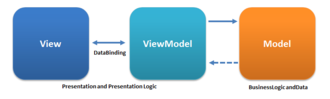
**AngularJS**

1. Open-Source, Web Frontend Framework
2. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–viewmodel (MVVM) architectures, along with components commonly used in rich Internet applications.
3. Model–view–viewmodel (MVVM) is a software architectural pattern.

MVVM facilitates a separation of development of the graphical user interface – be it via a markup language or GUI code – from development of the business logic or back-end logic (the data model).



1. Directives are HTML attributes that start with an ng prefix.
2. The **ng-app** directive defines an AngularJS application.
3. If you write in ng-app do not forget to define it in script.

Define this in script  
<div ng-app="**myApp**">

No need for definition for  
<div ng-app="">

1. The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data.

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

1. The **ng-bind** directive binds application data to the HTML view.

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

1. The **ng-init** directive initializes AngularJS application variables.

<div ng-app="" ng-init="firstName='John'">  
<p>The name is <span ng-bind="firstName"></span></p>  
</div>

<div ng-app="" ng-init="quantity=1;cost=5">  
<p>Total in dollar: {{ quantity \* cost }}</p>  
</div>

1. The **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>

The [] parameter in the module definition can be used to define dependent modules.

Without the [] parameter, you are not creating a new module, but retrieving an existing one.

1. AngularJS Strings

<div ng-app="" ng-init="firstName='John';lastName='Doe'">  
<p>The name is {{ firstName + " " + lastName }}</p>  
</div>

<div ng-app="" ng-init="firstName='John';lastName='Doe'">  
<p>The name is <span ng-bind="firstName + ' ' + lastName"></span></p>  
</div>

1. AngularJS Objects

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

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

1. AngularJS Arrays

<div ng-app="" ng-init="points=[1,15,19,2,40]">  
<p>The third result is {{ points[2] }}</p>  
</div>

<div ng-app="" ng-init="points=[1,15,19,2,40]">  
<p>The third result is <span ng-bind="points[2]"></span></p>  
</div>

1. AngularJS Expressions vs JavaScript Expressions

* Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.
* Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.
* AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.
* AngularJS expressions support filters, while JavaScript expressions do not.

1. Repeating HTML Elements

The **ng-repeat** directive repeats an HTML element:

<div ng-app="" ng-init="names=['Jani','Hege','Kai']">  
  <ul>  
    <li ng-repeat="x in names">  
      {{ x }}  
    </li>  
  </ul>  
</div>

1. S
2. S