**Citrya Editor Documentation**

**1) Web app structure:**

i) Understanding and working with the code will require the knowledge of the following:

* **JavaScript**
* **AngularJS** (Source Code) -

<https://github.com/angular/angular.js>

* **Using AngularJS Libraries and integrating them.**
* **Angular Gridster** (Library for draggable widgets) -

<https://github.com/ManifestWebDesign/angular-gridster>

* **Angular Dragdrop** (Library for dragging and dropping widgets)-

<https://github.com/codef0rmer/angular-dragdrop>

* **Angular Flexslider and Flexslider** (Library for Slider) -

<https://github.com/thenikso/angular-flexslider>

<https://github.com/woothemes/FlexSlider>

* **Angular Quill and Quill** (Library for TextEditor) -

<https://github.com/axilleasiv/angular-quill>

<https://github.com/quilljs/quill/>

* **Jquery** (JavaScript lighweight library) -

<http://www.w3schools.com/jquery/jquery_intro.asp>

<https://github.com/jquery/jquery>

* **Jquery UI** (Library for user interface interactions, effects, widgets, and themes) -

<https://github.com/jquery/jquery-ui>

* **Font-Awesome** (Library for icons) -

<https://github.com/FortAwesome/Font-Awesome>

* **NgStorage and Angular Local Storage** (**Optional** Library for local storage and session storage variable for saving a sessional work in Editor) -

<https://github.com/grevory/angular-local-storage>

<https://github.com/gsklee/ngStorage>

ii) We can add more libraries and integrate them with angular. **We can use *bower* to install an angular library under the *bower\_components/* folder.**

iii) To install bower follow the commands as below :-

*sudo apt-get install nodejs*

*sudo apt-get install npm*

*sudo npm install bower -g*

*sudo bower install 'package-name'*

**2) Adding a widget of type 'newWidget'.**

* Add widget to $scope.widget in ***app/script.js*** by adding the widget type as 'newWidget' and widget initial data (empty list or empty string).
* Create folder of name **'newWidget'** in ***app/templates***.
* Create three files in the folder newWidget. Namely: **'placeholder.html', 'settings.html', 'template.html'**
  + **placeholder.html:** Add the glyphicon (image which comes before we add any content to the widget) related to the newWidget or any other placeholder needed as initial view of the widget. (to find the glyphicon, go to [www.glyphicon.com](http://www.glyphicon.com/))
    - *<i class="glyphicon {{name of the glyphicon}} glyphicon-ph" ></i></br><span class="placeholder">{{Placeholder text}}</span>*
  + **settings.html:** Create a form to input the required data from the user and store it in form.data or other variables that are used by functions in app/script.js to give appropriate output.
  + **template.html:** Output of the widget that is seen in the app as a module. Configure according to requirement by adding the required html tag you want to showcase using output from processing the data stored in form.data using settings.html.
* Add functions to ***script.js*** in the default controller using the following syntax:
  + *$scope.{{function name}}=function(variables){ function body };*
* Add ‘newWidget’ button in ***app/view.html*** by adding it to the button table such that there are at the most two buttons in a row. Use the following syntax:
  + *<td><a class=" button button-primary btn-new" data-drag="true" data-jqyoui-options="{revert: 'invalid', helper: 'clone'}" ng-model="widgets[{{index of newWidget in $scope.widget}}] " jqyoui-draggable="{placeholder: 'keep', deepCopy: true}" ng-click="pushWidget(widgets[{{ index of newWidget in $scope.widget }}])" title="newWidget"><i class="glyphicon {{ name of the glyphicon }} glyphicon-lg"></i><br/><small>newWidget</small></a></td>*
* There is a sample app that can be used to understand how to add a widget.

**3) Description about AngularJS Controllers, Views, Directives, Filters and Scopes:**

* **Controllers:**
  + Looking at the starting of ***app/script.js*** file*angular.module('app')*

*.controller('DashboardCtrl', function($scope) {  
 //code  
}); and so on*

1. The AngularJS application is defined by  ng-app='app" here.
2. The **ng-controller='DashboardCtrl'** attribute is an AngularJs directive. It defines a controller.
3. The **DashboardCtrl** function is a JavaScript function.
4. Look more from <http://www.w3schools.com/angular/angular_controllers.asp>
   * Now there are three types of controllers in our source code:
5. **DashboardCtrl:** This is the main controller of the code. The **whole application scopes and attributes or properties** can be defined in this controller. **Defining the new widgets** and the **function for the dialog boxes which will open after clicking the widgets** should be added in this controller. The function anything related to the **properties of the widgets** such as resizing and dragging and dropping should be added in this controller.
6. **CustomWidgetCtrl:** This is the general controller for adding functions for opening settings view page ***(app/templates/'newWidget'/settings.html)*** for each widget.
7. **WidgetSettingsCtrl:** This is the controller for adding functions for the properties and attributes of the settings view page ***(app/templates/'newWidget'/settings.html)*** and template view page ***(app/templates/'newWidget'/template.html).***

* **Views:**
  + ***/index.html*:**This is the HTML page where we integrate all the angular libraries used and provide a **route** to the main controller i.e. **DashboardCtrl** and the **templateURL** i.e. ***app/view.html.***
  + ***app/view.html*:**This is the HTML page where we add the views for each widget. Also we integrate the controllers which are **CustomWidgetCtrl** and **WidgetSettingsCtrl.**
  + ***app/script.js*:** This is the JS file where we have put all the **controllers and their respective functions and directives and filters** *(will be explained later)*.
* **Directives:**
  + A directive is something that introduces new syntax. Directives are markers on a **DOM element** *(DIV, HTML, BODY* ***element*** *on a page)* which attach a special behavior to it. For example, static HTML does not know how to resize each widget accordingly compared to the height of the content inside it. To teach HTML this new syntax we need a directive. The directive will somehow create an element that behaves like a dynamic height checker.
  + Look more from <https://www.sitepoint.com/practical-guide-angularjs-directives/> and <http://www.w3schools.com/angular/angular_directives.asp>
  + Some directives used:
* **gridsterDynamicHeight:** Adds dynamic height function to the gridster to each widget. The dynamic height of widget is implemented through this directive. For this directive, we took help from the **github issue of the angular-gridster** <https://github.com/ManifestWebDesign/angular-gridster/issues/220>
* **htmldiv:** We are using this directive for the Output widget to convert the widget editor into **'HTML'** code.
* **Filters:**
  + Filters can be added in AngularJS to format data.
  + Look more from <http://www.w3schools.com/angular/angular_filters.asp>
  + Some filters used:
* **unsafe:** Formatting a variable to HTML value.
* **trustURL:** Formatting a variable to URL form. **Getting used in Image widget.** Look more from <http://stackoverflow.com/questions/20049261/sce-trustasresourceurl-globally>
* **getYoutubeID:** Getting the YoutubeID from the variable entered in the V**ideo widget** and formatting it to URL form. Look more from <http://stackoverflow.com/questions/23434635/angular-how-to-get-the-video-id-from-a-full-youtube-url>
* **columns:** Being used in the **Output widget.**
* **Scope:**
  + In an AngularJS application, the controller and view share an object called a scope; this object is at the core of its amazing two-way data binding. The controller sets properties on the scope, and the view binds to those properties. AngularJS takes responsibility for keeping the two in sync.
  + Look more from <http://blog.carbonfive.com/2014/02/11/angularjs-scopes-an-introduction/>

**4) Some important functions and variables in app/script.js:**

**a) DashboardCtrl:**

* $scope.widgets: A dictionary of widget with widget type and the initial data required for the widget. Used to add or remove widgets.
* $scope.dashboard: A dictionary of the widget modules currently in the workspace. A default widget module can be placed if required.
* $scope.gridsterOptions: A dictionary of options used to customize gridster. For more information, go to: <http://manifestwebdesign.github.io/angular-gridster/>
* $scope.cssList: Dictionary of the links to css files used in the working of the widgets. May be added here or in the template of the widget *(advised to add in the template)*.
* $scope.jsList: Dictionary of the links to js scripts used in the working of the widgets. May be added here or in the template of the widget *(advised to add in the template)*.
* $scope.pushWidget: Adds a new widget module to the workspace by pushing it to $scope.dashboard along with all the options.
* $scope.beforeDrop: A function for opening the settings modal before dropping the widgets onto mobile area *(Optional)*.
* **Slider Functions:**
  + $scope.next: Changes to the next slide
  + $scope.previous: Changes to the previous slide
  + $scope.onReadySwiper ///////write
  + $scope.addSlide*(in WidgetSettingsCrtl Controller)*: Used to add a slide to the slider widget module by pushing the new slide to the widget module data.
  + $scope.$watch: Watching the change in the specified variable in the $scope.$watch.
* $scope.compileOutput: Gives the output of the workspace by generating the html script for all the widget modules. Require knowledge of html DOM to understand and make changes to.
* $scope.download: Downloads the project so that it can be modified and uploaded later. Uses blob, MSBlobBuilder and DOM.
* $scope.openModal: Opens a setting modal window by taking the modal template from the settings.html file in ***app/templates/'newWidget'/***
* $scope.saveDashboardLayout: A function for opening the Save settings modal on clicking the Save button.
* $scope.openLayout: A function for opening the Open settings modal on clicking the Open button.
* $scope.clear: Clears the workspace by removing all widget modules.

**b) CustomWidgetCtrl:**

* $scope.openSettings: Open the settings modal for each widget.
* $scope.remove: Used to remove any individual widget.

**c)** **WidgetSettingsCtrl:**

* $scope.jsonToLocalStorage: Function that can be used to store files in the browser memory to enhance user experience. Not in use currently.
* $scope.addFile: Used with jsonToLocalStorage to add file to the browser memory. Can be used to enhance user experience.
* $scope.loadFile: Uploads file from the desktop to load existing project.
* $scope.remove: Remove a single widget module.
* $scope.dismiss: Close/dismiss the modal.
* $scope.check: Check if at least one of the options is selected *(helper function)*.
* $scope.reset: Reset the widget module to default.

**END**