1. Git provides with a special feature called “git clone branch version”.

Coz of that, I cud easily copy down the app structure from Ray’s github, in which just the mandatory files were added.

1. Package.json file contains starting basic info about our app
2. Readme file contains initial instructions, the commands to be executed.
3. .gitignore file – tells the git not to track certain files or folders in app
4. By running “npm install” we can install all our dependencies.
5. Make sure, if u have copied the file structure from some other user, and try to put that code into ur own repository, then don’t change the “.git” folder of ur repository. That is the unique identity of ur repository.
6. Whenever u put “ng-include” in ur index.html file, that means u want to include an entire file/template into ur main view. Whatever has been written inside that template will be shown there.
7. CREATING “modules” AND “routes” IS THE FIRST THING WE GONNA DO IN OUR APPLICATION.
8. So far, we're using an app without a name. That means that the application is not controlled by any particular JavaScript file. To set up a real Angular application, we should structure things a little bit different. We need to create something called a module to handle what we want the app to do. A module is a container for the different parts of our application. Once you create the module, we can start working on the pieces that control what our app is going to do.The module needs a name and we put it right here in this ng-app directive.
9. (a) Most of the time, we create our module in “app.js” file.

(b) We also have to include the path of this file, inside our in “index.html” file.

(c) Now by adding “ng-app” to <body> tag, the portion inside, wud be controlled by our module, going forward.

1. While we created our module, we also have to pass dependencies. Foremost, it’s [‘ngRoute’] module.
2. Note: To utilize ‘ngRoute’ module, we firstly have to add “angular-route.js” file inside our “index.html file.
3. Note: The name of the module is actually the inside one.

For e.g : var myApp = angular.module(‘myApp’, []);

<body ng-app=”myApp”>

1. The second thing that we define inside our “app.js” file is “routing”.
2. Why do we need $routeProvider service ?

Ans: By default, it'll look at a hashtag or something right after a hashtag and pretend like it's a page. So we need to set that up. Let's go back into our application and we're gonna use something called the routeProvider service.  so we'll say myApp.config and then I'll pass it along in quotations the routeProvider service as well as a function literal. This function is also gonna get a copy or some of the information from routeProvider and then comes our function part.

1. We put $routeProvider via “.config”. Basically, configuring means arrange or put together in a particular form.

ngView is a directive that complements the [$route](https://docs.angularjs.org/api/ngRoute/service/$route) service by including the rendered template of the current route into the main layout (index.html) file. Every time the current route changes, the included view changes with it according to the configuration of the $routeservice.

Requires the [ngRoute](https://docs.angularjs.org/api/ngRoute) module to be installed.

1. **To create secondary view in our app, we did:**
2. **Define routing in app.js such that a particular template is loaded upon hitting a specific URL.**
3. **Created that particular template (inside “views” folder)**
4. **Go into “index.html” , created a <div> and wrote “ng-view” inside it**

**e.g: <div ng-view> </div>**

**The whole of the template will be loaded inside this <div>.**

1. We can use form validations like $invalid, ng-pristine, ng-dirty, ng-valid.
2. We now start building controllers. In angular world, we have
3. Views
4. Model (or data)

To relate both of these, we need to write javascript code that resides in controllers.

1. In our app, we decide to handle both the “login” and “register” views via the same controller. Coz both have pretty much the same functionality.
2. To associate controllers to our template view, we define

controller: ‘RegistrationController’

inside routing. *Just the name of controller, not the path*.

Then, we define a path of controller js file inside our “index.html”

And finally, we wrote code inside that controller.

1. Did u notice that, inside “index.html” file, we didn’t have to give the path for our template views, but had to give the path for our controllers that are associated to that views.

Because, controllers are javascript files whose reference need to be given inside <script> tag, in “index,html”.

We only give path reference of html files, when we try to include them via “ng-include” command.

1. Angular has some wonderful built-in services. One of them, we have used is “$location”.
2. Suppose we want to use a text which we have added inside an <input> field, such that it can be accessed inside controller, then we need to provide “ng-model” to it. Then only that variable with the name inside “ng-model” will get availed to the controller.
3. While filling the form, u can make the “SUBMIT” button disabled by adding ng-disabled=” myform.$invalid” to the <button> tag.
4. While accessing firebase, found that quite a few functions and services name have been deprecated. New ways of including those functions have come. So, refer the documentation of firebase.
5. STEPS FOR PUSHING THE DATA INTO FIREBASE :
6. Create a reference of Firebase object :

*var ref = new Firebase("https://sarthakangularapp.firebaseio.com/meetings");*

1. Inside your chosen function, use the push method as follows:

*ref.push({*

*name: "Sarthak",*

*date: "29 Feb"*

*});*

1. Conceptual Talk: Firebase data, that we create, has a 3-way data binding, that is, between:
2. The data we type in addMeeting input box
3. Gets automatically updated to the view below
4. Also to firebase database
5. STEPS FOR PUSHING THE DATA INTO FIREBASE

Each data entry is stored inside firebase via a key. To delete that entry, we need to find that key. To do this, follow as:

1. Inside views file…. instead of writing :

*ng-repeat="meeting in meetings"*

write:

*ng-repeat="(key,meeting) in meetings"*

1. Utilize that key inside firebaseRef.remove() function, as follows:

*firebaseRef.remove(key);*

1. Note: For removing a particular node from Firebase database:

* Suppose, I want to remove a node with a defined key inside “meetings” node
* So, this is my original reference:

*var ref = new Firebase("https://sarthakangularapp.firebaseio.com/meetings");*

* Suppose the “key” returned to me is = “four”.
* Now, I will append this key to the URL so that it becomes:

[*https://sarthakangularapp.firebaseio.com/meetings/four*](https://sarthakangularapp.firebaseio.com/meetings/four)

* Now, I can directly remove by calling the function “remove()”, and the entire data inside this node will be deleted.
* To do this, I need to write the following code:

*var newRef = ref.child(key);*

*newRef.remove();*

1. Firebase has a beautiful feature called Login/Registration Authentication.

To utilize this feature, follow these steps:

* Go to Firebase-> Login & Auth-> Create a new User with Email and Password.
* Note: Don’t worry, it’s not the new credentials with which u have to login into ur firebase account.
* It’s actually the database entry, that a fictional user comes into ur database, who, u can use for ur login functionality.
* Now to test the login authentication functionality provided by Firebase, go into ur controller -> Include service $firebaseAuth.
* Create a ref of ur home URL:

var ref = new Firebase("https://sarthakangularapp.firebaseio.com/meetings");

* Utilize $firebaseAuth service and store the value into new variable:

var auth = $firebaseAuth(ref);

* Now dump your data inside:

auth.$authWithPassword({ …email…. password… });

* Now whatever email and password u have dumped, firebase will check against the original user email and password, and will return promise for either a success or an error.