Getting “MEAN”

- A Practical Workshop

Angular JS

Angular JS :

- AngularJS is an open source, JavaScript based web application development framework.

- It is used in **Single Page Application** (SPA) projects. It extends HTML DOM with additional attributes and makes it more responsive to user actions.

- AngularJS is a structural framework for dynamic web applications.

-It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google.

General Features :

- AngularJS provides developers an options to write client side applications using JavaScript in a clean **Model View Controller** (MVC) way.

- AngularJS is a efficient framework that can create **Rich Internet Applications** (RIA).

- Applications written in AngularJS are **cross-browser compliant**. AngularJS automatically handles JavaScript code suitable for each browser.

- 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.

- Its **two way data binding** and **dependency injection** eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner **with any server technology**.

Overall, AngularJS is a framework to build large scale, high performance, and easy-to-maintain web applications.

Core Features :

- **Data-binding**: It is the automatic synchronization of data between model and view components.

- **Dependency Injection:** AngularJS has a built-in dependency injection subsystem that helps the developer to create,understand, and test the applications easily.

- **Scope:** These are objects that refer to the model. They act as a glue between controller and view.

- **Controller:** These are JavaScript functions bound to a particular scope.

- **Directives:** Directives are markers on DOM elements such as elements, attributes, css, and more. These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives such as ngBind, ngModel etc.

- **Services:** AngularJS comes with several built-in services such as $http to make a XMLHttpRequests. These are singleton objects which are instantiated only once in app.

- **Filters:** These select a subset of items from an array and returns a new array.

- **Templates:** These are the rendered view with information from the controller and model. These can be a single file (such as index.html) or multiple views in one page using partials.

- **Routing:** Routing concept of angular js is used for switching views.

- **Model View Whatever**: MVW is a design pattern for dividing an application into different parts called Model, View, and Controller, each with distinct responsibilities. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel). The Angular JS team refers it humorously as Model View Whatever.

- **Deep Linking:** Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.

Advantages :

- AngularJS provides capability to create **Single Page Application** in a very clean and maintainable way.

- AngularJS provides data binding capability to HTML. Thus, it gives user a **rich and responsive experience**.

- AngularJS code is **unit testable**.

- AngularJS uses **dependency injection** and make use of separation of concerns.

- AngularJS provides **reusable components**.

- In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.

Note : On the top of everything, AngularJS applications can run on all major browsers and smart phones, including Android and iOS based phones/tablets.

Disadvantages :

Though AngularJS comes with a lot of merits, here are some points of concern:

- Not Secure: Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.

- Not degradable:(Disadvantage of being a JS framework**)** If the user of your application disables JavaScript, then nothing would be visible, except the basic page.

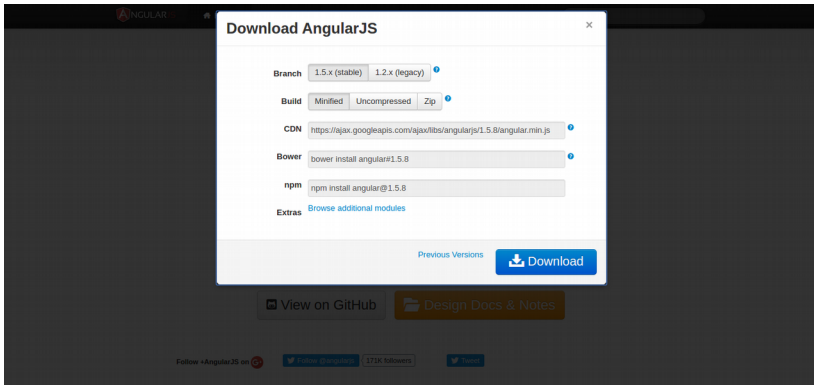
Installation

There is not much to do with the installation for Angular JS. As we use in frontend and runs on Browser we just need a library JS file of Angular JS.

There are two options to get the lib file of angular js.

i. Download the Angular JS file from the website(https://angularjs.org/)

Select the download Angular JS 1



ii. In HTML file, refer to CDN path of Angular JS (https://cdnjs.com/libraries/angular.js/)

Example :

<script scr="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.5.8/angular.min.js"> </script>

First Application

I will not waste time, lets directly dig into the AngularJS application which consists of following three important parts −

- **ng-app** − This directive defines and links an AngularJS application to HTML.
- **ng-model** − This directive binds the values of AngularJS application data to HTML input controls.
- **ng-bind** − This directive binds the AngularJS Application data to HTML tags.

Steps to create AngularJS Application

Step 1 − Load framework :

Being a pure JavaScript framework, It can be added using <Script> tag.

```
<script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.j
s"></script>
```

Step 2 − Define AngularJS Application using ng-app directive

```
<div ng-app = "">
      ...
      ...
</div>
```

Step 3 − Define a model name using ng-model directive

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

Step 4 − Bind the value of above model defined using ng-bind directive.

```
<p>Hello <span ng-bind = "name"></span>!</p>
```

or you can use curly braces.

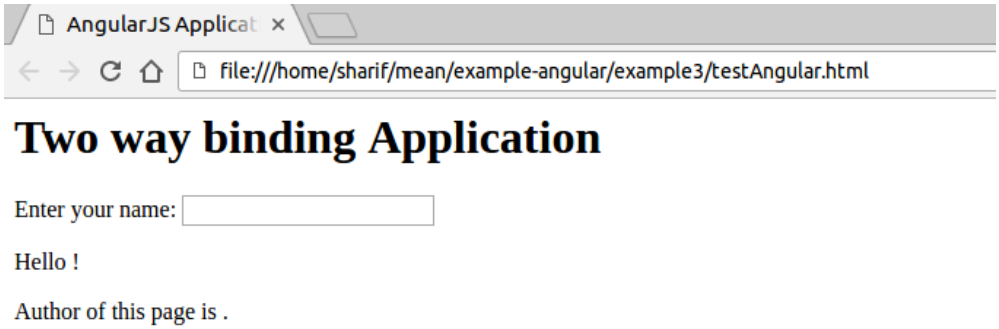
```
{{name}}
```

Conclusion of the above steps is Example 1 :

You can download the source code from github (https://github.com/virtualSharif/gettingMEAN/tree/master/angular/examples/example1)

To run AngularJS Application

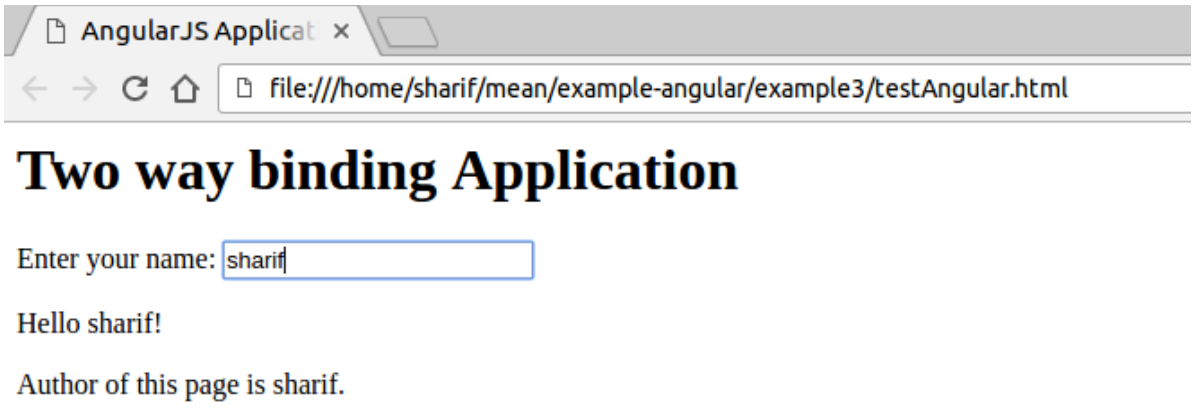
Open the file testAngular.html in browser.



Output :

Open textAngular.html in a web browser.

Enter your name and see the result.



How AngularJS integrates with HTML

- **ng-app** directive indicates the start of AngularJS application.
- **ng-model** directive then creates a model variable named "name" which can be used with the html page and within the div having ng-app directive.
- **ng-bind** then uses the name model to be displayed in the html span tag whenever user input something in the text box.
- **{{name}}** used as an alternative for ng-bind.
- Closing tag indicates the end of AngularJS application.

Data Binding

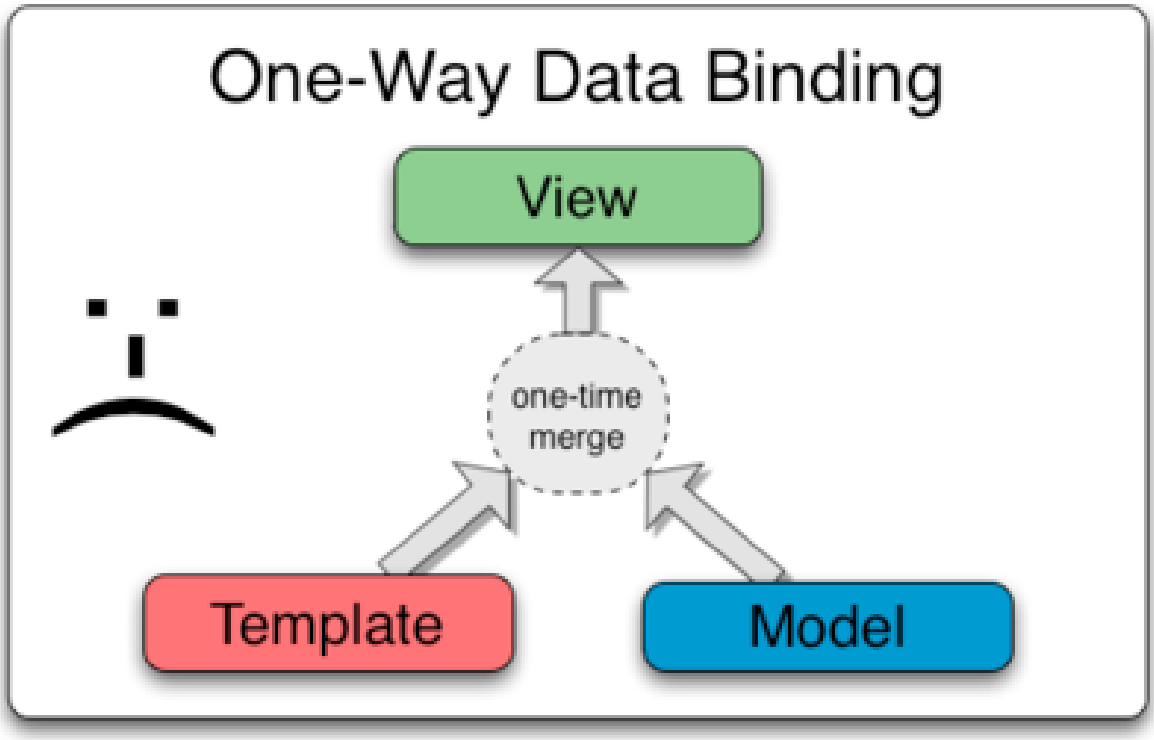
Data Binding :

Data-binding in Angular apps is the automatic synchronization of data between the model and view components.

The way that Angular implements data-binding lets you treat the model as the **single-source-of-truth** in your application.

The view is a projection of the model at all times. When the model changes, the view reflects the change, and vice versa.

Data Binding in Classical Template Systems



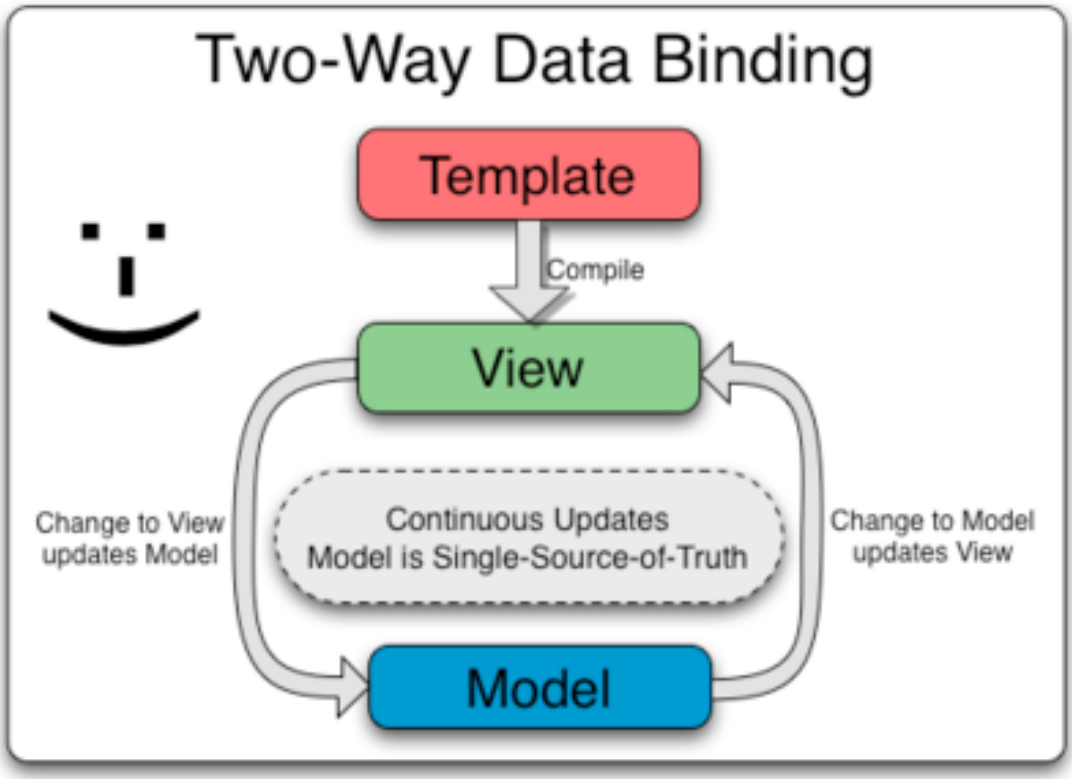
Most templating systems bind data in only one direction: they merge template and model components together into a view.

After the merge occurs, changes to the model or related sections of the view are **NOT** automatically reflected in the view.

Worse, any changes that the user makes to the view are not reflected in the model.

This means that the developer has to write code that constantly syncs the view with the model and the model with the view.

Data Binding in Angular Templates



Angular templates work differently.
First the template (which is the uncompiled HTML along with any additional markup or directives) is compiled on the browser.

The compilation step produces a live view. Any changes to the view are immediately reflected in the model, and any changes in the model are propagated to the view.

The model is the **single-source-of-truth** for the application state, greatly simplifying the programming model for the developer.

You can think of the view as simply an instant projection of your model.

Because the view is just a projection of the model, the controller is completely separated from the view and unaware of it.

This makes testing a snap because it is easy to test your controller in isolation without the view and the related DOM/browser dependency.

MVC

MVC Architecture

MVC stands for Model View Controller which is very popular design pattern in web world. It helps you to organise your application in three different layers and isolate the business logic from its presentation.

- **Model** - represent current state and data of your application
- **View** - responsible to display/show the data
- **Controller** - responsible to controls the relation between Models and Views.

Now let's understand how these three component works in AngularJS MVC Architecture step by step.

Model -

In AngularJS, model consists of all premitive data type such as integer, string and boolean and complex type in form of object.

In simple word **model is just a plain javascript object**.

But you can build your model from any database like Sql Server or Mysql or from JSON file.

In below given syntax **$scope** is an object, user is variable which is added to scope object.

Syntax :

```
<script>
	$scope.user =  {
		'name'    :   'Sharif Malik',
		'address' :   '12-13-283/A1, Chinchwad, Pune ',
		'email'   :   'malik_sharif@yahoo.com'
	}
</script>
```

View :

In AngularJS, view is the DOM elements which is used to display data.

To display data from controller, use **expression** in view. Since AngularJS supports two-way data binding any changes in model data, view will update automatically.

Example :

```
<html>
      <p> {{user.name}} </p>
      <p> {{user.address}} </p>
      <p> {{user.email}} </p>
</html>
```

Controller-

In AngularJS, controller provide great **control over view and model** in order to fetch the data as per request and display in the view.

Most important thing here is your model is lives inside the controller.

Syntax :

```
<script>
var app = angular.module('myApp', []);
app.controller('UserController', function($scope) {
   //Creating model here
      $scope.user =  {
       'name'    :   'Sharif Malik',
       'address' :   '12-13-283/A1, Chinchwad, Pune',
       'email'   :   'malik_sharif@yahoo.com'
      }
</script>
```

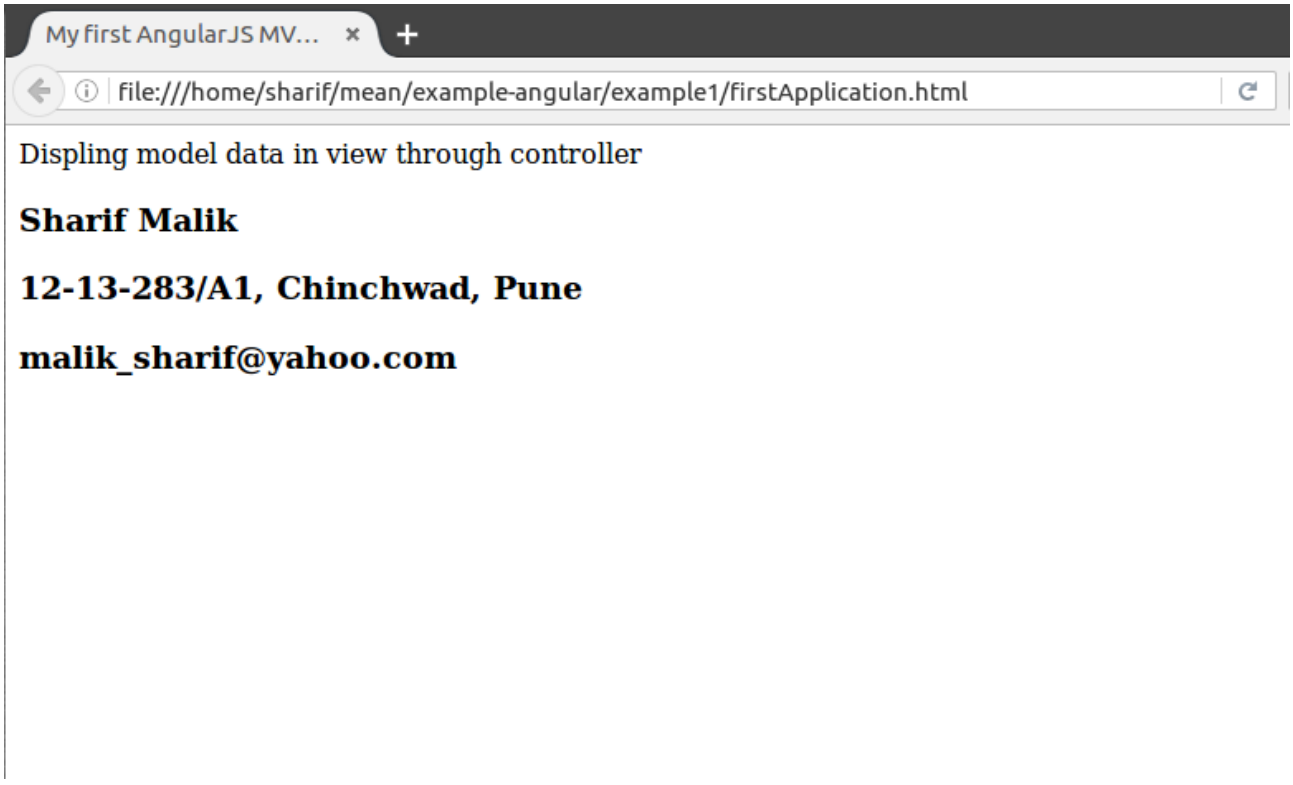
Second Application : (Fetching data from controller : MVC way**)**

Example 2 :
You can download the source code from the github link (https://github.com/virtualSharif/gettingMEAN/tree/master/angular/examples/example2)

To run the application :

Open the firstApplication.html with mozilla Firefox Web Browser

Output:



Example 3 :

Lets modularize the example2 in files :

We can move the controller in controller.js file and import in index.js

Example 3 :

You can download the source code from github link (https://github.com/virtualSharif/gettingMEAN/tree/master/angular/examples/example3)

Here I have seperated the script into seperate file and html into seperate file.

1. index.html : this is main page which will be called from browser.

2. app.js : this javascript file which contains all the angular module creation code and controller code.

Hence , you can try to run the code , output will be the same as above second application.

Expression

Angular JS Expression :

AngularJS binds data to HTML using **Expressions**.

AngularJS expressions can be written inside double braces **{{ expression }}**

AngularJS expressions can also be written inside a directive **ng-bind="expression"**

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.

Syntax Example :

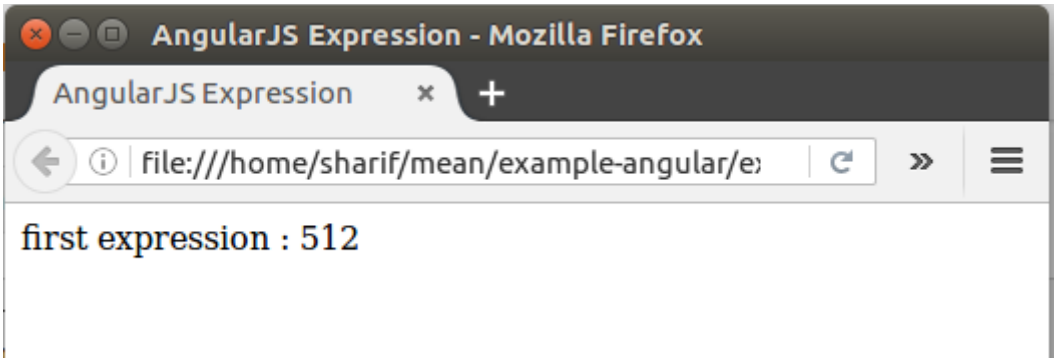
i. {{ 15 + 15 }} ii.{{ firstName + " " + lastName }}

iii. {{ quantity * cost }}

Example :
You can download the source code from the github : (https://github.com/virtualSharif/gettingMEAN/tree/master/angular/examples/example4-expression)

```
<html ng-app="">
<head>
<title>AngularJS Expression</title>
<script src="./lib/angular.min.js"></script>
</head> <body>
      <p>first expression : {{ 256 + 256 }}</p>
</body> </html>
```

Note : You can write expressions wherever you like, AngularJS will simply resolve the expression and return the result.



Modules

Angular JS Modules :

An AngularJS module defines an application.

The module is a container for the different parts of an application.

The module is a container for the application controllers.

Why modules ?

AngularJS supports modular approach.

Modules are used to separate logics say services, controllers, application etc. and keep the code clean.

We define modules in separate .js files and name them as per the module.js file.

Creating a Module :

A module is created by using the AngularJS function **angular.module**

Example :

You can download the source code from github (https://github.com/virtualSharif/gettingMEAN/tree/master/angular/examples/example5-modules)

Files :

In our example, there are two js files and one html file:

i. app.js

Content :

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

Note : The [] parameter in the module definition can be used to define dependent modules.

Without the [] parameter, you are not creating a new module, but retrieving an existing one.

ii. userController.js :

Content :

```
//Creating controller here
myApp.controller('UserController', function($scope) {
	//Creating model here
	$scope.user =  {
		'name'    :   'Sharif Malik',
		'address' :   '12-13-283/A1, Chinchwad, Pune',
		'email'   :   'malik_sharif@ymail.com'
	};
});
```

myApp.controller registers controller named as UserController which can used at runtime to bind with any html files.

iii. index.html :

Content :

```
<html ng-app="myApp">
<head>
<title>AngularJS MVC Architecture code</title>
<script src="./lib/angular.min.js"></script>
<!-- main app -->
<script src= "./app.js"></script>
<!-- controllers-->
<script src= "./userController.js"></script>
</head>
<body>
```

```
    <p>Displing model data in view through controller</p>
    <div ng-controller="UserController">
        <h3>{{ user.name }} </h3>
        <h3>{{ user.address }} </h3>
        <h3>{{ user.email }} </h3>
    </div>
</body>
</html>
```

When to Load the Library

While it is common in HTML applications to place scripts at the end of the **<body>** element, it is recommended that you load the AngularJS library either in the**<head>** or at the start of the**<body>**.

This is because calls to **angular.module** can only be compiled after the library has been loaded.

Explanation :

Now, when **ng-app="myApp"** says DOM that it needs to create angular context and look for myApp angular module.

Then it loads all the dependency file declared in head section i.e **app.js** and **userController.js**

It finds **myApp** module in app.js file and it parses index.html ahead.

When index.html finds the **ng-controller** and it looks for the controller, it searches in current file and in all the dependency files as well. It finds UserController in userController.js file as angular.module's controller.

{{firstname}} - these are the expressions and angular solve this expression using the scope objects from controllers. This is how it works.

Enjoy angular.js in modules **;)**

Directives

Directives :
AngularJS lets you **extend HTML with new attributes** called Directives.

AngularJS has a set of built-in directives which offers functionality to your applications.

AngularJS also lets you define your own directives.

AngularJS directives are extended HTML attributes with the prefix **ng-**.

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.

Remember : The **ng-app** directive also tells AngularJS that the element is the "**owner**" of the AngularJS application.

 # Repeating HTML Elements :

The **ng-repeat** directive repeats an HTML element:

Example :

```
<div ng-app="" ng-init="names=['Sharif','Atul','Ravi','Praphulla']">
	<ul>
		<li ng-repeat="name in names">
			{{ name }}
		</li>
	</ul>
</div>
```

Few are the list of angular built-in directives :

https://docs.angularjs.org/api/ng/directive

Controllers

Angular Controllers :

AngularJS controllers **control the data** of AngularJS applications.

AngularJS controllers are regular **JavaScript Objects**.

The **ng-controller** directive defines the application controller.

A controller is a **JavaScript Object**, created by a standard JavaScript **object constructor**.

Example :

You can refer the above Example 2 or Example 3 in this docs.

Explanation:

The AngularJS application is defined by **ng-app="myApp"**. The application runs inside the .

The **ng-controller="userContoller"** attribute is an AngularJS directive. It defines a controller.

The user**Controller** function is a JavaScript function.

AngularJS will invoke the controller with a **$scope** object.

In AngularJS, **$scope** is the application object (the owner of application variables and functions).

The controller creates two properties (variables) in the scope (**firstName** and **lastName)** and one method in the scope (getFullName) which return some value.

The **ng-model** directives bind the input fields to the controller properties (firstName and lastName).

Services

Angular Js Services :

In AngularJS, **a service is a function, or object**, that is available for, and limited to, your AngularJS application.

AngularJS has about 30 built-in services.

https://docs.angularjs.org/api/ng/service

Why to use services ?

For many services, like the **$location** service, it seems like you could use objects that are already in the DOM, like the **window.location** object, and you could, but it would have some limitations, at least for your AngularJS application.

AngularJS constantly supervises your application, and for it to handle changes and events properly, AngularJS prefers that you use the **$location service instead of the window.location object**.

There are many service as mentioned above, one of them is the $location service.

The $location service has methods which return information about the location of the current web page.

Example :

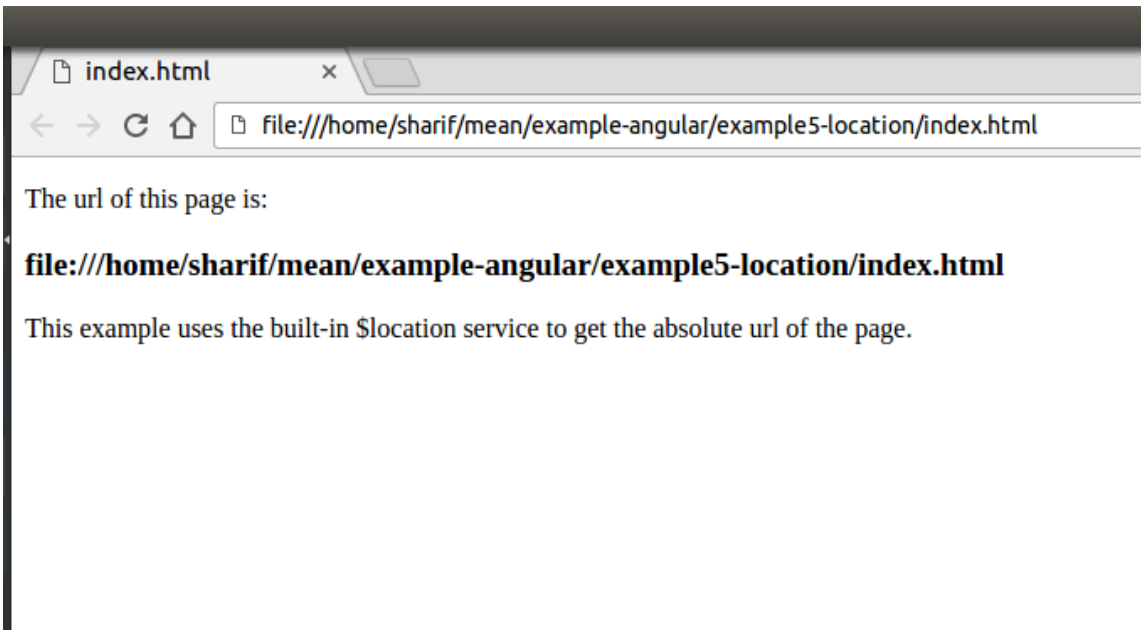
You can download the source code from the github link (https://github.com/virtualSharif/gettingMEAN/tree/master/angular/examples/example6-location)

Content :

```
<!DOCTYPE html>

<html>
<script src="./lib/angular.min.js"></script>

<body>

    <div ng-app="myApp" ng-controller="myCtrl">
```

```
        <p>The url of this page is:</p>
        <h3>{{myUrl}}</h3>
</div>
        <p>This example uses the built-in $location service to get the
        absolute url of the page.</p>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $location) {
   $scope.myUrl = $location.absUrl();
});
</script></body></html>
```

Output :



Note : the **$location service** is passed in to the controller as an argument. In order to use the service in the controller, it must be defined as a dependency.

To learn more about $location service :
https://docs.angularjs.org/api/ng/service/$location

AJAX in Angular JS

AJAX in Angular JS :

- AngularJS provides **$http** control which works as a **service** to read data from the server.

- The AngularJS **$http service makes a request to the server, and returns a response.**

- The server (written in any languages which is nothing but web serivces) internally processes the request (business logic or db call , etc) to get the desired response and generate the response(in this example we are expecting JSON format).

- Once the data is ready, $http can be used to get the data from server in the following manner −

$http Service :

The $http service is a core Angular service that facilitates communication with the remote HTTP servers via the browser's XMLHttpRequest object or via JSONP.

Syntax:

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

app.controller('userController', function($scope, $http) {
   $http({
      method : "GET",
      url : "http://localhost:1991/api/users”
      }).then(function mySucces(response) {
         $scope.users = response.data;
      }, function myError(response) {
       $scope.users= response.statusText;
   });
});
```

Methods:

.delete() : use to send the DELETE HTTP request

.get() : use to send the GET HTTP request

.patch() : use to send the PATCH HTTP request

.post() : use to send the POST HTTP request

.put() : use to send the PUT HTTP request

Note : These methods are shortcuts for the calling $http specific method.

Properties :

The **response from the server is an object** with these properties:

.config : the object used to generate the request.

.data : a string, or an object, carrying the response from the server.

.headers : a function to use to get header information.

.status : a number defining the HTTP status.

.statusText : a string defining the HTTP status.

Example :

For testing the **$http service**, we need the some server to response with expected JSON.

So lets use our server code for the web service and merge the angular JS code.

Please Note : Don't panic , Everything will be clear in the future.
Like : How I am adding angular.js code in the node.js application.

Example7–http :

You can download the source code from the github link (https://github.com/virtualSharif/gettingMEAN/tree/master/angular/examples/example7-http)

To run : Follow the mongodb docs last example, where you need to run the server of mongo i.e.

> **mongod**

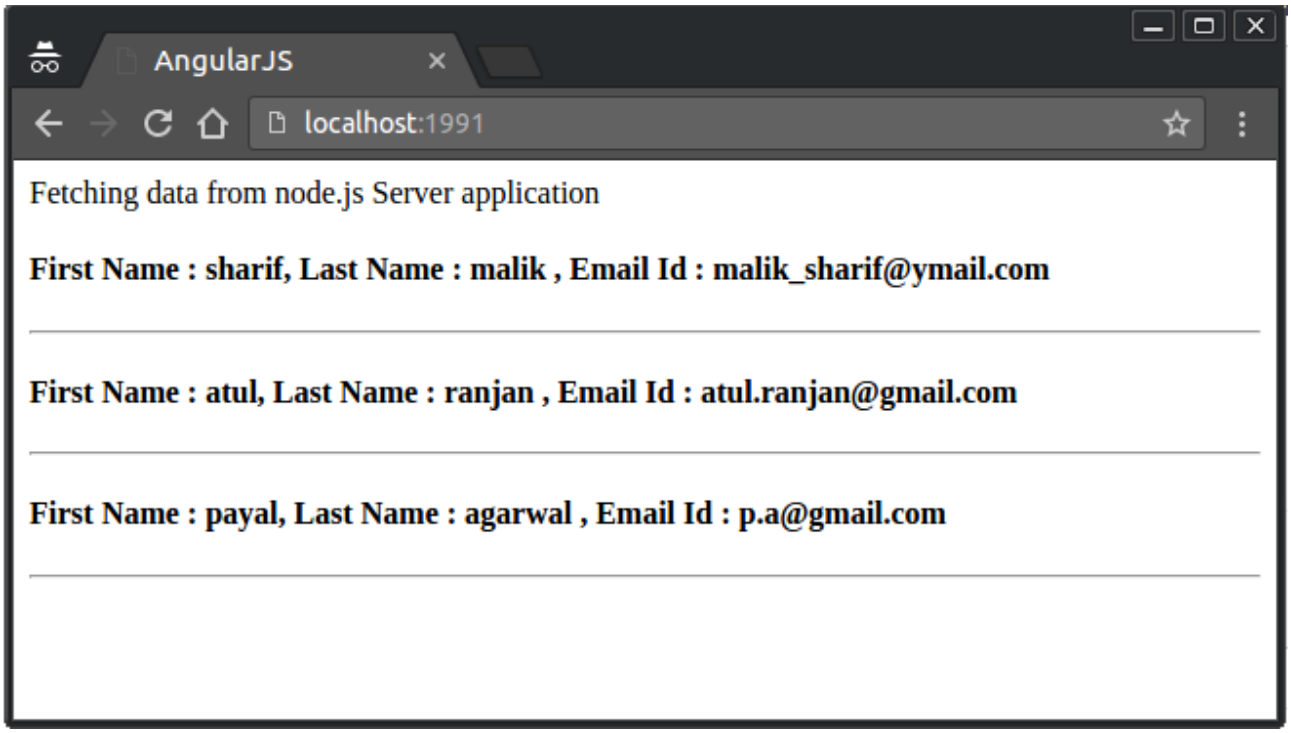
On new command prompt :

> **node app.js**

Output :

Application is running at http://localhost:1991

Now you can open the browser to see the angular js output:



Note : Output depends upon the values in your mongodb.

Explanation :

- The application backend is node, express and mongodb as we completed this in 3rd step of this workshop.

- So this example, uses that backend application to fetch the data using **$http service** from angular js.

- The application defines the **userController** controller, with a **$scope** and **$http** object.

- $http is an **XMLHttpRequest object** for requesting external data.

- $http.get() reads **JSON data** from http://localhost:1991/api/users – this api we have already designed in 3rd Step of the workshop.

- On success, the controller creates a property, $scope.users, in the scope, with JSON data from the server.

Routing / Views

Angular JS Routing :

AngularJS supports **Single Page Application** via **multiple views on a single page**. To do this AngularJS has provided **ng-view** and **ng-template** directives and **$routeProvider** services.

The **ngRoute** module helps your application to become a **Single Page Application**.

What is Routing in AngularJS?

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

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

Steps require to add routing in application ?

Step 1: include

To make your applications ready for routing, you must include the AngularJS Route module:

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

Step 2 : Dependency Injection

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

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

Step 3 :

Now your application has access to the route module, which provides the **$routeProvider**.

Use the **$routeProvider** to configure different routes in your application:

Example :

```
mainApp.config(function($routeProvider) {
        $routeProvider.
        when('/add', {
          templateUrl: 'addUser.html',
          controller: 'AddUserController'
        }).
        when('/view', {
          templateUrl: 'viewUser.html',
          controller: 'ViewUserController'
        }).
        otherwise({
          redirectTo: '/add'
        });
      });
```

Explanation :

$routeProvider : With the $routeProvider you can define what page to display when a user clicks a link.

Define the **$routeProvider** using the **config** method of your application. Work registered in the **config** method will be performed when the application is loading.

Additionally information :

Controllers :

With the $routeProvider you can also define a controller for each "view".

Example : Look into the above example of **config method** where we have assigned the controllers to specific view.

Template :

In the previous examples we have used the templateUrl property in the **$routeProvider.when** method.

You can also use the template property, which allows you to write HTML directly in the property value, and not refer to a page.

The otherwise method :

In the previous examples we have used the when method of the $routeProvider.

You can also use the **otherwise method**, which is the **default route** when none of the others get a match.

Example : In above example if route doesn't matches to any route then it by defaults route will take the application to **/add**.

Step 4 : ng-view container

Where does it template loads ?

Your application needs a container to put the content provided by the routing.

This container is the **ng-view** directive.

There are three different ways to include the **ng-view directive** in your application:

i. <div ng-view>

ii. <ng-view></ng-view>

iii.

Note : Applications can only have one ng-view directive, and this will be the placeholder for all views provided by the route.

Example8-route :

You can download the source code from the github link (https://github.com/virtualSharif/gettingMEAN/tree/master/angular/examples/example9-routing)

Files :

There are 3 files :

i. index.html : the main file to run the application which also contains the js script.

ii. addUser.html : it is partial which is loaded at runtime when users click on Add User link.

iii. viewUser.html : it is partial which is loaded at runtime when users click on View User link.

To run the application :

Open the index.html file with mozilla firefox

Output :

When we click on Add User link :

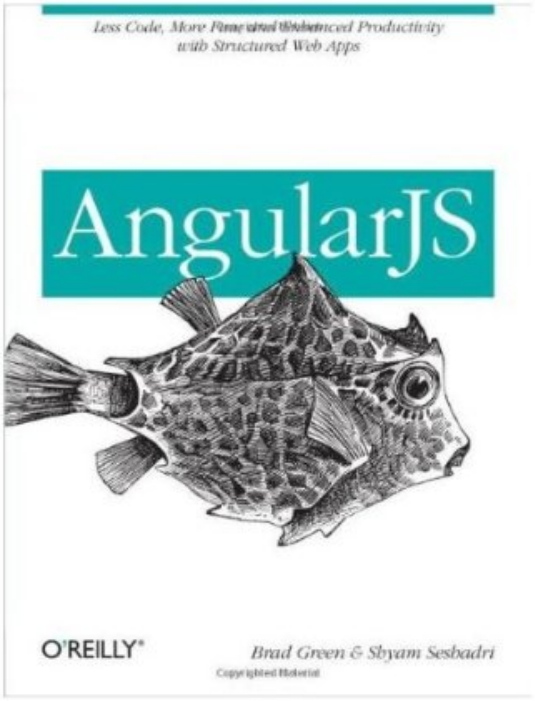
When we click on View users link:

Example9-routing-modularize :

Lets modularize the code of example8-routing. It means js script in js files. Controllers in controller files. Partials html in partials directory and so on.

Angular JS Books & Docs

Book : Angular JS, Brad Green and Sbyam Sesbadri, Oreilly



Online Links for documentation :

https://docs.angularjs.org/guide

Video :

https://www.youtube.com/user/angularjs

Online Free course :

https://www.codeschool.com/pages/angular-1-vs-2