## **What is AngularJS?**

AngularJS is a JavaScript framework that is used for making rich, extensible web applications. It runs on plain JavaScript and HTML, so you don't need any other dependencies to make it work, and it is CSS-agnostic so you can use whatever CSS framework/methodology you want when designing your Angular application.

## **How is an Angular app architectured?**

Angular applications at their most basic consist of three components.

### **The template**

The template is the HTML portion of the app. Writing a template is exactly like writing a static HTML page, except that templates contain additional syntax which allows data to be injected into them in order to provide a customized user experience. If you have ever written an HTML page using a server-side web framework you'll feel right at home writing templates in Angular. The feature that differentiates Angular templates from server-generated pages, however, is that in Angular data can be injected, modified and removed from templates without ever requiring a page refresh. This feature provides a more fluid experience to the end user and enhances the overall feel of Angular web applications.

### **The scope**

The scope is a very important component in Angular applications. The scope is the object that represents the "model" of your application. It contains fields that store data which is presented to the user via the template, as well as functions which can be called when the user performs certain actions such as clicking a button.

### **The controller**

The controller plays somewhat of a supporting role in Angular applications. The controller is a function which generally takes an empty scope object as a parameter and adds to it the fields and functions that will be later exposed to the user via the view.

## **ngModel**

Unlike values bound using the double curly brace syntax, ngModel allows us to bind values to HTML elements such as input fields. When using ngModel, not only are changes in the scope reflected in the view, **but changes in the view are reflected back into the scope.**

Use the ng-model directive to bind data from the model to the view on HTML controls (input, select, textarea)

### **Example**

<input ng-model="firstname">

Ref: <http://www.w3schools.com/angular/angular_databinding.asp>

**ngClass**

To bind a CSS class to an HTML element, we use the ngClass directive. ngClass takes as input an expression which must evaluate to one of the following.

* A string of space-delimited class names.
* An array of class names.
* A map (object) where the keys are class names and the values are boolean values indicating whether or not to apply the class.

Ex; To change the class based on the health value i.e fetched dynamically, use the following code snippet:

**Template**

**<tr data-ng-repeat="value in values" ng-class="checkhealth(value.overall\_health)"**>

<**td**>{{value.id}}</**td**>

<**td**>{{value.id}}</**td**>

<**td**>{{value.actual\_end\_time}}</**td**>

<**td**>{{value.overall\_health}}</**td**>

</**tr**>

**Angular JS Script**

app.controller(**"top"**, [**'$scope'**, **'Restangular'**, **function**($scope, Restangular){

console.log(**"Display the top 5 critical resources"**);

$scope.test = **'test12'**;

**var** resource\_list = Restangular.all(**"Top5Resource"**);

resource\_list.getList().then(**function**(projects){

$scope.values = projects;

console.log($scope.values);

})

**$scope.checkhealth = function(health) {**

**var className = '';**

**console.log(health);**

**if (health < 30) {**

**className = 'red';**

**}**

**else {**

**className = 'yellow';**

**}**

**console.log(console.log);**

**return className;**

**};**

**}])**

**References:**

<http://www.learn-angular.org/#!/lessons/binding-css-classes>

<http://stackoverflow.com/questions/41731533/change-td-style-based-on-value-in-angular>

**$Watch**

$watch provides us with a way to keep calculated values up to date when the values that they depend on change. The syntax for $watch looks like this.

$scope.$watch([expression returning watched value],  
 [change handler],  
 [objectEquality?]);

**Ref:** <http://www.learn-angular.org/#!/lessons/watch>

## **AngularJS $http**

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

### **Example**

Make a simple request to the server, and display the result in a header:

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

<p>Today's welcome message is:</p>

<h1>{{myWelcome}}</h1>

</div>

<script>

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

app.controller('myCtrl', function($scope, $http) {

$http.get("welcome.htm")

.then(function(response) {

$scope.myWelcome = response.data;

});

});

</script>

# **Data Validation**

Form validation is a key component of all but the simplest of web applications. AngularJS once again has us covered by providing a built in framework to tell users whether or not the data they have entered into a form is valid. It does this by applying a subset of the following four CSS classes.

* ng-valid
* ng-invalid
* ng-pristine
* ng-dirty

As developers, we can then go ahead and write CSS rules to apply different styles to fields and forms depending on their state.

Ref: <http://www.learn-angular.org/#!/lessons/data-validation>

## **AngularJS Filters**

AngularJS provides filters to transform data:

* currency Format a number to a currency format.
* date Format a date to a specified format.
* filter Select a subset of items from an array.
* json Format an object to a JSON string.
* limitTo Limits an array/string, into a specified number of elements/characters.
* lowercase Format a string to lower case.
* number Format a number to a string.
* orderBy Orders an array by an expression.
* uppercase Format a string to upper case.

Ref: <http://www.w3schools.com/angular/angular_filters.asp>

# **Repeaters**

It often happens that you need to display a collection of data to the user. For this task, Angular provides you with the ngRepeatdirective. ngRepeat gives you the power to define a template for a single item in a collection, and then have it be repeated for all the elements in a collection.

The syntax for ngRepeat is super easy. In attribute form, it looks like this.

ng-repeat="element in collection"

Reference ;

<http://www.learn-angular.org/#!/lessons/repeaters>

**ngShow and ngHide**

if you simply want to show or hide an element, there is a shortcut. The ngShow and ngHide directives display or hide an element from view by manipulating its display CSS property. They both take an expression as an argument which is expected to return a Boolean value.

**ng-if conditions :**

ng-if statement checks the statement is true or false and execute statement accordingly.

<li ng-repeat="rules in reportlogs.rulelist" ng-if="rules.rule\_status=='success'">{{rules.id}}</li>

**Filters:**

Filters can be added to expressions by using the pipe character |, followed by a filter.

1) The uppercase filter format strings to uppercase:

<div ng-app="myApp" ng-controller="personCtrl">

<p>The name is {{ lastName | uppercase }}</p>

</div>

2) The lowercase filter format strings to uppercase:

<div ng-app="myApp" ng-controller="personCtrl">

<p>The name is {{ lastName | lowercase }}</p>

</div>

3)The orderBy filter sorts an array:

<div ng-app="myApp" ng-controller="namesCtrl">

<ul>

<li ng-repeat="x in names | orderBy:'country'">

{{ x.name + ', ' + x.country }}

</li>

</ul>

</div>

<script>

angular.module('myApp', []).controller('namesCtrl', function($scope) {

$scope.names = [

{name:'Jani',country:'Norway'},

{name:'Carl',country:'Sweden'},

{name:'Margareth',country:'England'},

{name:'Hege',country:'Norway'},

{name:'Joe',country:'Denmark'},

{name:'Gustav',country:'Sweden'},

{name:'Birgit',country:'Denmark'},

{name:'Mary',country:'England'},

{name:'Kai',country:'Norway'}

];

});

</script>