PMO Dashboard Technical Document

@author: Simon Tiu / Deloitte Digital Consultant

@date: 5/1/2015

@version 1.0.0 (Prototype)

**Overview / History**

The PMO Dashboard maps epics onto a timeline by their planned start and end dates. They are organized into their respective HLS parents and rollup into one HLS ticket, which is also displayed.

**Development**

PMO Dashboard is a single-page application (SPA) developed with AngularJS and visJS. Building is done with Gulp. Here are the most critical technologies used in the development of this application and their respective functions:

|  |  |
| --- | --- |
| Technology | Function |
| AngularJS | JavaScript framework, 2-way binding, HTTP calls |
| visJS | Timeline library |
| jQuery | Used lightly in controlling modals |
| Underscore | JavaScript utility functions and array manipulation |
| MomentJS | Used for time calculations and manipulations |
| Gulp | Minification and inlining |
| Bower | Package Management |

The most important piece of technology used in this application is visJS, as it controls the rendering of the timeline and most of the options are standard features provided by the library. JQuery is also used occasionally to position custom DOM elements.

Development: Installing

To install, open the PMO Dashboard zip file and cd into the directory from the terminal.

1. bower install (to install bower dependencies)
2. npm install (to install npm dependencies)
3. Open index.html

Development: Preparing for development and deploying

Since Rally can only support one compiled HTML file, we use Gulp to grab our files and deploy.

1. Minifying the code
   * gulp watch (not recommended)
     + This will watch rallyconnect.js for changes and automatically minify it for changes, piping the result to rallyconnect.min.js, which is used by index.html
   * gulp minify (recommended)
     + Since gulp watch is CPU intensive, another way workflow is to change index.html to reference rallyconnect.js directly, and minify only after development is finished by running gulp minify. The reference back to rallyconect.min.js will need to be made manually (or another gulp task can be written for this). This will allow the developer to debug inside of chrome.
2. Inlining all of the code
   * gulp inline
     + To inline all of the JavaScript and CSS libraries
     + Note that font awesome is pulled through CDN <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.css">
3. Copy and paste code found in deploy/index.html into Rally.
   * As an alternate to steps 1 and 2, you may just run gulp and it will perform gulp minify and gulp inline sequentially)

Development: Overview of the code

On a high-level, the code pulls data from rally and puts them into various “item” object, which are then displayed on the topic based on attributes of those items.

1. Pull the data from Rally: releases, epics, hls, iterations
2. Iterate through the epic data from Rally to create epic items and release items (extrapolated from the Epics)
3. Iteration through hls data to create groups (to organize the epic items)
4. Extrapolate sprint data from ELA and make release lines
5. Render the timeline

The most useful tool in dealing with the timeline is to consult the vis.js documentation, which can be found here: <http://visjs.org/docs/index.html>