**Abstract**

This guide will assume that you’ve already read the config/installation guide.

This guide will teach you about some of the frameworks used, in particular AngularJS.

The webapp is considered a Single Page Application (SPA),the user does not need to navigate around.

**File Structure of public\_html**

The API folder contains API PHP-files that return JSON-formatted data depending on the requests and file.

The app folder contains our AngularJS files (no framework js stuff)

The assets folder contains data, and images for each test. See config guide for exact explanation.

The css folder contains a file for basic css changes (no framework css)

The include folder has PHP files called by the API folder that is often shared by many API calls, such as DB login credentials.

The index.php contains the basic webpage that also links to the dependencies and our AngularJS and CSS files.

**AngularJS**

AngularJS (hereby AJS) is a JavaScript framework that attempts to make web development a bit more like Model-View-Controller (MVC). There are several different capabilities of AJS.

The most notable is the separation of responsibilities of files. There are four different files that are most notable: Controllers, Directives, Services, and Templates.

* Template
  + A template is a HTML file that is used as the View in MVC
  + The HTML file can contain an entire application or a small piece of code that may be repeated.
  + The template file can be used in other templates as well
  + AngularJS can model inputs and selectively display and update models. See <http://www.w3schools.com/angular/angular_model.asp> for a more in-depth explanation
* Controller
  + The controller communicates with a view and provides the logic and models.
  + The controller can update variables that have been modeled. When the variables change, the view also changes. This occurs solely in the controller.
  + In a perfect world, the controller should be rather slim as it should just have assigners, ifs, and simple logic that dictates view and nothing further.
  + The models are also initiated in the controller.
* Directive
  + Directives control the flow of views and controllers. They connect specific templates to specific controllers. They tell what kind of view the template is (an html attribute, element, or class as well as have the capacity to pass data between views if necessary
* Service
  + Services are used as a means to eliminate bloat in the controller and to provide a single file to handle the request and collection of an API call.
  + You inject the Service into a controller, and call the service from inside the controller (on load or perhaps a button click or select or change).
    - The service will have a list of functions that can be called.
  + In order to call a GET request, inside a service we can inject the $http service. The $http service allows us to GET and POST to our PHP files.
    - This also handles the retrieval of the files, both a successful retrieval and when the PHP file throws an error.

**Bootstrap**

Bootstrap is for responsive websites - the websites should work on both mobile devices and desktops.

It provides a host of view-improving features; much too much to be written in a guide. I highly recommend viewing bootstrap’s site for demonstrations and w3’s site for additional assistance.

<http://getbootstrap.com/>

<http://www.w3schools.com/bootstrap/>

**Google Charts**

The existing documentation for google charts is fairly comprehensive and a bit technical but it is located: <https://developers.google.com/chart/>

**Adding a new test**

1. Add a new row in the *events* table - it’ll need a name, some comments, a directory name, and whether to make it active or not.
2. Go into public\_html/assets/<the directory name> and create three folders as described in the config file; a “data”, “imgs” and “resized” folder.
   1. Again, the csv file in the “data” folder needs to be as described in the config file
   2. The images in the other two folders also need to be named properly as described in the config file.

**Other**

Tabs as indentation is the devil’s work. Use spaces!

Two spaces for indentation