

# ANGULAR JS

(Getting-Started, Walkthroughs & Dev Insights)

This documentation is an inspiration and adaption from various web sites and books. All the references are mentioned in the reference page.

This document on AngularJS is an Individual work to promote and help developers learning AngularJS.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.



This document can be shared, edited and re-produced with or without permission of the Author.

Sale of this document either electronically or other ways is not permissible.

# INTRO

Author : Chandrahas Dodda & Keerthana Kammari

This documentation is purely for beginners, who are willing to learn Angular 1. To continue with this document, we assume readers have basic knowledge of HTML, CSS and JavaScript.

Having knowledge on JavaScript functions, function types and closures will help in understanding AngularJS.

This document helps learning AngularJS version 1, however learning version 2 requires further knowledge of ECMA Script 6 and TypeScript by Microsoft.

All the code snippets shown in this document are available at github for download. Readers can download, share, edit and re-produce freely.

**Github link:** <https://github.com/chandrahas-reddy/AngularJS-Docs-Insights.git>  
<https://github.com/keerthana-kammari/AngularJS-Documentation>

This document contains various developer notes under the name “dev note” which provide some facts and interesting insights.

Though the documentation is prepared with extreme care some mistakes/errors are inevitable. Please email the mistakes and help in contributing to better versions of this documentation.

**Email:** chandrahas95@gmail.com , keerthanakammari@outlook.com

# **CONTENTS**

## **Part-1: Getting Ready**

Chapter-1: Getting Ready

Chapter-2: MVC

Chapter-3: First Application

Chapter-4: Expressions

Chapter-5: Directives

Chapter-6: Data Binding

Chapter-7: Controllers & Scope

Chapter-8: Filters

Chapter-9: Services

Chapter-10: Routing

Chapter-11: Complete Application

## CHAPTER-1 Getting Ready

### **Angular JS:**

AngularJS (commonly referred to as "Angular" or "Angular.js") is a complete JavaScript-based open-source front-end web application framework.

It can be added to an HTML page with a `<script>` tag.

It extends HTML DOM with additional attributes and makes it more responsive to user actions.

AngularJS applications are built around a design pattern called Model-View-Controller (MVC). (\*MVC is explained in Chapter-2)

### **\*Dev Note:**

Before starting AngularJS, one should be familiar with the basics of web development, have an understanding of how HTML and CSS work, and ideally have a hands-on knowledge of JavaScript.

### **License & Versions:**

AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache license version 2.0.

This documentation is made on Angular 1.x.

AngularJS has evolved recently to Angular 2. This version of angular is developed using ECMA Script 6 and TypeScript.

### **\*Dev Note:**

This documentation does not help learning Angular2, although the core principles of Angular still persists.

## Environment Setup:

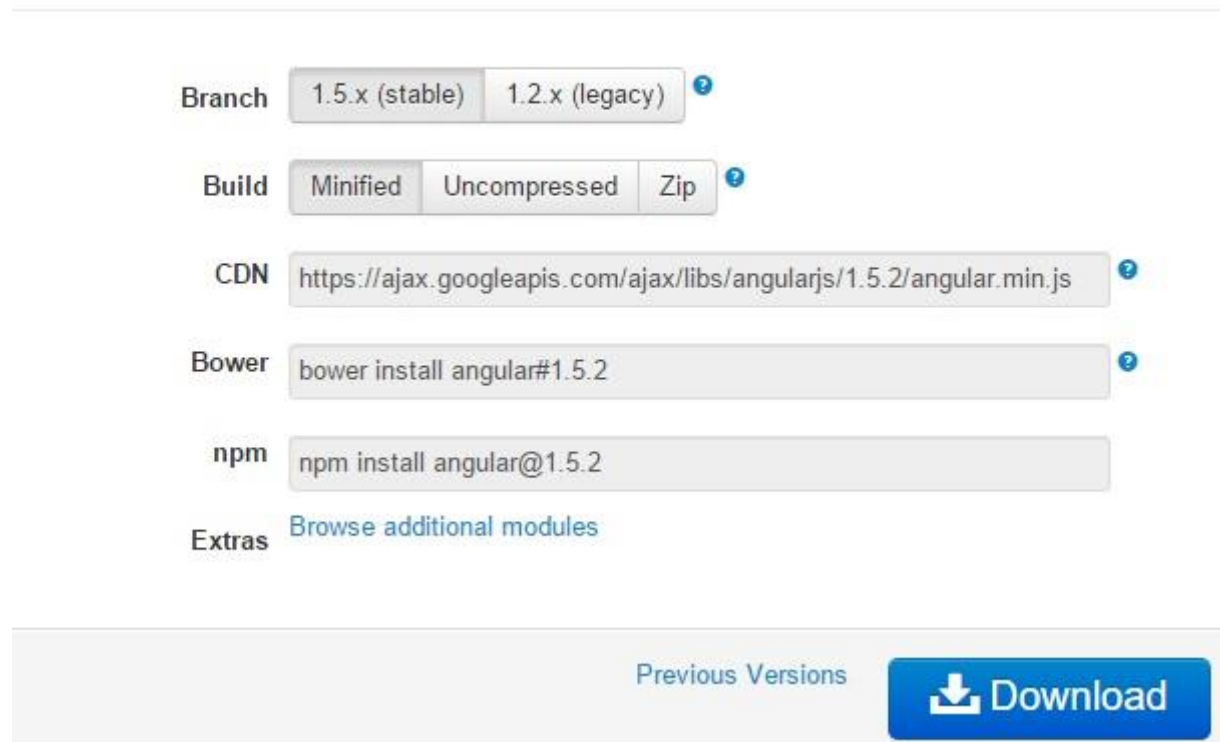
Angular JS can be downloaded from <https://angularjs.org>

Developing applications with angular is super easy and simple, we don't need any heavy IDE's to develop angular web applications. A simple text or web editor like Brackets, Sublime, Visual Studio Code are enough to develop.

(\*Code snippets and examples in this documentation are coded using Brackets editor)

Below is the snapshot showing downloads section of angularjs.org

### Download AngularJS



The screenshot shows the 'Download AngularJS' section of the AngularJS website. It features several interactive elements:

- Branch:** Two buttons labeled '1.5.x (stable)' and '1.2.x (legacy)', with a help icon (?) next to the second button.
- Build:** Three buttons labeled 'Minified', 'Uncompressed', and 'Zip', with a help icon (?) next to the 'Zip' button.
- CDN:** A text input field containing the URL 'https://ajax.googleapis.com/ajax/libs/angularjs/1.5.2/angular.min.js' and a help icon (?) to its right.
- Bower:** A text input field containing the command 'bower install angular#1.5.2' and a help icon (?) to its right.
- npm:** A text input field containing the command 'npm install angular@1.5.2'.
- Extras:** A link labeled 'Browse additional modules'.
- Footer:** A 'Previous Versions' link and a large blue 'Download' button with a download icon.

For developing angular applications we also require to download jQuery from <http://jquery.com/download/> and add it into our code project.

(jQuery is a simple JavaScript library which contributes to write less – do more motto. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.)

## CHAPTER-2 MVC

AngularJS applications are built around a design pattern called Model-View-Controller (MVC), which places an emphasis on creating applications that are

- **Extendable:** It is easy to figure out how even a complex AngularJS app works once you understand the basics—that means one can easily enhance applications to create new and useful features for users.
- **Maintainable:** AngularJS apps are easy to debug and fix, which means that long-term maintenance is simplified.
- **Testable:** AngularJS has good support for unit and end-to-end testing, meaning that one can find and fix defects before your users do.
- **Standardized:** AngularJS builds on the innate capabilities of the web browser without getting in your way, allowing you to create standards-compliant web apps that take advantage of the latest features (such as HTML5 APIs) and popular tools and frameworks.

Model-View –Controller as it is popularly called, is a software design pattern for developing web applications.

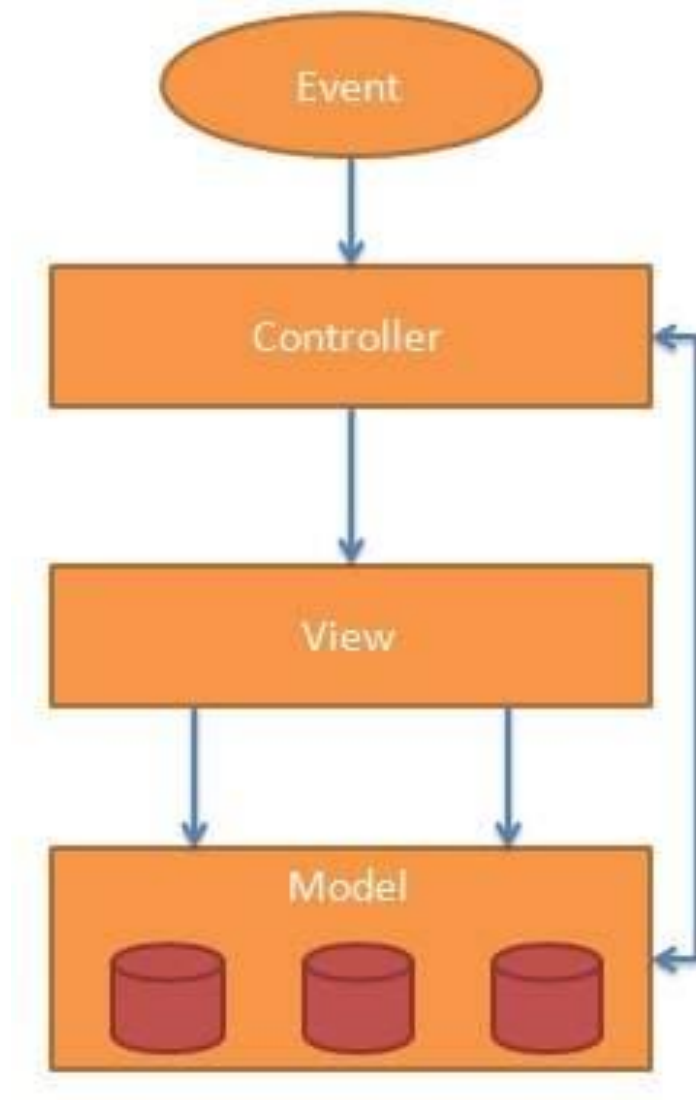
**Model:** It is the lowest level of the pattern responsible for maintaining data.

**View:** It is responsible for displaying all or a portion of the data to the user.

**Controller:** It is a software Code that controls the interactions between the Model and View.

MVC is popular because it isolates the application logic from the user interface layer and supports separation of concerns. The controller receives all requests for the application and then works with the model to prepare any data needed by the view. The view then uses the data prepared by the controller to generate a final presentable response.

## MVC Architecture



- The model is responsible for managing application data. It responds to the request from view and to the instructions from controller to update itself.
- A presentation of data in a particular format, triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
- The controller responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.



## CHAPTER-3 FIRST APPLICATION

The below application describes the basic structure of an angular application.

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4
5      <!--scripts-->
6      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.min.js">
7      </script>
8  </head>
9  <body ng-app>
10
11      <div>
12          <label>Name:</label>
13          <input type="text" ng-model="YourName" placeholder="Enter your name here">
14
15          <br>
16
17          <h1>Hello {{YourName}}!</h1>
18      </div>
19
20  </body>
21  </html>
```

The above example will give an output having Interpolating a textbox and a ‘Hello !’. Whenever a user types in his/her name in the textbox, name reflects after the word ‘Hello’. If the name is John Doe, then the browsers outputs Hello John Doe! In real time without any need for reloading the page.

### **\*Dev Note:**

Using a CDN AngularJS file or a downloaded AngularJS files is one and the same. But to increase performance and to ensure offline capabilities using a downloaded local AngularJS file is encouraged.

The above application has some keywords which require attention as they are new to any normal JavaScript developer.

The keywords that are introduced in the example are:

1. ng-app
2. ng-model
3. {{YourName}}

### **ng-app**

ng-app is basically an attribute to HTML element. But this attribute has some special purpose which is the key to an angular web application.

The purpose of ng-app is to define and link angular application to HTML. This attribute lets the HTML document and the AngularJS link together. In terms of angular, 'ng-app' is called a **directive**.

### **ng-model**

ng-model is another important directive in AngularJS and plays an important role in our application above.

This directive binds the values of AngularJS application data to HTML input controls. In simple terms, when this attribute is named (\*like "YourName" in our application) and linked to an input field and later linked to our expression (\*{{YourName}}), can reflect the user input on the browser window in real time.

### **{{YourName}}**

Characters between double braces {{ }} are called expressions. Expressions are used to bind application data to HTML. AngularJS application expressions are pure javascript expressions and outputs the data where they are used.

## CHAPTER-4 EXPRESSIONS

Expressions are used to bind application data to html, which are written inside double braces like `{{expression}}`. It behaves in a same way as ng-bind directives. (\*directives are discussed in next chapter in detail).

AngularJS will resolve the expression, and return the result exactly where the expression is written.

AngularJS expressions are much like JavaScript expressions: They can contain literals, operators, and variables.

Let us work with expressions in our second application.

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4
5      <!--scripts-->
6      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.min.js">
7      </script>
8  </head>
9  <body ng-app>
10
11  <div ng-init = "price = 5; cost = 3; employee = {firstname:'John',lastname:'Doe'};">
12      <h3>First Expression: {{2 + 7}}</h3> <!--prints 9-->
13      <br>
14      <h3>Second Expression: {{3 * 7}}</h3> <!--prints 21-->
15      <br>
16      <h3>Third Expression: {{price * cost}}</h3> <!--prints price*cost which is 15-->
17      <br>
18      <h3>Fourth Expression: {{employee.firstname + " " + employee.lastname}}</h3>
19      <!--prints John Doe-->
20  </div>
21
22 </body>
23 </html>
```

(\*we will discuss about ng-init in directives chapter)

This application clearly demonstrates the power of angular expressions. As in the first and second expression, angular simply adds/multiply accordingly and display the result on the output window. In the third expression and fourth, angular takes data from the ng-init directive which is given as an attribute/directive to the div element.

**\*Dev Note:**

Unlike JavaScript expressions, AngularJS expressions can be written inside HTML. AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do. AngularJS expressions support filters, while JavaScript expressions do not.

Our application with ng-init and expressions is an example of **Data Binding**.

Generally, data binding in AngularJS is the synchronization between the model and the view. Our application is an example of one way data binding, here data is directed by the directive ng-init and displayed with the help of expressions.

(\*Data Binding is explained in detail in the further chapters)

## CHAPTER-5 DIRECTIVES

AngularJS directives are used to extend HTML. These are special attributes starting with ng- prefix. AngularJS has a set of built-in directives which offers functionality to your applications. AngularJS also lets you define custom directives.

Below are some important directives we use the most:

ng-app, ng-init, ng-model, ng-repeat

- The **ng-app** directive initializes an AngularJS application.
- The **ng-init** directive initializes application data.
- The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.
- The **ng-repeat** directive repeats html elements for each item in a collection.

### ng-app

ng-app directive starts an AngularJS Application. It defines the root element. It automatically initializes or bootstraps the application when web page containing AngularJS Application is loaded.

Syntax:

```
<div ng-app = “ ”>  
    ---  
</div>
```

### ng-init

ng-init directive initializes an AngularJS Application data. It is used to insert values to the variables to be used in the application. Normally, we will not use ng-init. We will use a controller or module instead. (\*Controllers are explained in later chapters)

### **\*Dev Note:**

Data can be initialized even by using JSON format.

Syntax:

```
<div ng-app = "" ng-init = "countries = [{firstname:'Erik,lastname:Rover'},  
  {firstname:Mike,lastname:Owen}, {firstname:Akhil,lastname:Achha}]">  
  ---  
</div>
```

### **ng-model**

ng-model directive defines the model/variable to be used in AngularJS Application.

Syntax:

```
<h3>Enter your Name: <input type = "text" ng-model = "name"></h3>
```

### **ng-repeat**

ng-repeat directive repeats html elements for each item in a collection.

Syntax:

```
<div ng-app="" ng-init="names=['Hari','Chitra','Keerthana']">  
  <ul>  
    <li ng-repeat="n in names">  
      {{ n }}  
    </li>  
  </ul>  
</div>
```

## CHAPTER-6 DATA BINDING

Data binding in AngularJS is the synchronization between the model and the view.

Data binding in AngularJS is mainly, one-way data binding and two-way data binding. Let us understand these data bindings with examples.

### One-Way Data Binding

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4
5  <!--scripts-->
6  <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.min.js">
    </script>
7
8  </head>
9  <body ng-app>
10
11  <div ng-init="firstName = 'Keerthana'; lastName = 'K';">
12      <h3>First name: {{firstName}}</h3>
13      <h3>Last name: <span ng-bind="lastName"></span></h3>
14  </div>
15
16  </body>
17  </html>
```

This example clearly explains one-way data binding. Here, ng-init acts as a model which consists of data and both expression and also ng-bind which are views. AngularJS binds the model and view and helps printing the name on the output window.

#### **\*Dev Note:**

‘ng-bind’ is a directive that can be replaced in place of expressions. Both expressions and ng-bind do the same work.

## Two-Way Data Binding

Code:

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4
5  <!--scripts-->
6  <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.min.js">
    </script>
7
8  </head>
9  <body ng-app>
10 <!--<div ng-init="firstName = 'Keerthana'; lastName = 'K';">
11     <h3>First name: {{firstName}}</h3>
12     <h3>Last name: <span ng-bind="lastName"></span></h3>
13 </div-->
14 <div ng-app ng-init="firstName = 'Keerthana'; lastName = 'K';">
15     <h3>First name: {{firstName}}</h3>
16     <h3>Last name: <span ng-bind="lastName"></span></h3>
17     <br />
18     <label>Update the First name: <input type="text" ng-model="firstName"/></label>
19     <br />
20     <label>Update the Last name: <input type="text" ng-model="lastName"/></label>
21 </div>
22
23 </body>
24 </html>
```

Output:

Actual Output

**First name: Keerthana**

**Last name: K**

Update the First name:

Update the Last name:

As we update text in text fields, the output changes in real-time.

**First name: Hardhik**

**Last name: K**

Update the First name:

Update the Last name:

When data in the model changes, the view reflects the change, and when data in the view changes, the model is updated as well. This happens immediately and automatically, which makes sure that the model and the view is updated at all times.



## CHAPTER-7 CONTROLLERS & SCOPE

AngularJS application mainly relies on controllers to control the flow of data in the application. Controllers are regular JavaScript Objects.

A controller is defined using **ng-controller** directive.

Each controller accepts \$scope as a parameter which refers to the application/module that controller has to control.

Let us look at an example to understand controllers

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <!--scripts-->
5 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.min.js"></script>
6 </head>
7
8 <body ng-app="MyApp">
9
10 <div ng-controller="MyCtrl">
11
12     <label>Employee Name: <input type="text" ng-model="EMP_Name"></label><br>
13     <label>Employee ID: <input type="num" ng-model="EMP_ID"></label><br>
14
15     <h3>Employee Name: {{EMP_Name}}</h3>
16     <h3>Employee ID: {{EMP_ID}}</h3>
17 </div>
18
19 <!--script-->
20 <script type="text/javascript">
21     var app = angular.module('MyApp', []);
22
23     app.controller('MyCtrl', function($scope) {
24         $scope.EMP_Name = "Keerthana";
25         $scope.EMP_ID = "054236";
26     });
27 </script>
28 </body>
29 </html>
```

Our Application is a perfect example for two-way binding and use of controllers. Here, the above application is named 'MyApp' which is used for defining our application. ng-controller helped us defining controller responsible for action. AngularJS will invoke the controller with a \$scope object. \$scope is the application object (the owner of application variables and functions).

### **\*Dev Note:**

To understand controllers and their working, one must explore higher order functions like call back in JavaScript. We must also understand the working of an anonymous functions. Above application contains an anonymous function. To learn about all function types, check through the following website:

[http://www.w3schools.com/js/js\\_function\\_definition.asp](http://www.w3schools.com/js/js_function_definition.asp)

- ❖ In larger applications, it is common to store controllers in external files, by saving them with '.js' extension.

## **ANGULAR JS SCOPE**

The scope is the binding part between the HTML (view) and the JavaScript (controller). The scope is a JavaScript object with properties and methods, which are available for both the view and the controller.

When adding properties to the **\$scope** object in the controller, the view (HTML) gets access to these properties.

In the view, prefix \$scope is not required to be used, instead we can just refer to a propertyname, like **{{carname}}**.

## **ROOT SCOPE**

All applications have a **\$rootScope** which is the scope created on the HTML element that contains the ng-app directive.

The **rootScope** is available in the entire application.

If a variable has the same name in both the current scope and in the **rootScope**, the application use the one in the current scope.

### **\*Dev Note:**

**\$rootScope** should be accessed and used only once we clearly understand the usage of scope. **\$rootScope** is like a global declaration, it can be used in API calling and instantiation. Usage of **\$rootScope** to instantiate a variable or object help us to use it throughout the controllers.

## CHAPTER-8 FILTERS

Filters are used to change/modify the data and can be clubbed in expressions or directives using pipe character.

S.No	Name of Filter	Description
1	uppercase	converts a text to upper case text.
2	lowercase	converts a text to lower case text.
3	currency	formats text in a currency format.
4	date	format a date to a specified format.
5	number	format a number to a string.
6	limitTo	limits an array/string, into a specified number of elements/characters.
7	filter	filter the array to a subset of it based on provided criteria.
8	orderBy	orders the array based on provided criteria.
9	json	format an object to a JSON string.

Let us make use of filters in our Application.

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4      <!--scripts-->
5      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.min.js">
6      </script>
7  </head>
8  <body ng-app="MyApp">
9
10     <div ng-controller="MyCtrl">
11
12         <p>Employee First Name: {{ firstname | uppercase }}</p>
13         <p>Employee Last Name: {{ lastname | lowercase }}</p>
14         <p>Employee Salary: {{ salary | currency }}</p>
15     </div>
16
17     <!--script-->
18     <script type="text/javascript">
19         var app = angular.module('MyApp', []);
20
21         app.controller('MyCtrl', function($scope){
22             $scope.firstname = "Paul";
23             $scope.lastname = "Walker"
24             $scope.salary = 50000;
25         })
26     </script>
27 </body>
28 </html>
```

Output:

Employee First Name: PAUL

Employee Last Name: walker

Employee Salary: \$50,000.00

### **\*Dev Note:**

The currency filter can be formatted to represent a country before the actual figure of currency.

### **A walkthrough on 'ng-repeat' directive and orderBy filter:**

The ng-repeat directive repeats a set of HTML, a given number of times. The set of HTML will be repeated once per item in a collection. The collection must be an array or an object.

If you have an collection of objects, the ng-repeat directive is perfect for making a HTML table, displaying one table row for each object, and one table data for each object property.

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <!--scripts-->
5   <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.min.js">
6   </script>
7 </head>
8 <body ng-app="MyApp">
9
10  <div ng-controller="MyCtrl">
11    <p ng-repeat="x in countries">{{x}}</p>
12  </div>
13
14  <!--script-->
15  <script type="text/javascript">
16    var app = angular.module('MyApp', []);
17
18    app.controller('MyCtrl', function($scope){
19      $scope.countries = [
20        "India",
21        "New Zealand",
22        "United States of America",
23        "Poland"
24      ];
25    });
26  </script>
27 </body>
28 </html>
```

Output:

India  
New Zealand  
United States of America  
Poland

Here, we can observe ng-repeat has printed all the country names from the array which are defined in the controller.

**\*Dev Note:**

ng-repeat can be used in producing iterative data in many formats such as tables. This directive helps us reduce code on view side, by which loading of a web page will be faster.

Let's use 'orderBy' filter in the above application to print the country names in the alphabetic order.

```
<div ng-controller="MyCtrl">  
  <p ng-repeat="x in countries | orderBy">{{x}}</p>  
</div>
```

Output:

India  
New Zealand  
Poland  
United States of America

Just by adding filter, the names of countries are printed in alphabetical order. Before adding the filter, names were printed as given in the array.

\*position of Poland moved up and USA moved down.

## CHAPTER-9 SERVICES

Services are javascript functions and are responsible to do specific tasks only. This makes them an individual entity which is maintainable and testable. Controllers, filters can call services as on requirement basis. Services are normally injected using dependency injection mechanism of AngularJS.

AngularJS provides many inbuilt services for example, \$https:, \$route, \$window, \$location etc. AngularJS has about 30 built-in services. Each service is responsible for a specific task.

Example, \$https: is used to make ajax call to get the server data. \$route is used to define the routing information and so on. Inbuilt services are always prefixed with \$ symbol.

There are two ways to create a service:

- Factory
- Service

### Factory Method

```
var app = angular.module("myApp", []);

app.factory('MultiplyService', function() {
    var factory = {};

    factory.multiply = function(x, y) {
        return x*y;
    }

    return factory;
});
```

## Service Method

Let us look at examples which help understand the usage of inbuilt services provided by AngularJS and also custom services.

### The \$http Service

The \$http service is one of the most common used services in AngularJS applications. The service makes a request to the server, and lets your application handle the response.

```
var app = angular.module('myApp', []);

app.controller('myCtrl', function ($scope, $http) {
    $http.get("helloWorld.html").then(function (response) {
        $scope.myWelcome = response.data;
    });
});
```

### Custom Service

```
var app = angular.module('myApp', []);

app.service('AddService', function() {
    var result = {};

    result.add = function(a, b) {
        return a + b;
    }

    return result;
});
```

### \*Dev Note:

We can also inject other available services or factory into our custom services.

## CHAPTER-10 ROUTING

If we want to navigate to different pages in your application, but also wants the application to be a SPA (Single Page Application), with no page reloading, we can use the ngRoute module.

SPA (Single Page Application) reduces page loading time. Angular application when served, behaves differently when compared to usual/legacy web applications. Angular SPA loads layout/content based on routing, while keeping the rest of the layout/content universally.

### **Example,**

When we create a website, most of the part (i.e., Navigation header, footer, images, etc) is common for all the pages and only little content is changed from page to page. But the legacy system loads the whole page always, whenever requested thus increasing load time.

Whereas Angular SPA loads the common part first and other content is loaded based upon the request via routing, thus reducing loading time and increasing the smoothness of UI (User-Interface).

The ngRoute module routes the application to different pages without reloading the entire application.

### **\*Dev Note:**

An AngularJS module defines an application. The module is a container for the application controllers. Controllers always belong to a module.

### **\*What do We need for Routing?**

To make our application ready for routing, we must include the AngularJS route module:

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



Then we must add the `ngRoute` as a dependency in the application module:

```
var app = angular.module("myApp", ["ngRoute"]);
```

Example App:

```
<body ng-app="myApp">

  <nav>
    <a href="#/page1">About Us</a>
    <a href="#/page2">Our Services</a>
    <a href="#/page3">Contact</a>

    <ng-view></ng-view>
  </nav>

  <script>
    var app = angular.module('myApp', ["ngRoute"]);

    app.config(function($routeProvider) {
      $routeProvider
        .when("/", {
          templateUrl : "main.html"
        })
        .when("/page1", {
          templateUrl : "page1.html"
        })
        .when("/page2", {
          templateUrl : "page2.html"
        })
        .when("/page3", {
          templateUrl : "page3.html"
        })
    })
  </script>
</body>
```

\*`ngView` is a directive that complements the `$route` service by including the rendered template of the current route into the main layout ( `index.html` ) file.

## CHAPTER-11 COMPLETE APPLICATION

Before we start building a complete application there are certain issues we must first handle. An application is complete only once all the connections are knotted correctly. We have learned Angularjs so far and now it is time we use a backend to make our application complete.

There are two major and most used ways or options for us to continue with:

1. MEAN Stack (MongoDB, ExpressJS, AngularJS, NodeJS)
2. MBAAS / BAAS (Mobile Backend as a service/ Backend as a service)

In this documentation, we are going to develop app using a BAAS way.

### FIREBASE

[Firebase](#) is a mobile and web application platform with tools and infrastructure designed to help developers build high-quality apps. Firebase is made up of complementary features that developers can mix-and-match to fit their needs.

For our application, we are using firebase to authenticate users.

Firebase Auth is a service that can authenticate users using only client-side code. It supports social login providers like Facebook, GitHub, Twitter and Google. Additionally, it includes a user management system whereby developers can enable user authentication with email and password login stored with Firebase.

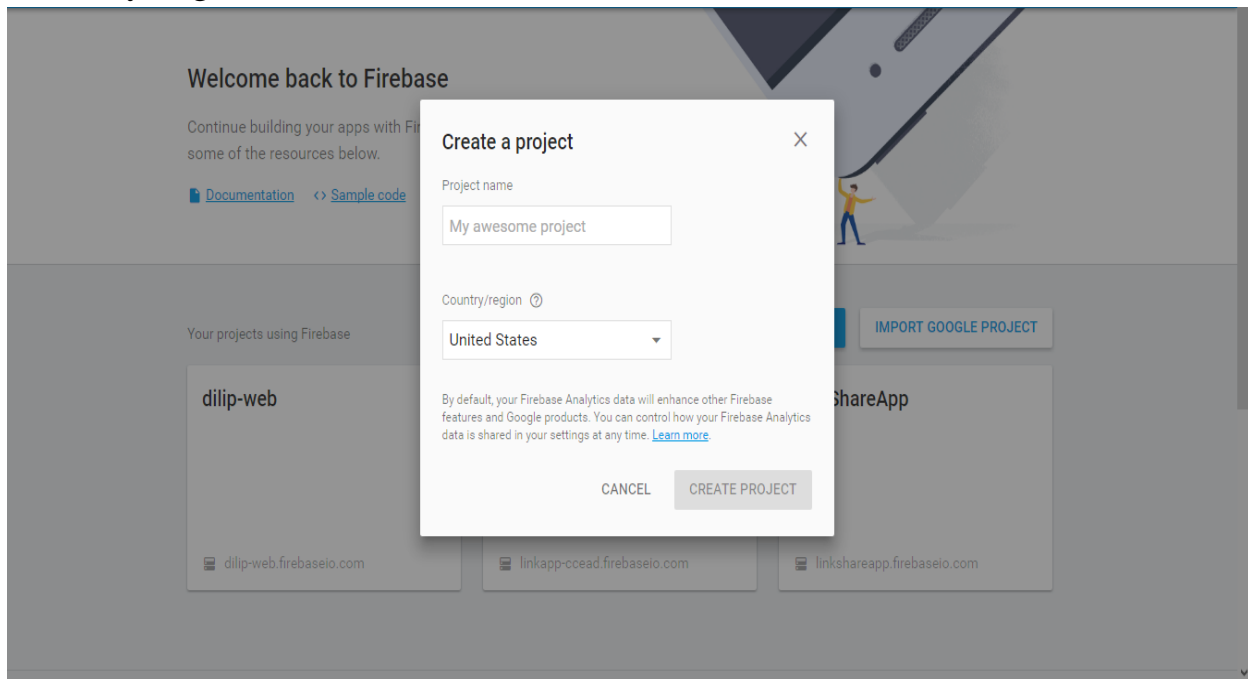
Firebase also provides several other services like:

- Real time Database
- Storage & Hosting
- Test Lab & Crash Reporting
- Notifications, Adwords & Analytics

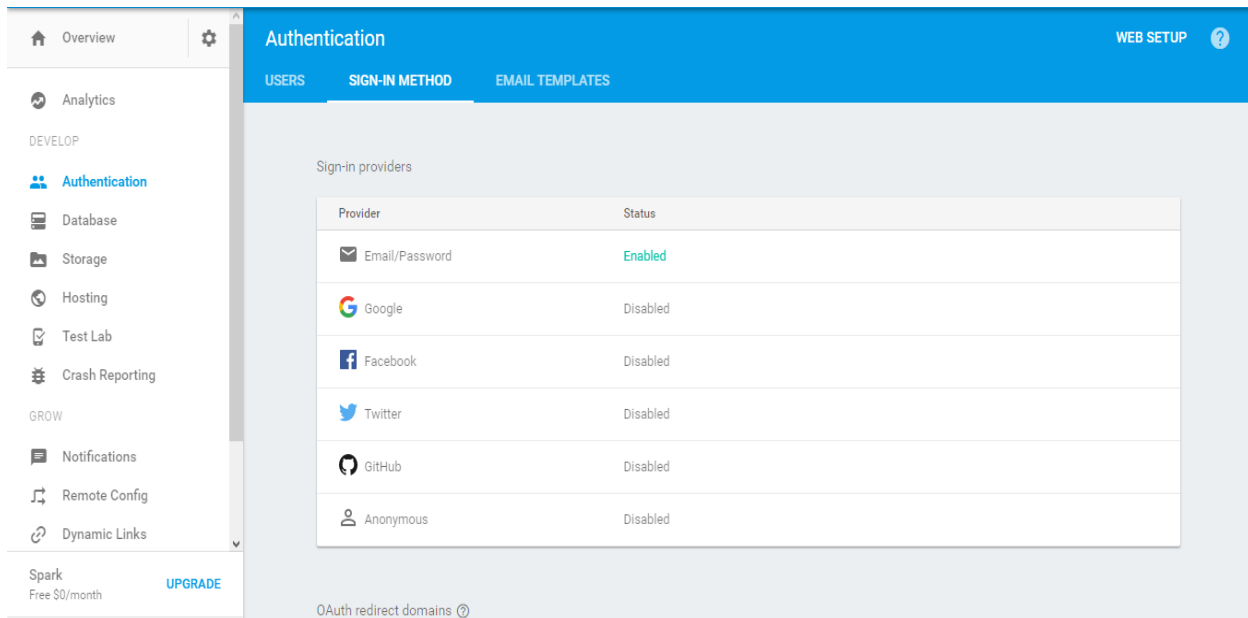
#### **\*Dev Note:**

Firebase is now a part of Google, and has a large developer community. It's Free!

Before we start coding our application, we must first visit [Firebase](#) website and create our application. Firebase website is very intuitive and just with few steps we are ready to go.



After creating our application, we can enable Authentication services with just a tap.



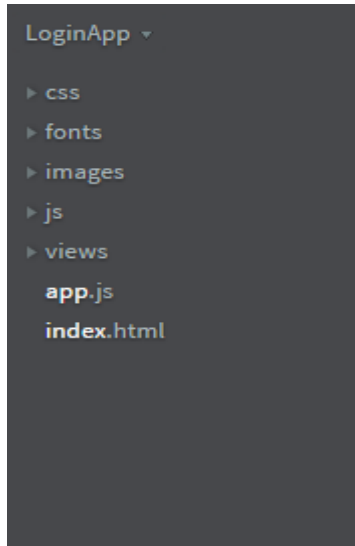
For our application, we use simple Email-Password auth type.

## Next Steps:

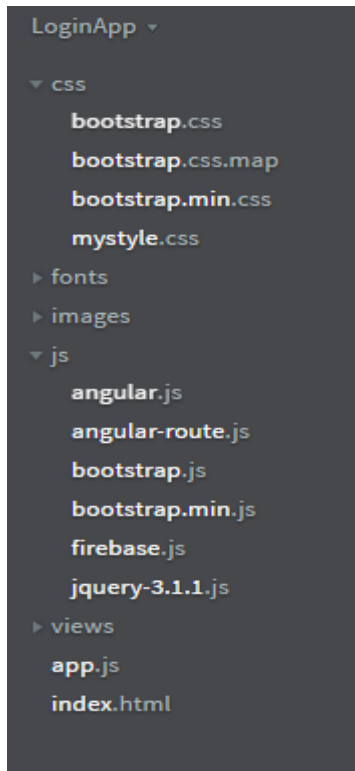
### Creating a Local Directory

Using Brackets or any other Developer Editor we must create our local app directory and include Bootstrap, angular and firebase files.

Our app file structure looks like this,



### Including Bootstrap and other CSS & JS files



## Basic HTML Structure

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <!--css-->
6      <link type="text/css" rel="stylesheet" href="css/mystyle.css" />
7      <link type="text/css" rel="stylesheet" href="css/bootstrap.min.css" media="screen,projection" />
8
9      <!--meta-->
10     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
11 </head>
12
13 <body ng-app="LoginApp">
14     <!--<nav>|
15         </nav>-->
16
17     <ng-view></ng-view>
18
19     <!--<footer>
20         </footer>-->
21
22     <!--Import jQuery before materialize.js-->
23     <script type="text/javascript" src="js/jquery-3.1.1.js"></script>
24     <script type="text/javascript" src="js/bootstrap.min.js"></script>
25     <script type="text/javascript" src="js/angular.js"></script>
26     <script type="text/javascript" src="js/angular-route.js"></script>
27     <script type="text/javascript" src="app.js"></script>
28 </body>
29
30 </html>
```

### Final Step:

After the application is connected to firebase and tested thoroughly, we can either host our application via Firebase hosting (or) Amazon Web Services (or) Azure (or) Other custom domain hosting services.

### \*Dev Note:

This application is a perfect example for full stack JavaScript app. Our app also depicts MVC architecture and introduces Baas (Backend as a Service). The entire code of the application is available at [github](#).

## REFERENCES

This documentation is an adaptation and inspiration from following books and websites or web journals:

1. Pro AngularJS by Adam Freeman
2. W3Schools - <http://www.w3schools.com/angular/>
3. Tutorials Point - <https://www.tutorialspoint.com/angularjs/>
4. AngularJS official - <https://docs.angularjs.org/guide/concepts>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

