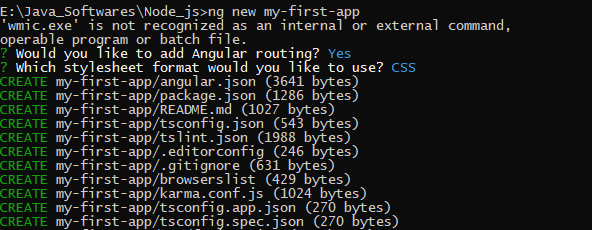
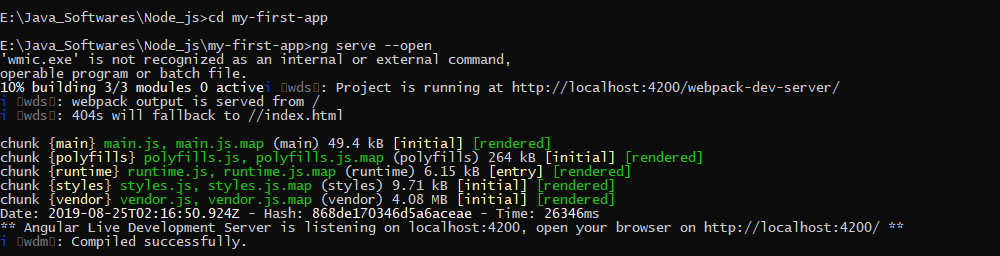
Angular\_js Doc





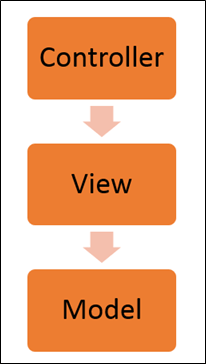
AngularJS is a Javascript framework used for building MVC based applications.

**MVC** – The framework is built on the famous concept of MVC (Model-View-Controller). This is a design pattern used in all modern day web applications. This pattern is based on splitting the business logic layer, the data layer, and presentation layer into separate sections. The division into different sections is done so that each one could be managed more easily.

**Data Model Binding** – You don't need to write special code to bind data to the HTML controls. This can be done by Angular by just adding a few snippets of code

**AngularJS Architecture**

Angular.js follows the MVC architecture, the diagram of the MVC framework as shown below.

[](https://www.guru99.com/images/AngularJS/010416_0549_AngularJSIn1.png)Angularjs Architecture Diagram

* The Controller represents the layer that has the business logic. User events trigger the functions which are stored inside your controller. The user events are part of the controller.
* Views are used to represent the presentation layer which is provided to the end users
* Models are used to represent your data. The data in your model can be as simple as just having primitive declarations. For example, if you are maintaining a student application, your data model could just have a student id and a name. Or it can also be complex by having a structured data model. If you are maintaining a car ownership application, you can have structures to define the vehicle itself in terms of its engine capacity, seating capacity, etc.

**AngularJS Advantages**

* Since it's an open source framework, you can expect the number of errors or issues to be minimal.

Two-way binding

Routing: Angular can take care of routing which means moving from one view to another. This is the key fundamental of single page applications; wherein you can move to different functionalities in your web application based on user interaction but still stay on the same page.

Advantage of AngularJS

There are a lot of JavaScript frameworks for building web applications. So, it is a genuine question, why to use Angular JS.

**Following are the advantages of AngularJS over other JavaScript frameworks:**

**Two way data binding:** Data-binding in AngularJS apps is the automatic synchronization of data between the model and view components. The way that AngularJS implements data-binding

**Testing:** Angular JS is designed in a way that we can test right from the start. So, it is very easy to test any of its components through unit testing and end-to-end testing.

* **Model View Controller:** In Angular JS, it is very easy to develop application in a clean MVC way. You just have to split your application code into MVC components i.e. Model, View and the Controller.
* Directives, filters, modules, routes etc.



The MVC pattern is made up of the following three parts:

1. **Model:** It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.
2. **View:** It is responsible for displaying all data or only a portion of data to the users. It also specifies the 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.
3. **Controller:** It is responsible to control the relation between models and views. It 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.
4. Directives
5. Expressions
6. $scope object
7. Controllers
8. Model
9. Module
10. Filters
11. Data binding



**Features**

The following are the features of AngularJs:

1. **Modules.**
2. **Directives.**
3. **Expressions.**
4. **Controller.**
5. **Scope.**
6. **Data Binding.**
7. **Validations.**
8. **Filters.**
9. **Services.**
10. **Routing.**
11. **Dependency Injection.**
12. **Testing.**

**1. Modules**

* An Angular module defines an application. A module is a container for the various parts of an application. All application controllers should belong to a module.
* An Angular module is simply a collection of functions defined in a JavaScript file.
* A module divides an Angular app into small, reusable and functional components that can be integrated with other Angular apps.
* Each module is identified by a unique name and can be dependent on other modules. In AngularJS, every web page (view) can have a single module assigned to it via ng-app directive.

**Example-2 Showing a module**

1. <! DOCTYPE html>
2. <html>
3. <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
4. <body>
5. <div ng-app="myApp" ng-controller="myCtrl"> //Referring module name myApp
6. {{ firstName + " " + lastName }}
7. </div>
8. <script src="myApp.js"></script>
9. <script src="myCtrl.js"></script>
10. </body>
11. </html>

**MyApp.js**

1. var app = angular.module("myApp", []); //defining the module name “myApp”

**myCtrl.js**

1. app.controller("myCtrl", function($scope) {
2. $scope.firstName = "Pradeep";
3. $scope.lastName= "K";
4. });
5. **Directives**  
   Directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tells the AngularJS HTML compiler ($compile) to attach a specified behaviour to that DOM element or even transform the DOM element and its children. The angular directives start with “ng-”. Some of the directives is listed here with their purpose.
6. *ng-app: To bootstrap the application*
7. *ng-controller: To set a controller on a view*
8. *ng-view: Indicates the portion of the page to be updated when route changes*
9. *ng-show / ng-hide: Shows/hides the content within the directive based on the boolean equivalent of value assigned*
10. *ng-if: Places or removes the DOM elements under this directive based on the boolean equivalent of value assigned*
11. *ng-model: Enables two-way data binding on any input controls and sends the validity of data in the input control to the enclosing form*
12. *ng-class: Provides an option to assign a value of a model to CSS, conditionally apply styles and use multiple models for CSS declaratively*
13. *ng-repeat: Loops through a list of items and copies the HTML for every record in the collection*
14. *ng-options: Used with HTML select element to render options based on data in a collection*
15. *ng-href: Assigns a model as hyperlink to an anchor element*
16. *ng-src: Assigns a model to source of an image element*
17. *ng-click: To handle the click event on an element*
18. *ng-change: Requires ng-model to be present along with it. Calls the event handler or evaluates the assigned expression when there is a change to the value of the model*
19. *ng-form: Works the same as a HTML form and allows the nesting of forms*
20. *ng-non-bindable: Prevents AngularJS from compiling or binding the contents of the current DOM element*
21. *ng-repeat-start and ng-repeat-end: Repeats top-level attributes*
22. *ng-include: Loads a partial view*
23. *ng-init: Used to evaluate an expression in the current scope*
24. *ng-switch conditionally displays elements*
25. *ng-cloak to prevent Angular HTML to load before bindings are applied*

**Some sample codes for directive**

**Example-3 ng-Repeat directive**

1. <!DOCTYPE html>
2. <html>
3. <script src= "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
4. <body>
5. <div ng-app="" ng-init="names=['Pradeep','John','Kai']">
6. <p>Looping with ng-repeat:</p>
7. <ul>
8. <li ng-repeat="x in names">
9. {{ x }}
10. </li>
11. </ul>
12. </div>
13. </body>
14. </html>

**Example-4 ng-options directive**

1. **<html**
2. **xmlns="http://www.w3.org/1999/xhtml">**
3. **<head>**
4. **<title>Binding both value and text in select dropdown in Angular.js</title>**
5. **<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.9/angular.min.js"></script>**
6. **<script type="text/javascript">**
7. **var app = angular.module('sampleapp', [])**
8. **app.controller('samplecontrol', function ($scope) {**
9. **$scope.sample = [{id: '1',**
10. **name: 'Bangalore'**
11. **}, {id: '2',**
12. **name: 'Chennai'**
13. **}, {id: '3',**
14. **name: 'Kolkata'**
15. **}, {id: '4',**
16. **name: 'Mumbai'**
17. **}, {id: '5',**
18. **name: 'Delhi'**
19. **}, {id: '6',**
20. **name: 'Pune'**
21. **}];**
22. **});**
23. **</script>**
24. **</head>**
25. **<body data-ng-app="sampleapp" ng-controller="samplecontrol">**
26. **<form id="form1">**
27. **Select Name:**
28. **<select ng-model="selectedItem" ng-options="s.id as s.name for s in sample" ng-model="col">**
29. **<option value="">--Select--</option>**
30. **</select>**
31. **<p>selected item is: {{selectedItem}}</p>**
32. **</form>**
33. **</body>**
34. **</html>**

**Output**



**. Expressions**

* The {{ }} are a declarative way of specifying a data binding location in HTML. AngularJS will automatically update this text.
* Using an expression we can do data binding, as in {{ }}.
* Angular expressions can be added inside the HTML templates.
* Angular expressions doesn't support control flow statements (conditionals, loops, or exceptions).
* Angular expressions support filters to format data before displaying it.

**Example-5 Showing Expressions**

1. <!DOCTYPE html>
2. <html>
3. <script src= "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
4. <body>
5. <div ng-app="" ng-init="firstName='John'">
6. <p>Input something in the input box:</p>
7. <p>Name:
8. <input type="text" ng-model="firstName">
9. </p>
10. <p>You wrote: {{ firstName }}</p>
11. </div>
12. </body>
13. </html>

{{ firstName}} is the expression here.

**4. Controller**

* AngularJS applications are controlled by controllers.
* The ng-controller directive defines the application controller.
* A controller is a JavaScript Object, created by a standard JavaScript object constructor.
* In Angular, a Controller is a JavaScript constructor function that augments the Angular Scope.

**Example-6 Showing controller**

1. <!DOCTYPE html>
2. <html>
3. <script src= "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
4. <body>
5. <div ng-app="myApp" ng-controller="myCtrl">
6. First Name:
7. <input type="text" ng-model="firstName">
8. <br>
9. Last Name:
10. <input type="text" ng-model="lastName">
11. <br>
12. <br>
13. Full Name: {{firstName + " " + lastName}}
14. </div>
15. <script>
16. var app = angular.module('myApp', []);
17. app.controller('myCtrl', function ($scope) {
18. $scope.firstName = "Pradeep";
19. $scope.lastName = "K";
20. });
21. </script>
22. </body>
23. </html>

**5. Scope**

A $scope is a JavaScript object that is used for communication between the controller and view. Basically, $scope binds a view (DOM element) to the viewmodel and functions defined in a controller.

* Scope is the bridge between the Controller and the View.
* Scope is the data source for angular js
* scope is considered as model and ViewModel in MV-Whatever design pattern
* scope provides two-way binding when the model data changes scope data changes by notification in Angular js
* We can add and remove a property when required.
* When calling a database or any service to do a CRUD operation, all the data manipulation and assignment of data happens through a scope object.

**6. Data Binding**

Two-way data binding synchronizes data between a model and a view. In other words, any change in model will update the view and vice versa. The ng-model directive is used for two-way data binding. In the first example we saw that when the user types into the text box the changed value shows in upper and lower case in the label. This is two-way binding.

AngularJS handles the data-binding mechanism using the three powerful functions: $watch(), $digest() and $apply (). Most of the time AngularJS will call the $watch() and $digest() functions on the scope object for you, but in some cases you may need to call these functions yourself to update new values.

**7. Validations**

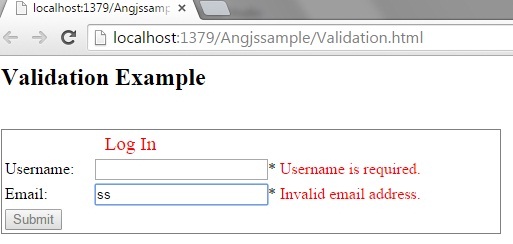
AngularJS forms and controls can validate input data.

AngularJS forms and controls can provide validation services and notify users of invalid data.

**Example-7 Showing Validations**

1. <!DOCTYPE html>
2. <html>
3. <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
4. <body>
5. <h2>Validation Example</h2>
6. <form ng-app="myApp" ng-controller="validateCtrl"
7. name="myForm" novalidate>
8. <table style="border: 1px solid gray; width: 500px;">
9. <tr>
10. <td colspan="2">
11. <span style="color: red; padding-left:100px; border: 3px;">Log In </span>
12. </td>
13. </tr>
14. <br />
15. <tr>
16. <td>Username:
17. </td>
18. <td>
19. <input type="text" name="user" ng-model="user" required>\*
20. <span style="color: red" ng-show="myForm.user.$dirty && myForm.user.$invalid">
21. <span ng-show="myForm.user.$error.required">Username is required.</span>
22. </span>
23. </td>
24. </tr>
25. <tr>
26. <td>Email:
27. </td>
28. <td>
29. <input type="email" name="email" ng-model="email" required>\*
30. <span style="color: red" ng-show="myForm.email.$dirty && myForm.email.$invalid">
31. <span ng-show="myForm.email.$error.required">Email is required.</span>
32. <span ng-show="myForm.email.$error.email">Invalid email address.</span>
33. </span>
34. </td>
35. </tr>
36. <tr>
37. <td colspan="2">
38. <p>
39. <input type="submit"
40. ng-disabled="myForm.user.$dirty && myForm.user.$invalid ||
41. myForm.email.$dirty && myForm.email.$invalid">
42. </p>
43. </td>
44. </tr>
45. </table>
46. </form>
47. <script>
48. var app = angular.module('myApp', []);
49. app.controller('validateCtrl', function ($scope) {
50. $scope.user = 'Pradeep K';
51. $scope.email = 'Pradeep.k@gmail.com';
52. });
53. </script>
54. </body>
55. </html>

**Output**



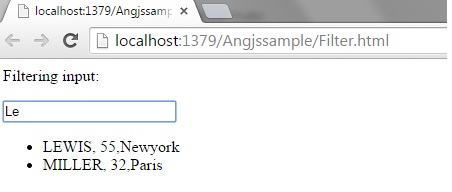
**8. Filters**

* Filters are used to extend the behaviour of binding expressions and directives.
* Filters allow formatting the data to be displayed on the DOM.
* Filters are invoked in the HTML with the | (pipe) character inside expressions {{ }}.
* They are used to format values or to apply certain conditions. They are executed whenever the value bound in the binding expression is updated.

**Example-7 Showing Filter example**

1. <html>
2. <head>
3. <script src= "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
4. <script>
5. var app = angular.module('myApp', []);
6. app.controller("namesCtrl", function ($scope) {
7. $scope.friends = [
8. { name: "Karl", age: 27, city: "Bangalore" },
9. { name: "Lewis", age: 55, city: "Newyork" },
10. { name: "Adarsh", age: 20, city: "London" },
11. { name: "John", age: 21, city: "Newyork" },
12. { name: "David", age: 20, city: "Chenai" },
13. { name: "Miller", age: 32, city: "Paris" }
14. ];
15. });
16. </script>
17. </head>
18. <div ng-app="myApp" ng-controller="namesCtrl">
19. <p>Filtering input:</p>
20. <p>
21. <input type="text" ng-model="test">
22. </p>
23. <ul>
24. <li ng-repeat="x in friends | filter:test | orderBy:'name'">
25. {{ (x.name | uppercase) + ', ' + x.age +','+x.city }}
26. </li>
27. </ul>
28. </div>
29. </html>

**Output**



**9. Services**

If you need to share state across your application, or need a solution for data storage or cache, think of Services. Services are singletons and can be used by other components such as directives, controllers, filters and even other services. Services do not have a scope of their own, so it is permissible to add eventlisteners in Services using $rootScope.

There are the following few services that we use frequently:

* *$http – Used for Ajax calls for server communications*
* *$window-Provide a reference to a DOM object.*
* *$Location- Provides reference to the browser location*
* *$timeout –Provides a reference to window.settimeout function*
* *$Log- Used for logging*
* *$sanitize- used to avoid script injections and display raw html in page*
* *$Rootscope –Used for scope hierarchy manipulation*
* *$Route –used to display browser based path in browser URL.*
* *$Filter –Used for providing filter access.*
* *$resource- used to work with Restful API*
* *$document – used to access window. Document object*
* *$exceptionHandler- used for handling exceptions.*
* *$q- provides a promise object*

**10. Routing**

AngularJS Routing helps you to divide your app into multiple views and bind multiple views to Controllers. The magic of Routing is taken care by an AngularJS service $routeProvider. The $routeProvider service provides the methods when() and otherwise() to define the routes for your app. Routing has dependency on the ngRoute module.

It helps us to divide a Single Page Application (SPA) into multiple views. Dividing a SPA into multiple views helps to logically divide the app and make it more manageable. Routing in an Angular application is taken care of by a service provider known as “$routeProvider”. The $routeProvider is configured in app the module's config() function. $routeProvider provides the following two methods:

1. when(): that matches a pattern
2. otherwise(): otherwise() method to define a
3. **default** route.
4. App.config(['$routeProvider',
5. function($routeProvider)
6. {
7. $routeProvider.
8. when('/List',
9. {
10. templateUrl: 'Views/list.html',
11. controller: 'ListController'
12. }).
13. when('/Add',
14. {
15. templateUrl: 'Views/add.html',
16. controller: 'AddController'
17. }).
18. otherwise({
19. redirectTo: '/List'
20. });
21. }]);

**11. Dependency Injection**

Dependency Injection (DI) is a software design pattern that deals with how components get hold of their dependencies. Dependency injection helps us to create a loosely-coupled architecture. Dependency Injection is designed on the following two principles.

*A. High-level modules should not depend on low-level modules. Both should depend on abstractions.*

*B. Abstractions should not depend on details. Details should depend on abstractions.*

The Angular injector subsystem is in charge of creating components, resolving their dependencies and providing them to other components when needed. Components such as services, directives, filters and animations are defined by an injectable factory method or constructor function. These components can be injected with "service" and "value" components as dependencies. AngularJS has a built-in dependency injector that keeps track of all components (services, values and so on) and returns instances of needed components using dependency injection. The Dependency injector of AngularJS works based on names of the components.

A simple case of dependency injection in Angular js:

1. AppModule.controller("AppController", **function**($scope, $window, $log,$http)
2. {
4. });

Here, $scope, $window and $log, $http are passed into the controller using dependency injection.

**12. Testing**

As said before, Angular components supports dependency injections. This dependency injections allows Angular codes to be testable. Use any kind of component in Angular, it can't be written without getting some of the external components injected in. The testing framework like Jasmine and Karma are the most widely used testing framework with Angular. These two frameworks support mocking and are highly configurable using a JSON file and the use of various plug-ins.