Harsha Vardhan

Harsha Vardhan
(Web UI Expert)

Introduction to AngularJS

What is AngularJS?

- AngularJS is a "JavaScript Library" of "JavaScript Framework", which is used to create "data bindings" in the web pages.
- "JavaScript Library" means "collection of pre-defined functions".
- "JavaScript Framework" means "collection of many technologies like HTML + CSS + JavaScript + JSON + AJAX.
- AngularJS is developed by Google.
- AngularJS is very popular & mostly demanded.
- AngularJS is open source, light weight and cross-browser compatible.

Harsha Vardhan
(Web UI Expert)

Problems before AngularJS

Problems before AngularJS

- **No Separation between "HTML" and "JavaScript":** Generally, we can use "Normal JavaScript" or "jQuery" to perform DOM manipulations directly. In this case, we will use ID's (or css class names) in the JavaScript / jQuery code. Thus we depend on the html elements directly in the JavaScript / jQuery code. So both components can't be individually developed. So this makes a maintenance problem.

- **No Modules:** Large projects will be very complex and need to be divided into many modules. In "html, css, javascript", there is no way to divide the large project into modules.

- **No URL Routing:** In "html, css, javascript", there is no direct and easy way to create URL routing. URL routing makes the URL (address at browser's address bar) understandable and reflect the dynamic changes made in the web page. **Ex:** #inbox.

- **No Unit Testing:** By default, JavaScript functions are not unit-testable, as they perform DOM manipulations directly.

Harsha Vardhan
(Web UI Expert)

Features of AngularJS

Features of AngularJS

- **Clean separation between HTML and JavaScript code:** In AngularJS, html and javascript code can be developed individually by two different programmers / teams. This is possible by using the concept of "data bindings". In data bindings, we will create relationship between an html tag and a variable; and then every time when the html tag's value has been changed, angularjs will change the value of the variable automatically and vice versa.

- **Modules:** AngularJS supports modules. Modules are used to divide the large projects into parts; each part is called as "module". So that the project will be easy-understandable.

- **URL Routing:** In AngularJS, we can use "AngularJS Routing" concept to implement "URL Routing". URL routing is used to make the URL's represent the current status of the web page. **Ex:** #inbox. URL Routing is very important for modern applications.

- **Declarative Code:** HTML is meant for static views. AngularJS extends HTML language by adding "declarative code" to html. AngularJS is lets you to describe dynamic views in declarative way in a readable, expressive manner.

- **Unit Testing:** AngularJS supports unit testing. That means many AngularJS components, such as controllers, directives, services, factories etc., are unit-testable. That means each small component of angularjs can be individually run, by sending a dummy request, and then we can make sure the component returns a correct value.

Harsha Vardhan
(Web UI Expert)

Types of web applications

Types of web applications

- Web applications are two types.
    1. Multi Page Applications
    2. Single Page Applications

1. Multi Page Application:

- Whenever a web application has multiple web pages, and the previous web page gets closed before another page gets opened, we call it as "multiple page application".
- **Ex:** www.facebook.com
- **Drawback:** It takes much time to load full page.

2. Single Page Application:

- Whenever a web application has only one web page & subpages are displayed in the same web page, we call it as "single page application".
- **Ex:** www.gmail.com
- **Advantage:** It takes less time to load subpage.
- Single page application requires "clean separation between html and javascript code"; Angularjs provides the same.
- Angularjs is mainly meant for single applications; however we can use angularjs for multipage applications also, to get its benefits.

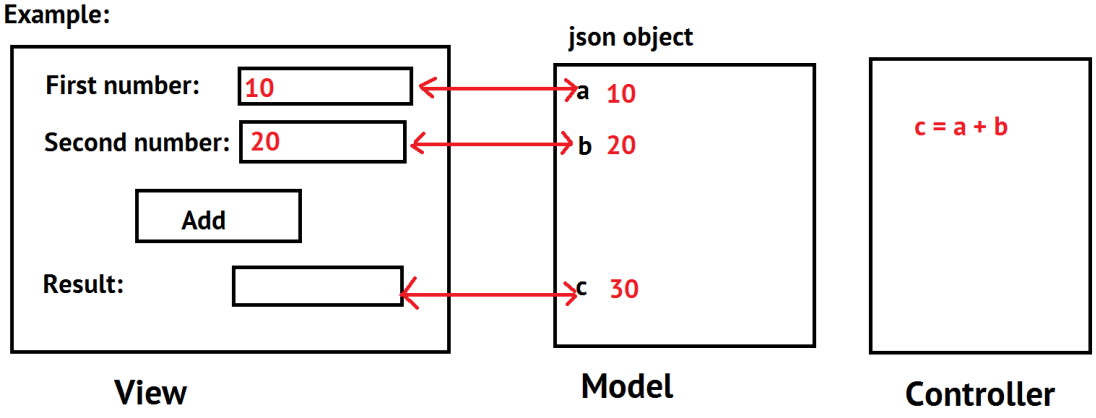
Harsha Vardhan
(Web UI Expert)

AngularJS Data bindings

Data bindings

- Data bindings are used to maintain relationship between an "html tag" and a "variable"; so that its values will be mutually updated. That means when the html tag's value has been changed automatically angularjs will change the value of the variable; and vice versa.

Example:



MVC Architecture

MVC Pattern in AngularJS

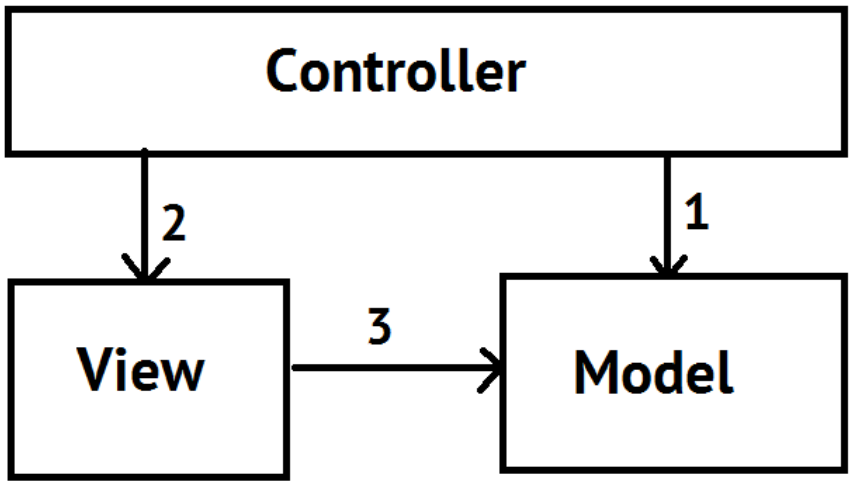
- "MVC" is an "architecture" and "design pattern".
    - Architecture is pictural representation of the components of the project; design pattern is a set of guidelines to write the programs.
- MVC dictates you how to split your code into multiple parts.
- The project code is divided as 3 major parts:
    1. Model
    2. View
    3. Controller
- **Model:** It is a JSON object, which stores data of the application.
- **View:** It is a set of html tags to display model data. View can access model. So "data bindings" are maintained between "model" and "view". When some changes happen in the "view", then automatically AngularJS changes the same in the "model". When some changes happen in the "model", then automatically AngularJS changes the same in the "view".
- **Controller:** It is a JavaScript function, which manipulates the model & view. Controller contains code for functionality.

3 basic principles of MVC:

1. Controller calls model.
2. Controller calls view.
3. View calls model.

Harsha Vardhan
(Web UI Expert)

MVC Architecture



Harsha Vardhan
(Web UI Expert)

Most important concepts of AngularJS

Most Important Concepts of AngularJS

- In AngularJS, the following concepts are very important:

1. AngularJS - Introduction
2. AngularJS - Getting started
3. AngularJS - First Example
4. AngularJS - Views
5. AngularJS - Directives
6. AngularJS - Models
7. AngularJS - Controllers
8. AngularJS - Modules
9. AngularJS - Scopes
10. AngularJS - Dependency Injection
11. AngularJS – Bootstrapping
12. AngularJS – Data Bindings
13. AngularJS - $watch
14. AngularJS – Filters
15. AngularJS – Ng-repeat
16. AngularJS - AJAX
17. AngularJS – Animations
18. AngularJS – Validations
19. AngularJS - $q
20. AngularJS - Routing
21. AngularJS - Unit Testing

AngularJS Directives

What are AngularJS Directives

- AngularJS directives are "attributes" that are provided by angularjs.
- AngularJS directives are used to add additional functionality to an html tag.
- AngularJS supports "pre-defined directives" and "user-defined directives".
- **List of important pre-defined AngularJS Directives:**

Sl. No	Description
1	**ng-app** It is used to create angularjs application (root view).
2	**ng-init="variable = value"** It is used to initialize variables of current model.
3	**ng-model="property"** It is used to bind (connect) a model property to an <input> tag (or) <select> tag.
4	**ng-bind="property"** It is used to bind (connect) a model property to any other html tag (except <input> and <select) tags).
5	**ng-controller="controller name"** It is used to bind (connect) an "angularjs application" with an "angularjs controller".

6	**ng-strict-di** It is used to make "dependency injection" as strict. If the developer won't do dependency injection, then the angularjs application will not run.
7	**ng-repeat="variable in array"** It is used to read all the elements from an array in a sequence (just like foreach loop) and repeat the current html tag for each array element once.
8	**ng-click="function name"** It is used to handle "click" event.
9	**ng-dbl-click ="function name"** It is used to handle "dbl-click" event.
10	**ng-mouseover="function name"** It is used to handle "mouseover" event.
11	**ng-mouseout="function name"** It is used to handle "mouseout" event.
12	**ng-keypress="function name"** It is used to handle "keypress" event.
13	**ng-keyup="function name"** It is used to handle "keyup" event.
14	**ng-change="function name"** It is used to handle "change" event.

15	**ng-disabled="true | false"** **true:** The element will be disabled. **false:** The element will be enabled.
16	**ng-show="true | false"** **true:** The element will be visible. **false:** The element will be invisible.
17	**ng-hide="true | false"** **true:** The element will be invisible. **false:** The element will be visible.
18	**ng-class="[class1, class2, ...]"** It is used to apply css classes dynamically to an html element.
19	**ng-style="{ property: value, property: value, ... }"** It is used to set css styles dynamically to an html element

Getting started with AngularJS

Getting started with AngularJS

- Download "angular.js" file from the website.
- Copy and paste the "angular.js" file into the project folder.
- Import the "angular.js" file into the web page.
- Create angularjs application.

How to download "angular.js" file:

- Go to https://angularjs.org/ and click on "Download AngularJS 1".
- Select the latest stable version. Ex: 1.5.x
- Click on "Uncompressed".
    - **Minified:** Minified and obfuscated version of the AngularJS base code. Use this in your deployed application.
    - **Uncompressed:** The main AngularJS source code, as is. Useful for debugging and development purpose, but should ideally not be used in your deployed application.
    - **Zipped:** The zipped version of the Angular Build, which contains both the builds of AngularJS, as well as documentation and other extras.
- Click on "Download".
- You will get a javascript file. Go to the browser menu and click on "Save page". The file name is "angular.js"; select "Desktop" and click on "Save". Thus you got the "angular.js" file into your desktop. Copy and paste it from "Desktop" to "your current working application folder" Ex: C:\angularjs.

Harsha Vardhan
(Web UI Expert)

How to import “angular.js” file into your web page

- Create an html file (Ex: first.html) and save it in your current working application folder. (Ex: C:\angularjs).
- You must import “angular.js” file in the web page, before writing angularjs code.
- **Syntax:**

```
<script type="text/javascript" src="angular.js">
</script>
```

- This will download the “angular.js” file into the browser; based on which, all the angularjs code runs on the browser.

Harsha Vardhan
(Web UI Expert)

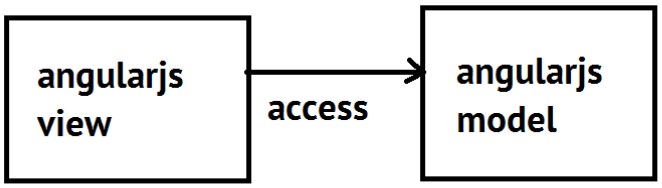
AngularJS Application

"AngularJS application" or "AngularJS View"

- "ng-app" is pre-defined directive, which is used to create an angularjs application (angularjs root view).
- You can use all angularjs concepts such as directives, filters etc., only in the angularjs application.
- Syntax to create angularjs application (or) angularjs view:

```
<div ng-app>
        Your html / angularjs code here
</div>
```

- When we create an angularjs application, angularjs will create an "angularjs model" automatically.



{

}

JSON object:

```
{
    "property": "value", "property": "value", ...
}
```

Example of JSON object:

```
{
    "carno": "1234", "carmodel": "vento", "carcolor": "white"
}
```

Harsha Vardhan
(Web UI Expert)

AngularJS Expressions

AngularJS Expressions

- AngularJS expressions are used to display a value of a variable in the view.
- **Syntax to create an expression:**

```
{{ expression }}
```

Harsha Vardhan
(Web UI Expert)

AngularJS – First Example

first.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>AngularJS - First Example</title>
		<style type="text/css">
			body, input
			{
				font-family: 'Tahoma';
				font-size: 30px;
			}
		</style>

		<!-- import angular-js script file -->
		<script type="text/javascript" src="angular.js">
		</script>

	</head>
	<body>
		<!-- creating an angularjs application -->
		<div ng-app>
			{{10+20}}
		</div>
		<!-- end of angularjs application -->
	</body>
</html>
```

AngularJS – ng-init - Example

nginit.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>AngularJS - ng-init</title>
		<style type="text/css">
			body, input
			{
				font-family: 'Tahoma';
				font-size: 30px;
			}
		</style>

		<!-- import angular-js script file -->
		<script type="text/javascript" src="angular.js">
		</script>
	</head>
	<body>
		<!-- creating an angularjs application -->
		<div ng-app ng-init="x=10; y=20; z=x+y">
		  <p>{{x}}</p>
		  <p>{{y}}</p>
		  <p>{{z}}</p>
		</div>
		<!-- end of angularjs application -->
	</body>
</html>
```

Harsha Vardhan
(Web UI Expert)

AngularJS – ng-model - Example

ngmodel.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>AngularJS - ng-model</title>
		<style type="text/css">
			body, input
			{
				font-family: 'Tahoma';
				font-size: 30px;
			}
		</style>

		<!-- import angular-js script file -->
		<script type="text/javascript" src="angular.js">
		</script>
	</head>
	<body>
		<!-- creating an angularjs application -->
		<div ng-app ng-init="empid=101; empname='scott'; salary=4500">
			Emp ID: <input type="text" ng-model="empid"><br>
			Emp Name: <input type="text" ng-model="empname"><br>
			Salary: <input type="text" ng-model="salary"><br>
		</div>
		<!-- end of angularjs application -->
	</body>
</html>
```

Harsha Vardhan
(Web UI Expert)

AngularJS – ng-bind - Example

ngbind.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>AngularJS - ng-bind</title>
		<style type="text/css">
			body, input
			{
				font-family: 'Tahoma';
				font-size: 30px;
			}
		</style>

		<!-- import angular-js script file -->
		<script type="text/javascript" src="angular.js">
		</script>
	</head>
	<body>
		<!-- creating an angularjs application -->
		<div ng-app ng-init="empid=101; empname='scott'; salary=4500">
			Emp ID: <span ng-bind="empid"></span><br>
			Emp Name: <span ng-bind="empname"></span><br>
			Salary: <span ng-bind="salary"></span><br>
		</div>
		<!-- end of angularjs application -->
	</body>
</html>
```

Harsha Vardhan
(Web UI Expert)

AngularJS – JSON - Example

json.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>AngularJS - JSON</title>
		<style type="text/css">
			body, input
			{
				font-family: 'Tahoma';
				font-size: 30px;
			}
		</style>

		<!-- import angular-js script file -->
		<script type="text/javascript" src="angular.js">
		</script>

	</head>
	<body>

		<!-- creating an angularjs application -->
		<div ng-app ng-init="employee = { empid: 101, empname:'scott',
salary:5000 }; student = { stuid: 201, stuname: 'allen', marks: 80 }">

			<p>Emp ID: {{employee.empid}}</p>
			<p>Emp Name: {{employee.empname}}</p>
			<p>Salary: {{employee.salary}}</p>
			<hr>
```

```
                <p>Student ID: {{student.stuid}}</p>
                <p>Student Name: {{student.stuname}}</p>
                <p>Marks: {{student.marks}}</p>

            </div>
            <!-- end of angularjs application -->

        </body>
</html>
```

Harsha Vardhan
(Web UI Expert)

AngularJS Modules

AngularJS Modules

- AngularJS project can be divided into parts; each part is a "module".
- AngularJS module is a part of the project.
- **Example:** "Bank" project contains various modules like "UserAccounts module", "SavingsBank module", "Loans module" etc.
- AngularJS module is a collection of views, controllers, models and other components such as services, factories, directives, values, providers, filters etc.
- **Syntax to create a module:**

```
var app = angular.module("module name", [ dependencies ] );
```

- In the above statement, the variable "app" stores the reference of the module.

Harsha Vardhan
(Web UI Expert)

AngularJS Controllers

AngularJS Controllers

- AngularJS controllers contain code for manipulating model and view.
- AngularJS loads data into model & manipulates model; so that the view will display model data automatically.

Syntax to create controller

```
app.run(
        function($rootScope) {
                any code here
        }
);
```

Note: "$rootScope" represents model.

Harsha Vardhan
(Web UI Expert)

AngularJS – Modules - Example

modules.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>AngularJS - Modules</title>
		<style type="text/css">
			body, input
			{
				font-family: 'Tahoma';
				font-size: 30px;
			}
		</style>

		<!-- import angular-js script file -->
		<script type="text/javascript" src="angular.js">
		</script>

		<script type="text/javascript">

			//creating a module called "mymodule"
			var app = angular.module("mymodule", [ ] );

			//creating "run" function
			app.run(function($rootScope)
			{
```

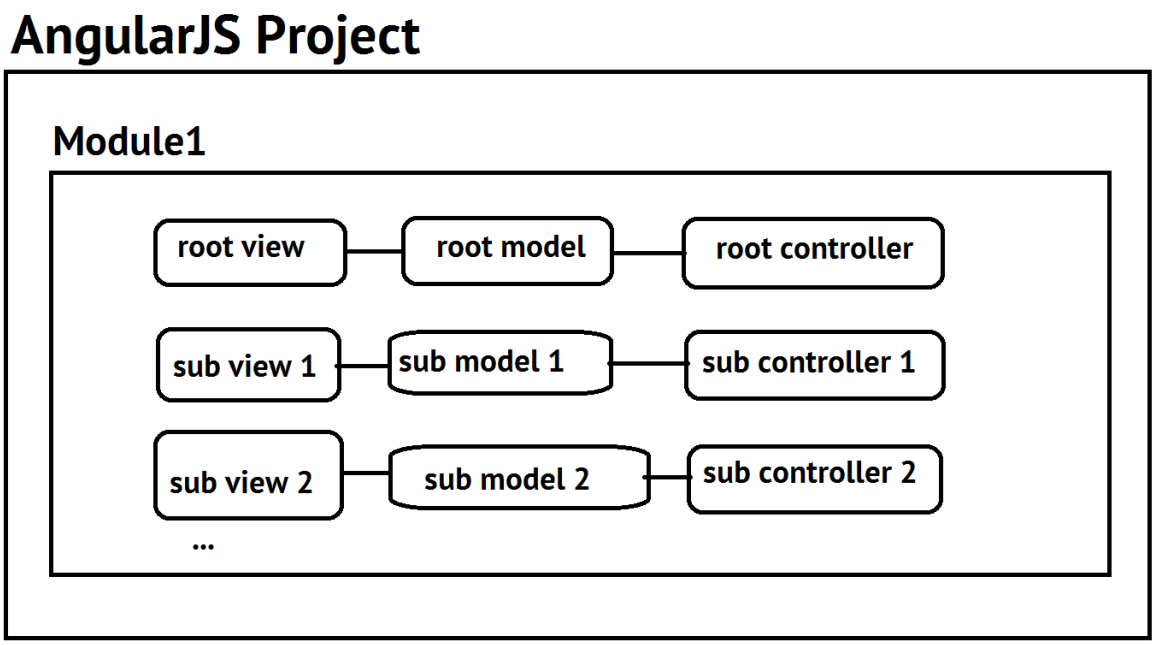
```
                //$rootScope means "model".
                $rootScope.empid = 101;
                $rootScope.empname = "Scott";
                $rootScope.salary = 4000;
            });

        </script>

    </head>
    <body>

        <!-- creating an angularjs application -->
        <div ng-app="mymodule">

            Emp ID: {{empid}}<br>
            Emp Name: {{empname}}<br>
            Salary: {{salary}}<br>

        </div>
        <!-- end of angularjs application -->

    </body>
</html>
```

Harsha Vardhan
(Web UI Expert)

Sub controllers

Sub controllers

- An angularjs project is a collection of modules.
- An angularjs module is a collection of "one root view + one root model + one root controller" and "any no. of sub views + any no. of sub models + any no. of sub controllers".

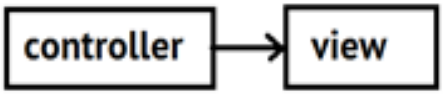


Harsha Vardhan
(Web UI Expert)

Creating "Sub Model", "Sub View" and "Sub Controller:

1. Creating sub controller:

app.controller("controller name", function( ) { *code here* });



Note: Angularjs executes the controller first; and then view.

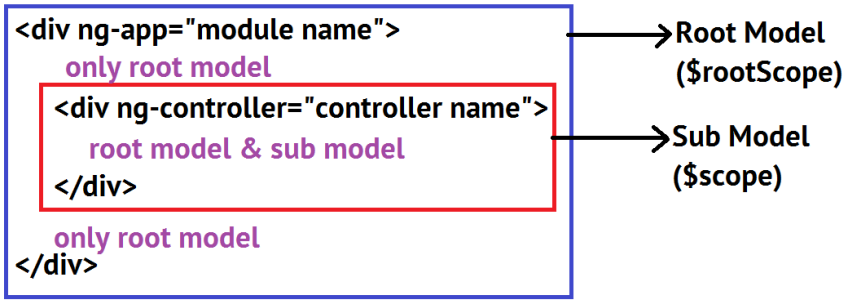
2. Creating sub view:

```
<div ng-controller="controller name">
    Any code here
</div>
```

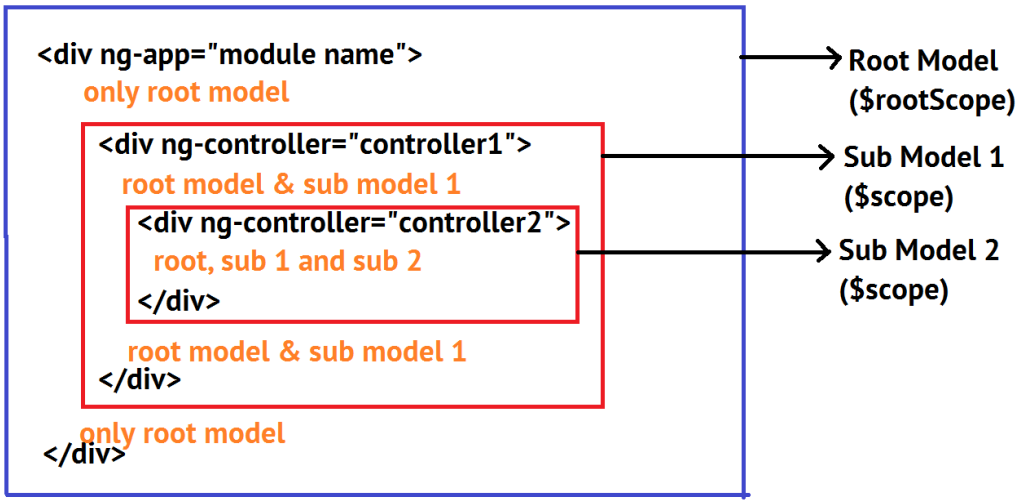
Harsha Vardhan
(Web UI Expert)

Nested Views:

1. "Sub view" inside "Root View":



2. "Sub Model 2" in another "Sub Model 1":



Sub controllers - Example

subcontrollers.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>AngularJS - Controllers</title>
		<style type="text/css">
			body, input
			{
				font-family: 'Tahoma';
				font-size: 30px;
			}
		</style>

		<!-- import angular-js script file -->
		<script type="text/javascript" src="angular.js">
		</script>

		<script type="text/javascript">

			//creating a module called "mymodule"
			var app = angular.module("mymodule", [ ] );

			//creating a controller called "mycontroller", in a module
called "mymodule"
			app.controller("mycontroller", function($scope)
```

```
                {
                        //$scope means "model".
                        $scope.empid = 101;
                        $scope.empname = "Scott";
                        $scope.salary = 4000;
                });

        </script>

    </head>
    <body>

        <!-- creating an angularjs application -->
        <div ng-app="mymodule" ng-controller="mycontroller">

            Emp ID: {{empid}}<br>
            Emp Name: {{empname}}<br>
            Salary: {{salary}}<br>

        </div>
        <!-- end of angularjs application -->

    </body>
</html>
```

$rootScope and $scope - Example

scope.html

```
<!DOCTYPE html>
<html>
	<head>
		<title>AngularJS - Scope</title>
		<style type="text/css">
			body, input
			{
				font-family: 'Tahoma';
				font-size: 30px;
			}
		</style>

		<!-- import angular-js script file -->
		<script type="text/javascript" src="angular.js">
		</script>

		<script type="text/javascript">

			//creating a module called "mymodule"
			var app = angular.module("mymodule", [ ] );

			//creating a controller called "mycontroller1", in a module
called "mymodule"
```

```
                app.controller("mycontroller1", function($scope,
$rootScope)
                {
                        //$scope means "scope of mycontroller1".
                        $scope.empid = 101;
                        $scope.empname = "scott";
                        $scope.salary = 4000;

                        //$rootScope means "scope of entire ng-app"
                        $rootScope.message = "Hello";
                });

                //creating a controller called "mycontroller2", in a module
called "mymodule"
                app.controller("mycontroller2", function($scope)
                {
                        //$scope means "scope of mycontroller2".
                        $scope.productid = 201;
                        $scope.productname = "mobile";
                        $scope.price = 45000;
                });
        </script>
    </head>
    <body>
        <!-- creating an angularjs application -->
        <div ng-app="mymodule">
            <!-- scope of mycontroller1 starts -->
```

```
            <div ng-controller="mycontroller1">
                Emp ID: {{empid}}<br>
                Emp Name: {{empname}}<br>
                Salary: {{salary}}<br>
                <hr>
            </div>
            <!-- scope of mycontroller1 ends -->

            <!-- scope of mycontroller2 starts -->
            <div ng-controller="mycontroller2">
                Product ID: {{productid}}<br>
                Product Name: {{productname}}<br>
                Price: {{price}}<br>
                <hr>
                {{message}}
            </div>
            <!-- scope of mycontroller2 ends -->
            <hr>
            {{message}}
        </div>
        <!-- end of angularjs application -->
    </body>
</html>
```

Harsha Vardhan
(Web UI Expert)

AngularJS – Dependency Injection

Dependency

- **Dependency means:** Based on "base component", the "destination component" works.

Dependency in programming

```
function fun1()
{
}
```

```
function fun2()
{
    fun1();
}
```

- "fun2" depends on "fun1".
- That means system has to load "fun1" into memory, then it can execute "fun2".

Dependency in angularjs programming

- A module consists of "models, views, controllers, filters, values, factories, services, providers and directives".
- We can call any component in any other controller. If you do so, the system has to load "base component" first; and then it can execute the "destination component".

Dependency injection

- Specifying the dependency of one component to another component in a systematic way programmatically is called as "dependency injection".

Dependency Injection in AngularJS

- AngularJS supports 3 types of dependency injection
    1. Automatic dependency injection
    2. Inline array dependency injection
    3. $inject dependency injection

1) Automatic dependency injection

- This is enabled by default in angularjs.
- Whenever you write name of the "base component" as argument in the destination component function, then angularjs automatically loads the base component first; and then it executes the destination component. This is called as "automatic dependency injection".

```
function(basecomponent1, basecomponent2, ...)
{
        Your code here
}
```

2) Inline array dependency injection

- This is mostly recommended in realtime.
- You have to specify the base component names as elements in the string array and the component function should be the last element in that array. Then angularjs automatically loads the base components and assigns them into respective variables.
- **Advantage:** The code works even after minifying the javascript code.

```
[ “basecomponent1”, “basecomponent2”, ...,
function(variable1, variable2, ...)
{
        Your code here
} ]
```

Harsha Vardhan
(Web UI Expert)

2) Inline array dependency injection

- This is mostly recommended in realtime.
- You have to specify the base component names as elements in the string array and assigned to $inject.
- **Drawback:** The component function must be a named function; can't be anonymous function.

```
function(variable1, variable2, ...)
{
        Your code here
}
functionname.$inject = [ "basecomponent1",  basecomponent2"];
```