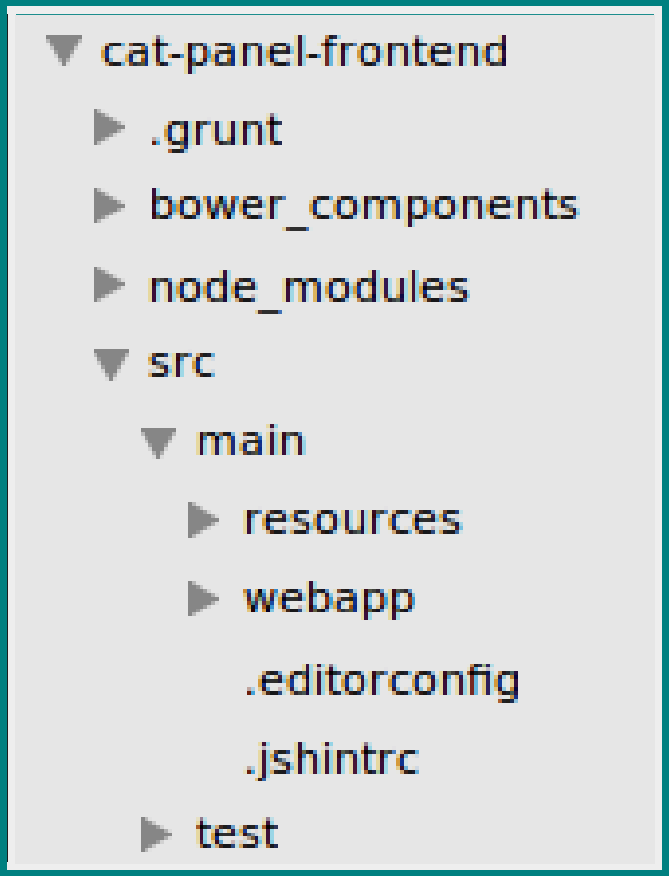
WHY USE GRUNT?

In one word: automation. The less work you have to do when performing repetitive tasks like minification, compilation, unit testing, linting, etc, the easier your job becomes. After you've configured it, a task runner can do most of that mundane work for you—and your team—with basically zero effort.

List of Grunt task available see more at: Grunt

STRUCT

From now on, we are going to try a new struct to develope new backbone apps like could be catalitycs or trends



1 - SRC/TEST

test folder has been created to store Jasmine tests

1 - SRC/MAIN/RESOURCES

resources folder has been created to store documentation and style preprocess files

Contain:

- Documentation generated with Yuidoc
- Style Sass files (not Css)

1 - SRC/MAIN/WEBAPP/JS/APPS

apps folder has been created to store Backbone applications

BACKBONE APP

In order to create or check a new Backbone application, creator must take the following steps:

- Router
- Create Struct
- BaseView
- Check Require

1 - ROUTER

inside js/router.js file, must appear URL and METHOD

```
"url":"method"
```

Important! About notation:

- URL: must be lowercase, as specific as could be, and use underscore (_) for composite words.
- Methods & Backbone Apps: must be using camelCase notation

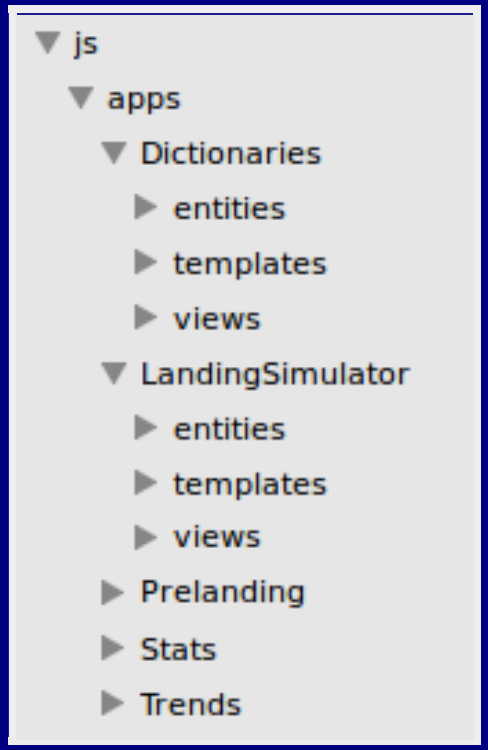
```
trends_Backbone: function() {
  this.loadBackbone('Trends');
},

// landingSimulator
landingSimulator: function() {
  this.loadBackbone('LandingSimulator');
},

prelanding_Backbone: function(){
  this.loadBackbone('Prelanding');
},
```

```
'dictionaries/:id': 'dictionaries',
'dictionaries_new': 'dictionaries_new',
'dictionaries_new/:id': 'dictionaries_new',
'ranking': 'ranking',
'conversionfunnel': 'conversionfunnel',
'catalytics': 'newstats',

// Refactor routes
'landing_simulator': 'landingSimulator',
'new_stats': 'newstats_Backbone',
'newdic': 'dictionaries_Backbone',
'trends': 'trends_Backbone',
'prelanding':'prelanding_Backbone',

//404 Always the last route because is a deafult
'*path': 'section404'
```

2 - STRUCT

Every Backbone App must contain the following folders:

- entities: models and collections are placed here
- templates: every Jade/Handlebars must be here
- views: every view must be here ATENTION! inside this folder must appear always a yourappnameBaseView.js



3 - BASEVIEW

In order to simplify create Backbone apps, inside 'router.js' there is a method called 'load backbone' that search inside 'apps/yourBackboneApp/views/yourBackboneAppBaseView.js to init the app, thats why must exist a BaseView.js file.

```
loadBackbone: function(appName){
  console.info("navigateTo: " + appName);
  var route = 'apps/' + appName + '/views/' + appName + 'BaseView';
  var self = this;
  this.changeCurrentSection(appName);
  $('#loading').css('display', 'none');

  require([route], function (requireBaseView) {
    var BaseView = new requireBaseView(self);
    $("title").html('CAT2 - ' + appName);
    $('#section-container').html( BaseView.render().el );

    // Changing style
    self.changeStyle(2);
  });

},
```

4 - REQUIREJS

When maven production task start, uglify every Js code, and use requireJs to avoid ask server multiple times for the same file. To do that, require.js needs a dependencies map, wich is located on 'js/config.js' and contain every dependencies that the app needs.

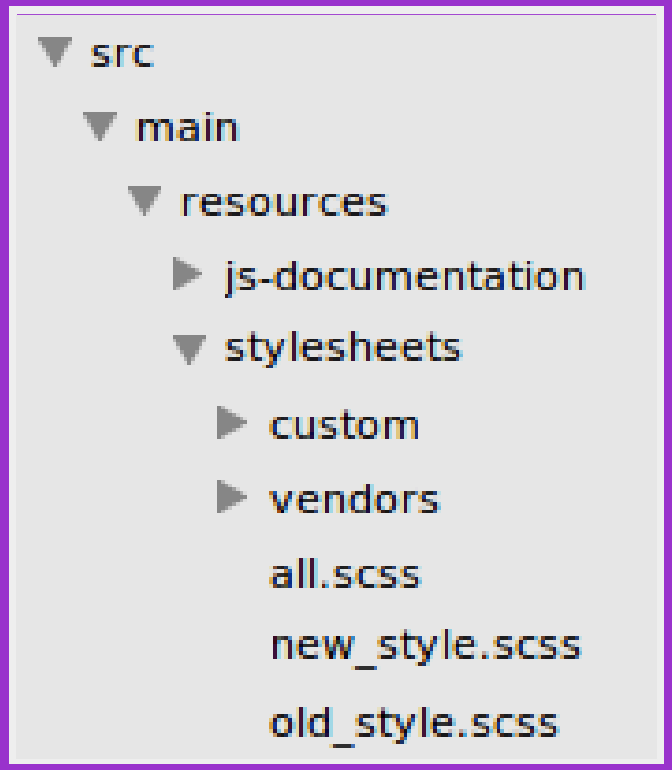
```
require.config({
  //urlArgs: "bust=${project.version}",
  waitSeconds: 0,
  // baseUrl: './js',
  paths: {
    // Libraries
    backbone: "libs/backbone-0.9.10.min",
    bootstrap: 'libs/bootstrap',
    extendbackbone: 'helpers/extend_backbone/main'
    handlebars: "libs/handlebars-1.0.rc.1.min",
    helpers: 'helpers',
    jquery: "libs/jquery-1.8.3",
    json: 'plugins/requirejs-plugins-master/json',
    modernizr: "libs/modernizr-2.6.2.min",
```

STYLE

```
grunt style
```

In order to create or check a style file, creator must take the following steps:

- Check deps
- Create a file inside '/custom'
- Link to init file



1 - CHECK DEPS

Inside 'resources/stylesheets' folder there are two folders:

- custom : wich are located every style file made by us
- vendors : wich are located every external css lib that our code need

So, if you have to import external css, place it into resources/stylesheets/vendors

2 - CREATE A FILE INSIDE /CUSTOM

Create a file inside 'resources/stylesheets/custom' folder and wrap it using a div

3 - LINK TO INIT FILE

Inside resources/stylesheets folder there are two files:

- new_sytle : listed some scss file located on custom folder and use BOOTSTRAP v3 to generate new_style.css
- old_sytle: listed some scss file located on custom folder and use BOOTSTRAP v2 to generate old_style.css

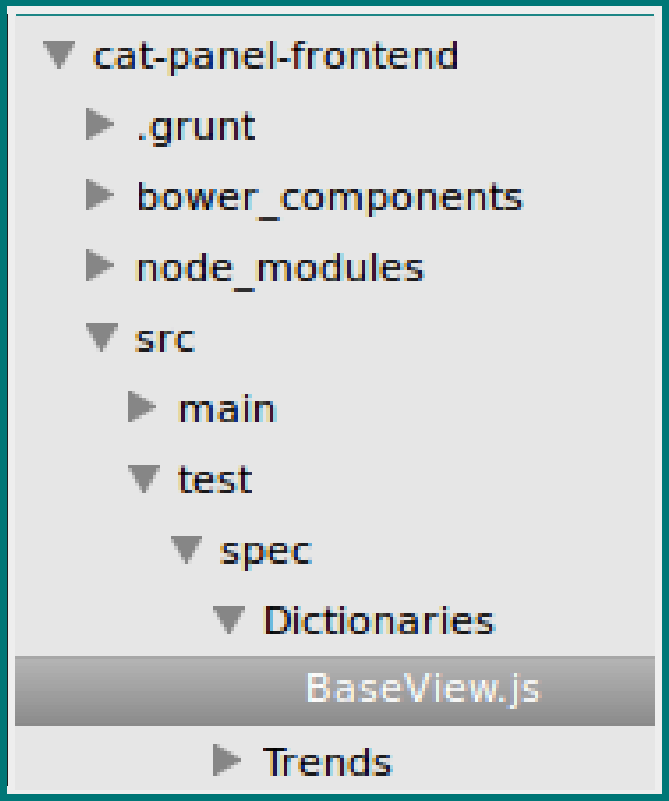
WHY TESTING CODE?

```
grunt test
```

- It is our proof/evidence that our code works properly
- Also, is a way to avoid future unexpected problems
- It runs in a controlled environment

GENERATING A SPEC

Inside 'test/spec' are listed all tthe backbone applications that right now exist, just create another folder and using require call your new aplication and start to add tests!



GENERATING A SPEC

```
define(function(require) {
  // Using 'strict mode', forbids create gloabl variables in order to use on test met

  var DictionariesBaseView = require('apps/Dictionaries/views/DictionariesBaseView');

  beforeEach(function() {
    // Test need A router
    Router = {
      changeCurrentSection : function(arg) {
       console.log(arg);
      }
    };
  });


  describe("Dictionaries", function(){

    describe("Base View", function(){

      it("View has initialize method", function() {
        var dictionariesBaseView = new DictionariesBaseView(Router);
        expect(dictionariesBaseView).toBeDefined();
      });
```

USING GRUNT MANY IMPORTANT TASK HAS BEEN CREATED

- Download external deps/libs using Bower
- Genarting Html files from jade/handlebars
- Generating style files from scss (sass)
- Evaluating JS code syntax using JsHint
- Evaluating JS code works using Jasmine
- Compress and uglify every js files in order to generate main-built using requireJs
- Generating documentation using Yuidoc
- Clean workspace

MAIN TASKS

Command	Bower	template	style	Jshint	Test	requireJs	Doc
`grunt production`	✓	✓	✓	✗	✗	✓	✓
`grunt bower`	✓	✗	✗	✗	✗	✗	✗
`grunt init`	✓	✓	✓	✗	✗	✗	✓
`grunt`	✗	✓	✓	✓	✗	✗	✓
`grunt fast`	✗	✓	✓	✗	✗	✗	✗
`grunt test`	✗	✗	✗	✓	✓	✗	✗
`grunt style`	✗	✓	✗	✗	✗	✗	✗
`grunt notest`	✗	✓	✓	✗	✗	✗	✓

DOCUMENTATION

Inside 'resources/js-documentation' exist a file named 'index.html' where you could see every comment that appears on js files.

THE END

BY MARCO A. PAJARES

Skype: marco.pajares