\*\*Our first angular app\*\*  
  
Module  
  
 a module is the core property of angular. Using this module we define our app.  
  
 syntax:  
 angular.module(<name of our app in string>,<array of dependencies>);  
 examle:  
 angular.module('myApp',[]);  
2.dependency injuction  
 this is a unique feature introduced in angular where we can access a function or  
 an object from one js file to another js  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 MVC approach of angular app  
  
 1. Data binding:  
  
 there are lets say some input fields inside our html. then our controller in  
 angular should look upon them and record the values what user types.  
  
  
 M --> Model inside our html  
  
  
 V --> view as our html itself  
  
  
 C --> controller this looks for every change in our view  
  
 <!-------------------------------------------------->  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
M --> model  
  
v --> view  
  
vm --> view model  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
To define a module we write like this:  
  
 angular.module('myApp',[]);  
  
 to call a module, we write like this:  
  
 angular.module('myApp');  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
creating controllers:  
  
(function(angular){  
  
 function ProjectController ('$scope'){  
 console.log('$scope')  
 }  
  
//injecting contoller  
ProjectController.$inject = ['$scope'];  
  
//calling module  
angular.module('ProjectApp').controller('projectController',ProjectController);  
})(window.angular || (window.angular = {}));  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
  
we can create any number of modules but we have to injuct in app.js  
  
\*\*we can nest controllers but we should not nest ng app  
  
\*\*for one way data binding we use ng-bind  
  
\*\*for two way data binding we use ng-model  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
one way data binding:  
  
 Getting the data from the controller and putting the same onto the html is what we call one way  
 data binding  
 there are two ways of performing one way data binding:  
 1.ng-bind  
 sytax:  
 <span ng-bind="<pass the property of vm>"</span>  
  
 example:  
 <span ng-bind="user.firstName"></span>  
  
 2.Angular exressions.  
 we can alternativerly use angular expressions these are basically double curly  
 braces to bind the data  
  
 syntax:  
 <span>{{<pass the property of vm>}}</span>  
  
 example:  
 <span>{{user.firstName}}</span>  
  
 \*\*the issue with the angular expressions is that unitll the data is loaded in the controller the html shows  
 empty curly braces  
 this is a bad user experience.  
  
 \*\*preferred way of performing one way data binding using ng-bind attribute  
  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 \*\*two way data binding:  
  
 the concept of sending the data form the hmtl to controller and back from the html is what we call  
 as two way data binding since there is only the form fields that we use for user to input some data we  
 can make use on angular ng-model to perform two way data binding.  
  
 syntax:  
 <input type="text" ng-model="<pass the property of vm>"/>  
  
 exmple:  
 <input type="text" ng-model="user.firstName"/>  
  
 \*\*only ng-model attribute is used ot perform two way data binding.  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\*\*Routing in Angularjs\*\*  
  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

services  
  
services are a special type of functions or objects that can be shared or stored a data across  
the applications the same services can also be used to perform business logic that is only access by the controller  
  
  
\*\*my controller is only connected to view model  
should not connect to the business logic. because controller job to only controlled the view model only  
  
\*\*we should not be writing the business logic in the controller. since controller is directly connected to the view  
it should only initialize the view model it should not perform any business logic  
  
there are two main types of services in angular  
  
 1.single service (in angular it’s called FACTORY)  
 this service is an object that should be throughout the application  
 or  
 this service is an object that can be reused through the application  
  
 syntax:  
 angular. Module(<module name>).factory(<name of the factory>,<call back function for the service>);  
  
 exmple:  
 //iife construct  
 (function (angular) {  
 'use strict';  
 //callback for the factory service type  
 function UserFactroy() {  
 }  
 //Injecting the dependencies  
 UserFactroy.$inject = [];  
 angular.module('projectApp.services').factory('UserFactory',UserFactroy);  
 })(window.angular || (window.angular = {}));  
  
\*\*angular introduced 'angular. Copy' to perform deeply linked copy of this is similar to object.create() in plan js.  
  
\*\*to add another property to existing property we use 'angular.extend'  
  
2.service type  
  
 this is used to write our business logic.. this can be treated as a factory type if needed.  
  
 syntax:  
 angular.module(<module name>).services(<name of the factory>,<call back function for the service>);  
  
\*\*service type is always treated like a class!!  
  
  
---what are the different types of services supports?  
  
factory type: to share the data between 2 controllers. other functions  
  
service type: is to write business logic.()  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-  
\*\*MVVM architecture\*\*  
  
1. we treat our controller as a class which will only initialize the view model.  
2 all the business logic will sit inside the services.  
3. we then share the services across the application.  
 this way we are indirectly eliminating the use of $scope and $rootScope  
  
 to setup the mvvm approach in angular follow these steps:  
  
 1. in app.js  
 whenever we are writing the routes inside the controller make sure to add the alias ot  
 additional property called 'controller as'  
  
 $routeProvider  
 .when('/home',{  
 templateUrl:'../template/home.temp.html',  
 controller: 'HomeControler as HomeCtrl'  
 });  
  
 or  
 $routeProvider  
 .when('/home',{  
 templateUrl:'../template/home.temp.html',  
 controller: 'HomeControler';  
 controlllerAs: 'HomeCtrl'  
 });  
  
if there are no routes and we are writing controller in the html itself  
the we write like this  
  
<div ng-controler= "HomeController"></div>  
  
of  
  
<div ng-controler= "HomeController as HomeCtrl"></div>  
  
  
2.In html  
 now we have create an alias (namespace) for our controller we need to make use of these  
 alias to bind our data.  
  
 example:  
 change  
 <input type="text" id="fName" name="fName" ng-model="user.firstName"/>  
 To  
 <input type="text" id="fName" name="fName" ng-model="HomeCtrl.user.firstName"/>  
  
3.In controller.js  
 we need to eliminate the $scope as much as possbile. so we wont be injuecting the $scope unless  
 and untill its needed  
  
\*\*\*\* Avoid scope scooping\*\*\*\*  
  
Lets say we have this html:  
 <div ng-controller="HomeControler"> &lt;!&ndash;$scope&ndash;&gt;  
 <div ng-controller="ProjectController">  
 <input type="text" ng-model="ProjectCtrl.user.username"/>  
 <input type="password" ng-model="ProjectCtrl.user.password"/>  
 </div>  
 </div>  
  
  
 Lets assume for a second that both homecontroller now the question arises that from which controller this scenario  
 is called "scope scoop"  
 this confusion will make our application inconsistent and sometimes crash.  
  
  
 to avoid this issue we use mvvm architechtural apporoach. we give aliases to our controllers and use those alias  
 to write code  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\*\*Directives:\*\*  
the entire angular framework sits purely on the concept of directives.  
we can safely say that from html prospective directives are custom attributes.  
  
From angulars prospective these are special types of attributes using which:  
1. we can render an entire html  
2.we can get the values from the controller as well as from the view model  
3. we can alter html as soon as it is loaded/render  
4. we can perform window(dom) related logics  
  
 there mainly three types of directives:  
 1.attribute level directive  
 2.element level directive  
 3.class level directive  
  
  
\*\*ngModel --> ng-model  
  
 example:  
1. in directive.js file  
function KrishnaElement() {  
 var self = this;  
 self.element = {  
  
 //restrict -> what kind of directive it is:  
 /\*E -> element  
 A-> Atribute  
 C -> class\*/  
 restrict: 'E',  
 //prelink (optional)-> perform some logic before the element is rendered (or) the directive kicks in  
  
 prelink: function () {  
  
 },  
 //postlink -> perform the logic after the element is rendered.  
 postlink: function(){  
  
 },  
  
 //scope(optional) -> similar to $scope and $rootScope we can define ouir own scope for the directive  
 scope: {  
  
 },  
 //trasclude -> transculstion means that we are basically making sure that the controller communicates with  
 //our directives it is a boolean value  
 transclude: true/false,  
  
 //template-> this is used to render our html  
 template: '<div class="">  
  
 '</div>',  
 //templateURL --> TO AVOID THE UPPER HTML VALUE SCENARIO WE CAN SOURCE AN ENTIRE HTML FILE  
 templateUrl: '../templates/home.temp.html'  
  
 };  
  
 return self.element;  
 }  
---------------------------------------------------------------------------------------  
2.in html:  
<body>  
  
<krishna-element> </krishna-element>  
</body>

Filters:

Any angularjs filter usually takes input and gives output

Can be used to:

Format data

Transform data

Serach/filter data

Can be used in:

Templates

Controllers

Services

Types of filters:

Built in (comes with angularjs ): number, data, filter, uppercase, lowercase , OrderBy

Custom: developed by us

1.uppercase: {{employee.name | uppercase}}

2.lowercase : {{employee.name | lowercase}}

3.number:2 {{employee.name | number:2}}

4.currency:”$”:2 {{employee.name | currency: “$”:2}}

5.filters: <**tr ng-repeat="*employee* in employees | filter:{name:'s'}"**>

$http service in angualrjs:

$http service is used to make http requests to remote server

$http service is a function that has a single input parameters i.e configuration object

Example: the following example issues a GET request to the specified url

$http({

Method: ‘GET’,

url:’’

});

Shortcut methods like get,post,put and delete are also available to be used with $http service

$http service returns a promise object

Use the $log service to log the response object to the console

Built in services: $http service is use to make ajax calls $log service to use log an object to console

Services: A service in angular is simply an object that provide some sort of service that can be reused with in an angular application

Why do we need services?

Services encapsulate reusable logic that does not belong anywhere else(directives, filters views models and controllers)

What are the benefits of using services?

.reusability-

Dependency injection

Testability

Angularjs anchorscroll:

$anchorscroll service is used to jump to a specified element on the page

$location service hash method appends hash fragments to the URL

$anchorscroll() method reads the has fragment in the URL and jumps to that element on the page

|  |
| --- |
|  |
| Features. |
|  | Angularjs is powerful javascript based development framework to cre4ate Rich Internet Applicaiton(RIA) |
|  |  |
|  | Angularjs Provides developers options to write client side application(using javascript) in a clean side application in a clean MVC(model view controller) |
|  | Application written in angularjs is cross-browser compliant. Angularjs autometically handles js code suitable for each browser. |
|  | Ajs is open source, completely free, and used by thousands of developers around the world. It is licensed under the apache license version 2.0 |
|  |  |
|  | ajs is perfect for single page applications(SPA) |
|  |  |
|  | basics of angularjs: directives, expressions, filters, modules, and controllers, |
|  |  |
|  | ajs extends html with ng-directives. |
|  |  |
|  | the ng-app directive degines an ajs application. |
|  |  |
|  | the ng-model directive dinds the value of html controls(input, select, textarea)to application data. |
|  |  |
|  | the ng-bind directive binds application data to the html view. |
|  |  |
|  | the ng-init directive intializes ajs application variables. |
|  |  |
|  | the ng-repeat directive repeats html elemts for each item in a collection. |
|  |  |
|  | ajs directives are html attributes with an ng prefix. |
|  | --------------------------------------------------------- |
|  | you can use data-ng- , insted of ng- , if you want to make your html page valid. |
|  | ------------------------------------------------------------ |
|  | ==expressions== |
|  |  |
|  | 2+5 |
|  |  |
|  | {{2+5}} --double braces-- |
|  |  |
|  | written inside a directive: ng-bind="expression" |
|  |  |
|  | using expression:: |
|  |  |
|  | <div data-ng-app="" data-ng-init="mago=20;cost=50"> |
|  | <p>total {{mago \* cost}}</p> |
|  |  |
|  |  |
|  | using bind |
|  |  |
|  | <div data-ng-app="" data-ng-init="mago=20;cost=50"> |
|  | <p>total <span ng-bind="mago \* cost"></span></p> |
|  |  |
|  | ------------------------------------------------------------------------ |
|  | strings== |
|  | <div data-ng-app="" data-ng-init="fname='krishna';lname='kanth'"> |
|  | <p>total <span data-ng-bind="fname+lname"></span></p> |
|  |  |
|  | <div data-ng-app="" data-ng-init="pet={fname:'krishna',lname:'kanth'}"> |
|  | <p>total name is : {{pet.fname}}</p> |
|  |  |
|  | ---adding variables is called concordinate-- |
|  |  |
|  | ags objects:: |
|  |  |
|  | <div data-ng-app="" data-ng-init="pet={fname:'krishna',lname:'kanth'}"> |
|  | <p>total name is : {{pet.fname}}</p> |
|  |  |
|  | --------pet = object |
|  | fname = object property |
|  | -------------------------------------------------------------------------- |
|  |  |
|  | Ajs Arrays |
|  |  |
|  | <div data-ng-app="" data-ng-init="names=[1,'krishna',19,'kanth']"> |
|  | <p>total name is : {{names[1]}}</p> |
|  | <p> total name is <span data-ng-bind="names[1]"></span></p> |
|  | --------------------------------------------------- |
|  | ajs expressions vs js expressions |
|  |  |
|  | like javascript expressions ajs expressions can contain literals, operators and variables. |
|  |  |
|  | unlike javascript expressions, ajs expressions can be written inside html. |
|  |  |
|  | ajs expressions do not support conditions, loops and exceptions while javascript do. |
|  |  |
|  | ajs expressions support filters while javascriot expressions do not. |
|  | -------------------------------------------------------------- |
|  | Mvc based frame work: |
|  |  |
|  | A model view controller pattern is made up of the following three parts:- |
|  |  |
|  | Model: it is the lowest level of the pattern resposible for maintaing data. |
|  | the model is resposible for managing application, it responds to the request from view and to the instructions from controller to update itself. |
|  |  |
|  | the model is the part of the application that handles the logic for the application data. often model objects reterieve data from a database. |
|  | view: it is responsible for displaying all or a portion of the data to the user. |
|  | A presentation of data in a particular format, triggered by the controller decision to present tht data. They are script based template systems such as jsp asp php and very |
|  | easy to integrate with ajax technology, |
|  | the view is the parts of the application that handles the display of the data. most often the views are created from teh model data. |
|  | Controller - it is a software code that controls the interactions between the model and view. |
|  | the controller responds to user input and perfroms interaction on the data model objects. the controller receives inpout validates |
|  | it and then performs busiess operations that modify the state of the data model. |
|  | The controller is the part of the application that hadles user interaction typically controller read data from a view, control user input and send input data to the model. |
|  | ----------------------------------------------------------------------------------------- |
|  | browser---(http request)-->controller----(excusion)-->model---(resulting)-->controller----(resulting data arrays)--> view----(gui content)-->controller---(http response)---browser |
|  | ------------------------------------------------------------------------------ |
|  | ng repeat directive in ajs |
|  |  |
|  |  |
|  | two dimentional array. |
|  | <div ng-app="" n-init="myfriends=[{name:'krishna',country:'usa'},{name:'krishna',country:'usa'},] |
|  |  |
|  | <p>looping with objects:</p> |
|  | <ul> |
|  | <li ng-repeat="x in myfriends"> |
|  | {{ x.name+x.country}} |
|  | ---------------------------------------------------------------------------------------- |
|  | create new directives in ajs |
|  |  |
|  | we can create using .directive function |
|  |  |
|  | new directives are created by using the directive function |
|  | when naming a directive you must use a camel case name, but when invoking it, you must use sepatated name, |
|  |  |
|  | <kri-Shna></kri-Shna> |
|  | you can invoke a directive ny using: |
|  |  |
|  | Element name <kri-Shna></kri-Shna> |
|  | attribute <div kri-Shna></div> |
|  | class <div class="kri-Shna"></div> |
|  |  |
|  | Restrictions: |
|  | you can restrict your directives to onlu be invoked by some of the methods. |
|  | the legal restrict values are: |
|  | E for element name |
|  | A for attribute |
|  | C for class |
|  | m for comment |
|  |  |
|  | by default the value is EA meaning that both element names and attribute names can invoke the directive. |
|  | ----------------------------------------------------------------------- |
|  | controllers: |
|  |  |
|  | ajs contllers control the data of ajs applications. |
|  | ajs are regular js objects. |
|  |  |
|  | Ajs application mainly relies on controllers to control the flow of data in the application. |
|  |  |
|  | a controller is defined using ng-controller directive. |
|  |  |
|  | a controller is a javascript obect containing attriutes and functions. |
|  | each controller accepts $scope as a parameter which refers to the application that controller is to control. |
|  |  |

Core Features:

* Data-binding: It is the automatic synchronization of data between model and view components.
* **Scope:** These are objects that refer to the model. They act as a glue between controller and view.
* **Controller:** These are JavaScript functions that are bound to a particular scope.
* **Services:** AngularJS come with several built-in services for example $https: 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.
* **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 (ngBind, ngModel...)
* Routing: It is concept of switching views.
* Model view whatever –

Ng-repeat, ng-model, ng-init, ng-bind example

<html>

<head>

<title>AngularJS Directives</title>

</head>

<body>

<h1>Sample Application</h1>

<div ng-app = "" ng-init = "countries = [{locale:'en-US',name:'United States'}, {locale:'en-GB',name:'United Kingdom'}, {locale:'en-FR',name:'France'}]">

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

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

<p>List of Countries with locale:</p>

<ol>

<li ng-repeat = "country in countries">

{{ 'Country: ' + country.name + ', Locale: ' + country.locale }}

</li>

</ol>

</div>

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

</body>

</html>

Events

AngularJS provides multiple events which can be associated with the HTML controls. For example ng-click is normally associated with button. Following are supported events in Angular JS.

* ng-click
* ng-dbl-click
* ng-mousedown
* ng-mouseup
* ng-mouseenter
* ng-mouseleave
* ng-mousemove
* ng-mouseover
* ng-keydown
* ng-keyup
* ng-keypress
* ng-change

## Validate data

Following can be used to track error.

* **$dirty** − states that value has been changed.
* **$invalid** − states that value entered is invalid.
* **$error** − states the exact error.

Angularjs -Includes

HTML does not support embedding html pages within html page. To achieve this functionality following ways are used −

* **Using Ajax** − Make a server call to get the corresponding html page and set it in innerHTML of html control.
* **Using Server Side Includes** − JSP, PHP and other web side server technologies can include html pages within a dynamic page.

Using AngularJS, we can embed HTML pages within a HTML page using ng-include directive.

<div ng-app = "" ng-controller = "studentController">

<div ng-include = "'main.htm'"></div>

<div ng-include = "'subjects.htm'"></div>

</div>

Angularjs- Ajax

AngularJS provides $https: control which works as a service to read data from the server. The server makes a database call to get the desired records. AngularJS needs data in JSON format. Once the data is ready, $https: can be used to get the data from server in the following manner

function studentController($scope,$https:) {

var url = "data.txt";

$https:.get(url).success( function(response) {

$scope.students = response;

});

}

Here, the file data.txt contains student records. $http: service makes an ajax call and sets response to its property students. Sudents model can be used to draw tables in html.

AngularJS- Scopes:

Scope is a special javascript object which plays the role of joining controller with the views. Scope contains the model data. In controllers model data is accessed via $scope object.

$scope is passed as first argument to controller during its constructor definition.

$scope.message and $scope.type are the models which are to be used in the html page

We have set values to models which will be reflected in the application module whose controller is shapecontroller.

We can define function as well in $scope

Scope inheritance:

Scope are controllers specific. If we defines nested controllers then child controller will inherit the scope of its parent controller

<script>

var mainApp = angular.module("mainApp", []);

mainApp.controller("shapeController", function($scope) {

$scope.message = "In shape controller";

$scope.type = "Shape";

});

mainApp.controller("circleController", function($scope) {

$scope.message = "In circle controller";

});

</script>

Following are the important points to be considered in above example:

1.we have set values to models In shape controller.

2.we have overridden message in child controller circle controller when message is used within module of controller circle controller the overridden message will be used

Angularjs Serivces: Angularjs supports the concepts of separation of concetns using servives architecture.services are javascript functions and are responsible to do a specific tasks only. This makes them an individual entitiy which is maintainable and testable. Controllers filters can call them as on requirement basis services are normally injected using dependency injection mechanism of angularjs

AngularJS provides many inbuilt services for example, $https:, $route, $window, $location etc. Each service is responsible for a specific task for example, $https: is used to make ajax call to get the server data. $route is used to define the routing information and so on. Inbuilt services are always prefixed with $ symbol.

There are two ways to create a service.

factory

service

services are singleton applications

Angularjs Dependency injection:

Dependency injectin is a software design pattern in which components are given their dependencies instead of hard coding them within the component. The relieves a component from locating the dependency and makes dependencies configurable. This helps in making components reusable maintainable and testable.

AngularJS provides a supreme Dependency Injection mechanism. It provides following core components which can be injected into each other as dependencies.

* value
* factory
* service
* provider
* constant