Simple AJS app {{step-by-step}} approach

1. Define structure of the project – one example could be
   1. Folder js for controllers, services, filters, directives, etc.
   2. Folder view for HTML templates
   3. index.html for entry point for the project
      1. include basic AJS libraries like

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>  
<script

src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular-route.js"></script>

1. Set directive ng-app in index.html on <html> or <body> i.e. root level
2. If you assign name on directive ng-app like ng-app=”app” you have to define module named “app”
   1. Include it in index.html

<script src="js/app.js"></script>

with implementation like

angular.module('app', [  
 'app.controllers',  
 'app.directives',  
 'ngRoute'  
 ])

The dependencies in brackets could vary.

1. Define route / $routeProvider

(**function** () {  
 'use strict';  
  
 angular.module('app', [  
 'app.controllers',  
 'app.directives',  
 'ngRoute'  
 ])  
  
 .config(**function** ($routeProvider, $locationProvider) {  
  
 $routeProvider  
 .when('/hello1', {  
 templateUrl: 'view/hello1.html',  
 controller: 'Hello1Controller',  
 controllerAs: 'hello1'  
 })  
 .when('/hello2', {  
 templateUrl: 'view/hello2.html',  
 controller: 'Hello2Controller',  
 controllerAs: 'hello2'  
 })  
 .when('/template', {  
 templateUrl: 'view/template.html',  
 controller: 'TempController',  
 controllerAs: 'template'  
 })  
 .otherwise({  
 redirectTo: '/'  
 });  
  
 $locationProvider.html5Mode(**true**);  
 });  
})();

1. Modifying DOM
   1. There are several ways to include HTML piece:
      1. Using ng-view like <div ng-view=""></div>
         1. Create a HTML component like hello1.html

<div ng-controller="app.controllers as hello">  
 <input ng-model="hello.greeting.text">  
  
 <p>{{hello.greeting.text}}, World</p>  
</div>

* + - 1. Define entry point in the route like using controllerAs controller syntax

.when('/hello1', {  
 templateUrl: 'view/hello1.html',  
 controller: 'Hello1Controller',  
 controllerAs: 'hello1'  
})

* + - 1. Define controller in folder js like controllers.js
      2. Create controller implementation like using var vm = this; instead of $scope.

Use a capture variable for this when using the controllerAs syntax. Choose a consistent variable name such as vm, which stands for ViewModel.

Why?: The this keyword is contextual and when used within a function inside a controller may change its context. Capturing the context of this avoids encountering this problem.

(**function** () {  
 'use strict';  
  
 angular.module('app.controllers', [])  
 .controller('Hello1Controller', Hello1Controller);  
  
  
 **function** Hello1Controller() {  
 **var** vm = **this**;  
 vm.greeting = {text: 'Hello 1 ...'};  
 // use $location for something good here...  
  
 console.log('Hello1Controller', vm);  
  
 }  
  
})();

1. Include it in index.html

<script src="js/controllers.js"></script>

* + 1. Using customer tags like <my-hello>…..</my-hello>
       1. Create a file js/directives.js
       2. Define directive like
       3. Include it in index.html

<script src="js/directives.js"></script>

This kind of DOM modification can be made through Directives. Example for such directive is myHello. It uses also syntax controllerAs, bindToController and controller. The controller implementation is in the directive as internal function.

(**function**() {  
 'use strict';  
  
 angular.module('app.directives', [])  
 .directive('myHello', myHello);  
  
 **function** myHello() {  
 **function** TempController() {  
 **var** vm = **this**;  
 vm.greeting = {text: 'Template ...'};  
 // use $location for something good here...  
  
 console.log('TempController', vm);  
  
 }  
 **return** {  
 restrict: 'EA',  
 scope: {  
 greeting: '='  
 },  
 templateUrl: 'view/template.html',  
 controllerAs: "template",  
 bindToController: **true**,  
 controller: TempController  
 };  
 }  
  
})();