**Angular JS**

Angular JS is a framework that **extends** HTML attributes with **Directives**, and **binds data** to HTML with **Expressions**.

It is included as an external script file to an HTML page like so.

<scriptsrc="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

**Directives**

AngularJS extends HTML with ng-directives. All Directives use the **ng prefix**.

Ex.

<div ng-app="" ng-init="firstName='John'”>  
  <p>Name: <input type="text" ng-model="name"></p>  
  <p ng-bind="name"></p>  
</div>

**ng-app:** This directive is used to define an AngularJS application. It is used as an attribute in an HTML file body like so.

It tells the div element that it is the “owner” of an AngularJS application.

You must have an overall ng-app layer, and then if you want more ng-app layers, you must have them as child elements.

**ng-model**: directive that binds the value of HTML controls (inputs, select, textarea) to application data

In this example we are binding the text input into the text input to the variable “name”

**ng-bind:** directive that binds application data to the HTML view.

This example binds the innerHTML of the <p> tag to the variable “name”

**ng-init:** this directive initialize AngularJS application variables.

ng-init is not very common, it is better to initialize data in controllers.

Even Objects and arrays can be initialized.

NOTE: In order to make the html page a valid html

**Expressions**

AngularJS expressions are written in double braces: {{ expression }}.

It must be a part of the ng-app directive to resolve the expression. If the ng-app directive is not there, the expression is displayed as it is.

It will output the data exactly where the expression is written.

<div ng-app="">  
  <p>My first expression: {{ 5 + 5 }}</p>  
</div>

AngularJS expressions are much like JavaScript expressions, they can have literals, operators, and variables.

Expressions bind AngularJS data to HTML the same way as the ng-bind directive does.

They can even be written inside the directive itself.

ng-bind="expression"

<div ng-app="">  
  <p>Name: <input type="text" ng-model="name"></p>  
  <p>{{name}}</p>  
</div>

Object initialization example.

<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">  
<p>The name is {{ person.lastName }}</p>  
</div>

**General Application Info**

AngularJS Modules Define the AngularJS applications. The module name is placed in the ng-app parameter.

AngularJS Controllers control the application

This examples module is myApp.

The ng-app directive defines the application, ng-controller directive defines the controller

<div ng-app="**myApp**" ng-controller="**myCtrl**">  
  
First Name: <input type="text" ng-model="firstName"><br>  
Last Name: <input type="text" ng-model="lastName"><br>  
<br>  
Full Name: {{firstName + " " + lastName}}  
</div>  
  
<script>  
var app = angular.module('**myApp**', []);  
app.controller('**myCtrl**', function($scope) {  
    $scope.firstName= "John";  
    $scope.lastName= "Doe";  
});  
</script>

Modules define applications. While controllers control applications.

**Namespaces**

AngularJS modules keep all functions and variables local to their modules.

What is the scope variable.

If we consider an AngularJS application to consist of:

* View, which is the HTML.
* Model, which is the data available for the current view.
* Controller, which is the JavaScript function that makes/changes/removes/controls the data.

Then the scope is the Model.

The scope is a JavaScript object with properties and methods, which are available for both the view and the controller.