**AngularJS**

* Powerful javascript framework, mainly used for single page applications
* First version was released on 2011 and is supported by google.
* Extends HTML DOM with additional attributes with **Directives** and binds data to html with **expressions**
* Compact and makes it easy to write javascript.
* AngularJS is not a DOM manipulation library like jQuery, but it uses a subset of jQuery called jQLite.
  + A library performs specific, well-defined operations.
  + A framework is a skeleton where the application defines the "meat" of the operation by filling out the skeleton.
* AngularJS is primarily based on HTML attributes that you can add to your HTML tags
* Angular is good fit for CRUD (Create, Read, Update, Delete) applications
* Not a good fit for applications which involves graphics, and tricky DOM manipulation, for such kind of application jQuery and similar libraries would be a good fit
* Angular provides browser new syntax through directives, which includes
  + **Data binding**
  + **DOM control structures for repeating, showing, hiding DOM fragments**
  + **Support for forms and form validation**
  + **Attaching new behavior to DOM elements**
  + **Grouping of HTML into reusable components**
* Cross-browser compliant
* Download and install angularjs from <https://angularjs.org/>
* Version used to be used
  + <https://code.angularjs.org/1.5.5/angular.min.js>
* Install http-server
  + Install Node.js
  + **npm install http-server -g** // install http-server
    - **http-server** //starts http-server

**Directives**

* As said earlier angularjs extends html with the help of directives which is denoted by ng-directives
* Directives are makers in your dom element which helps to attach specific behavior / transform dom elements
* Three most important directives for an angularjs application
* **ng-app :** defines an angularjs application
* **ng-model :** binds value of html form elements to application data // works only with selected input elements
* **ng-bind :** binds application data to html view
* **ng** stands for angular
* These basic directives makes html file dynamic without writing any scripts
* <div ng-app="">  
   <input type="text" ng-model="name\_input" />  
   <p ng-bind="name\_input"></p>  
   <p ng-bind="name\_input">{{name\_input}}</p>  
   </div>
* ng-app tells that <div> element is the owner of the application
* ng-model binds value of input element to application variable **name\_input**
* ng-bind binds the html of **<p>** element to application variable **name\_input**
* **ng-init :** initializes angularjs variables
  + <div ng-app="" ng-init="firstname='Arungopan'; lastname='Gopakumar'">  
     <p ng-bind="'Your name is :'+firstname +' '+lastname"></p>  
     </div>
* You can we directives in angularjs application starts with **ng- ,** you could also use **data-ng-** instead of **ng-** to make you html valid.

**Expressions**

* Used to bind data to html
* Can be written inside double braces **{{expression}}**
* Behaves the same way as **ng-bind** directives
* Outputs the data where they are used.
* <div ng-init="firstname = 'Arungopan'; lastname = 'Gopakumar'">  
   <p>My name is : {{ firstname + ' ' + lastname}}</p>  
   <p>Operators : {{10 + 10}}</p>  
   <input type=”text” ng-model="firstname" />{{firstname}}  
  </div>
* **Using Objects :**
  + <div ng-app="" ng-init="user\_obj={firstname:'Arungopan',lastname:'Gopakumar'}">  
      
     <p>Using bind <span ng-bind="user\_obj.firstname + ' ' + user\_obj.lastname"></span></p> <br />  
     <p>Using expression <span> {{user\_obj.firstname + ' ' + user\_obj.lastname}}</span></p>  
      
     </div>
* **Using Arrays :** 
  + <div ng-app="" ng-init="users=['Arungopan', 'Gopakumar', 'Naveen', 'Nandakumar']">  
      
    <p>Result 1 : {{ users[0] }}</p>  
    <p>Result 2 : {{ users[1] }}</p>  
    <p>Result 3 : {{ users[2] }}</p>  
    <p>Result 4 : {{ users[3] }}</p>  
    </div>

**ng-repeat**

* Directive which helps to repeat set of data.
* Helps you to have loops and thereby helping repeat html content without writing a single line of script
  + <div ng-init="users\_obj=[{'name':'Arun', 'location':'India'}, {'name':'Nanda', 'location':'US'}, {'name':'Naveen', 'location':'US'}]">   
     <p ng-repeat="user in users\_obj">  
     Name : {{user.name}} <br />  
     Location : {{user.location}}  
     </p>  
    </div>

**HTML DOM**

Following directives can be used to bind application to html elements

* ng-disabled : disables input elements
  + <input type = "checkbox" ng-model="disableBtn">Disable Button  
     <button ng-disabled="disableBtn">Button 1</button>
* ng-show : show an element which is hidden
  + <input type="checkbox" ng-model="showBtn">Show Button  
     <button ng-show="showBtn">Button</button>
* ng-hide : hides an element which is visible
  + <input type="checkbox" ng-model="hideBtn">Hide Button  
     <button ng-hide="hideBtn">Button</button>
* ng-click : binds a click event to an element
  + <p>Total click: {{ cntr }}</p>  
     <button ng-click = "cntr = cntr + 1">Update</button>
  + You can even pass a function to ng-click, which we will later.
* ng-change :   
  + <select ng-model="sel\_options" ng-change="sel\_val2(sel\_options)">  
     <option>--Select--</option>  
     <option ng-repeat="user in user\_obj" value="{{user.age}}">  
     {{user.name}}  
     </option>  
     </select>
    - $scope.sel\_val2 = function(option\_sel) {  
       alert(option\_sel);  
       };
  + <select ng-model="option\_sel"  
     ng-options="user.name for user in user\_obj"  
     ng-change="sel\_val(option\_sel)"  
     >  
     </select>  
    - $scope.sel\_val = function(option\_sel) {  
       alert(option\_sel.age);  
       };

**MVC (Model View Controller)**

* Is a software design pattern for developing web applications.
* **Model** is the lowest level in the pattern and maintains data
* **View** displays data to the user
* **Controller** Controls the interactions between model and view, business logic
* **Module** is a container for the different parts of your app – controllers, services, filters, directives, etc

**Controllers**

* Controls data of angularjs application
* ng-controller directive defines the application controller
* Controller is just a javascript object created using javascript constructor function
  + <div ng-controller="sports">  
     Select your favourite sport :  
     <input type="radio" name="sport\_selctn" ng-click="select\_sports($event)" value="Cricket" />Cricket  
     <input type="radio" name="sport\_selctn" ng-click="select\_sports($event)" value="Football" />Football  
     <input type="radio" name="sport\_selctn" ng-click="select\_sports($event)" value="Hockey" />Hockey  
     <input type="radio" name="sport\_selctn" ng-click="select\_sports($event)" value="Tennis" />Tennis  
     <div>Your favourite sport is : {{fav\_sport}}</div>  
     </div>  
    <script>  
     var my\_application = angular.module('my\_app', []);  
     my\_application.controller('sports', function($scope) {  
     $scope.fa\_sport = '';  
     $scope.select\_sports = function($event) {  
     $scope.fav\_sport = $event.target.value;  
     }  
     });  
      
     my\_application.controller('sports', [‘$scope’, function($scope) {   
     }]);  
       
     </script>
* In the above example we have created a controller **sports** using ng-controller which is a angularjs directive for creating controller
* **sports** is a javascript function which invoke the controller using **$scope** object.
* **$scope** is the application object, has access to all variables and functions within the controller, and event handlers attached.
* **$scope** here creates a function **select\_sports** which accepts **$event** object as argument and access the value of the current element using $event and updates the value of **fav\_sport**
* **Rootscope : $rootScope** 
  + Is available in the entire application
  + If the same variable is available in the both the scope, then application uses the one in the current element

**Filters**

* Way of formatting the output
* Filters can be added to expression by using pipe character “|” followed by filter
  + uppercase
  + lowercase
  + number
    - <input ng-model="ngfilter" />  
      <div>{{ngfilter | uppercase}}</div>
  + currency
    - <div>{{ngfilter | currency:"Rs. ":3}}</div>
  + orderBy
    - <ul>  
       <li ng-repeat="user\_dtls in fa\_users | orderBy:'age':'reverse'">  
       {{'Name : '+user\_dtls.name +', Age : '+user\_dtls.age +', Location : '+user\_dtls.location}}  
       </li>  
       </ul>
  + Filter
    - Return the subset of an array containing the matching items
    - ng-repeat="user\_dtls in fa\_users | filter:'u'"
    - ng-repeat="user\_dtls in fa\_users | filter:'India':true" // for exact match
    - ng-repeat="user\_dtls in fa\_users | filter:custom\_filter" // use custom\_filter in ng-model
    - Multiple filters separated using | :
      * <li ng-repeat='user\_dtls in users | orderBy:"location":"reverse" | filter:custom\_filter'>  
         {{' Name = '+user\_dtls.name}}  
        </li>
  + Custom filters
    - Creating user defined filters use **.filter()**
    - app.filter('myFilter', function() {  
       return function (value) {  
       return 'Username is = '+value.toUpperCase();  
       };  
      });  
        
      app.controller('my\_cntrl', ['$scope', function($scope) {  
       $scope.users = [  
       'Nanda', 'Arun', 'Hang', 'Naveen'  
       ];  
      }]);

**Routing**

* Way to handle multiple url context in a website.
* **ng-view :** container where all the data is shown
* While creating module we need to specify the dependencies **ngRoute**
* **$routeProvider :** configure mapping of controllers and views to urls
  + $route watches the url and directs browser to correct route
* **$routeParams :** to capture retrieve parameters being passed on URL
* **$route service** is configured using the provider function internally, and that is the reason why we can use the service inside the config function
* **config (**function which runs at the configuration phase**)** function to specify different url
  + **$routeProvider** serviceprovides method when() and otherwise() to define the routes for your app
  + **.when** function defines the url action on what to be shown when a url is loaded
  + **template** in .when function if the content you need to change is just a static text
  + **templateUrl** if you need to load data from a url
  + **.otherwise()** : to capture all the invalid url access
    - redirectTo: redirects to a particular action
* **Resolve property :** let's you to specify a list of dependencies that will be injected into the controller, if you specify a promise in the resolve property, the controller will wait for the promise to resolve.
* Sample example for routing given below
* var app = angular.module('my\_app', ['ngRoute']);  
    
  app.config(function($routeProvider) {  
   $routeProvider.when('/', {  
   template:'Hello every one, welcome to my site'  
   })  
   .when('/home', {  
   template:'This is my website home page !!!!'  
   })  
   .when('/profile', {  
   templateUrl:'routes.html'  
   })  
   .when('/users', {  
   templateUrl:'users.html',  
   controller:'users'  
   })  
   .when('/users/:id', {  
   templateUrl:'users\_details.html',  
   controller:'users\_details'  
   })  
   .when('/logout', {  
   redirectTo:'/'  
   })  
   .when('/error', {  
   template:'The page you have requested cannot be found. Please try later..!!!'  
   })  
   .otherwise({  
   redirectTo: '/error'  
   });  
  });  
    
  app.controller('my\_cntrl', ['$scope', '$location',function($scope, $location) {  
     
  }]);  
    
  app.controller('users', ['$scope', '$rootScope', 'getData', function($scope, $rootScope, getData) {  
   console.log(getData);  
   $scope.users = [  
   {  
   "name":"Arun",  
   "loc":"India",  
   "email":"arun@marlabs.com"  
   },  
   {  
   "name":"Nanda",  
   "loc":"US",  
   "email":"nanda@marlabs.com"  
   },  
   {  
   "name":"Naveen",  
   "loc":"Canada",  
   "email":"naveen@marlabs.com"  
   }  
   ];  
   $rootScope.users = $scope.users;  
  }]);  
    
  app.controller('users\_details', function($scope, $routeParams, $rootScope) {  
   $scope.user\_details = $rootScope.users[$routeParams.id];  
  });
* Html code would look like
* <body ng-app="my\_app" ng-controller="my\_cntrl">  
   <a href="#/home">Home</a> |   
   <a href="#/profile">Profile</a> |   
   <a href="#/users">User List</a> |   
   <a href="#/logout">logout</a>  
   <br /><br />  
   <div ng-view=""></div>  
   </body>

**Dependency Injection**

* Dependency Injection is a software design pattern in which components are given their dependencies instead of hard coding them within the component.

**Services**

* Services are javascript functions and are responsible to do a specific tasks
* Best way to implement reusable code
* Clean and maintainable code
* To use the service in the controller, it must be defined as a dependency.
  + **$location :** returns information about the location of a web page.
    - $location.path(); to redirect to a url
  + **$http :** most widely used service, makes request to the server.
  + **$timeout :** equivalent to **window.setTimeout** function in javascript
  + **$interval :** equivalent to **window.setInterval** function in javascript
  + **$log :** Simple service for logging message to browser console. Similar to console.log()
    - **$log.log()**
      * Writes a log message
    - **$log.info()**
      * Writes a informatio message
    - **$log.warn()**
      * Write a warning message
    - **$log.error()**
      * Write an error message
    - **$log.debug()**
      * Write a debug message

**$watch**

* Used to watch the changes in a $scope variable
* When you create a data binding in your view to a variable in the scope object, angular create a watch internally
* Created using **$scope.$watch()**
* $scope.$watch('text\_bx', function(new\_val, old\_val) {  
   $scope.new\_val = new\_val;  
   $scope.old\_val = old\_val;  
   });

**$digest**

* Created using $scope.$digest()
* Iterates through all the watches and calls the value function and checks if there is any difference from the value returned last time, if it finds a difference in the value then listener function for watch is called.
* In some case, if you are using javascript to update the $scope variables, then angular does not call the $digest() function automatically. In such case you will have to call $scope.$digest() (or $scope.apply() which will inturn call $digest()) explicitly.
* $scope.my\_name = 'Arungopan';  
   document.getElementById('my\_btn').addEventListener('click', function() {  
   $scope.my\_name = 'Jacob';  
   $scope.$digest();  
   });

**Custom directives**

* You can create your own directives using **.directive** function  
  + var my\_app = angular.module('my\_app', []);  
     my\_app.directive('testDirective', function() {  
     return {  
     template : "This is a custom directive",  
     templateUrl : "http://localhost/training/angularjs/custom\_directives/custom\_data.html"  
     }  
     });
* Can invoke by
  + Html tag with directive name
    - <test-directive></test-directive>
  + Attribute
    - <div test-directive></div>
  + Class name
    - <div class=”test-directive”></div>
* Restrict
  + Restricts directive to be only be invoked by a particular method
  + Different values for restrict
    - E for Element name
    - A for Attribute
    - C for Class
* Link property : attach dom listeners.
  + Angularjs comes with jqlite library by default.
* Directives by default has shared scope
* Scope property : used to change scope, different values
  + true : local scope
  + false : global scope, default scope
  + {} : isolated scope prevent prototype inheritance, completely detached from the parent scope, does not inherit.
* Isolated scope attributes
  + @ : for text binding
  + = : two way binding
  + & : one way binding, used to pass functions to directive
* Example for custom directive
* **script.js file**
* var app = angular.module('my\_app', []);  
    
  var app = angular.module('my\_app', []);  
    
  app.directive('myDirective', function($compile) {  
     
   return {  
   templateUrl:'template\_file.html',  
   restrict:'EAC',   
   scope:{  
   username:'@',  
   location:'=',  
   dirfn:'&'  
   },  
   link:function(scope, elems, attrs) {  
   scope.scope\_var = "Hello Arun here....!!!";  
   document.querySelector('#abc').addEventListener('click', function() {  
   document.getElementById('abc\_cntr').innerHTML = '<div>{{scope\_var}}</div>';  
   $compile(elems.contents())(scope);  
   });  
   }  
   };  
     
  });  
    
  app.controller('my\_cntrl', function($scope) {  
   $scope.user\_info = {  
   "name":"Arun",  
   "loc":"United States",  
   "display\_details":function() {  
   alert(this.name+", "+this.loc);  
   }  
   };  
  });
* **template\_file.html**
* <div style="padding:10px;background-color:black;color:white">  
   This is my first custom directive <br /><br />  
     
   Name = <input type="text" ng-model="username" />, <br /><br />  
     
   Location = <input type="text" ng-model="location" />, <br /><br />  
     
   <button ng-click="dirfn()">Alert details</button>  
   <br /><br />  
   <button id="abc">Add html</button>  
   <br /><br />  
   <div id="abc\_cntr"></div>  
   <br /><br />  
   Scope variable : {{scope\_var}}  
  </div>
* **index.html file**
* <html>  
   <head>  
   <title>Angular</title>  
   <script src="https://code.angularjs.org/1.5.5/angular.min.js"></script>  
   <script src="script.js"></script>  
   </head>  
   <body>  
   <div ng-app="my\_app" ng-controller="my\_cntrl">  
   <my-directive username="{{user\_info.name}}" location="user\_info.loc" dirfn="user\_info.display\_details()"></my-directive><br /><br />  
   {{user\_info.name}}<br /><br />  
   {{user\_info.loc}} <br /><br />  
   </div>  
   </body>  
  </html>

**$compile :**

* At the lowest level angular has the compilation cycle.
* When the page loads, angularjs is then loaded
* Angular then listens the ready event and goes and looks for the application in your html, which is the ng-app.
* Angular will then compile all the service, controller etc in your module and then goes through the DOM and check all the directives used and generates a template out of that.
* And it links all together, it this template is getting this scope and binds it together and view is ready.

HTML **=>** DOM **=>** DOM LOAD EVENT **=>** NG-APP MODULE **=>** USE INJECTOR SERVICE TO INJECT ALL THE SERVICES **=>** $COMPILE **=>** DYNAMIC DOM

**Example for a compile function:**

var my\_app = angular.module('my\_app', []);

**// adding new element in the directive dynamically.**

my\_app.directive('contentItem', function($compile) {

return {

link: function(scope, element){

var template = "<button ng-click='testfn()'>{{btn\_label}}</button>";

element.html(template);

**$compile(element.contents())(scope);**

}

}

});

my\_app.controller('my\_cntrl', function($scope) {

$scope.btn\_label = "My button";

$scope.testfn = function() {

alert('Hello Marlabs');

}

});

**Display html data using $sce service**

Service needs to injected just like all the other service in order to use it.

$scope.display\_html = **$sce.trustAsHtml**("<div style='border:1px solid #000'>Hello everyone..!!</div>");

<span ng-bind-html="display\_html"></span>