ANGULAR JS INTERVIEW QUESTIONS FOR BEGINNERS TO EXPERIENCED

1) What is AngularJS?

AngularJS is a framework to build large scale and high performance web application while keeping them as easy-to-maintain. Following are the features of AngularJS framework.

- AngularJS is a powerful JavaScript based development framework to create RICH Internet Application RIA.
- AngularJS provides developers options to write client side application usingJavaScript in a clean MVC ModelViewController way.
- Application written in AngularJS is cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
- AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache License version 2.0.

2) What are the features of AngularJS?

- **MVC :** In AngularJS, you just have to split your application code into MVC

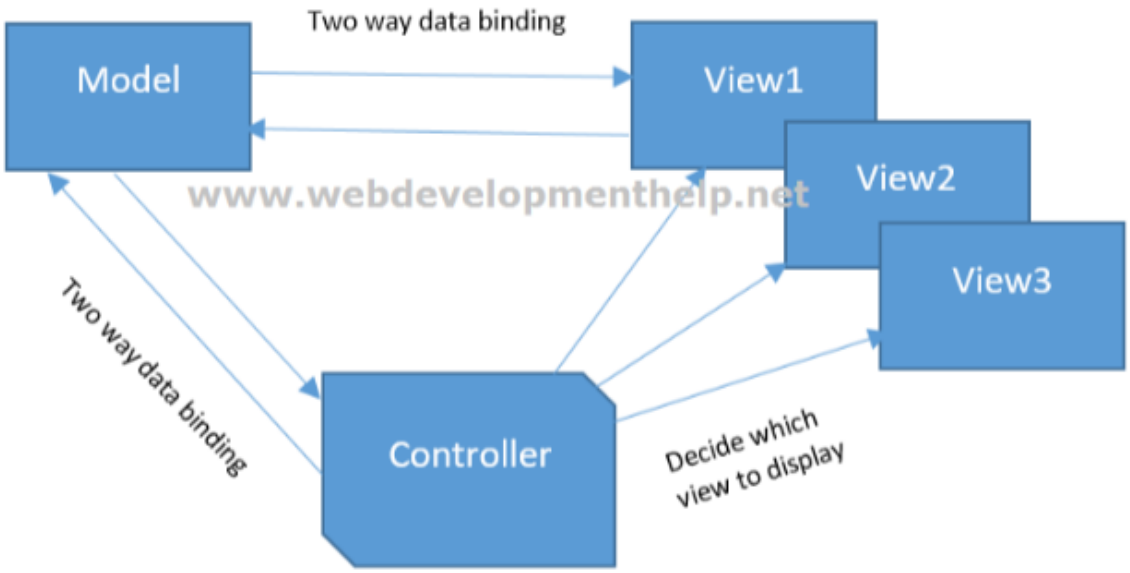
components, i.e., Model, View and the Controller.

- **Validations :** It performs client-side form validation.
- **Modules :** It defines an application.
- **Directives :** It specifies behavior on the DOM element.
- **Templates :** It renders the dynamic view.
- **Scope :** It joins the controller with the views.
- **Expressions :** It binds application data to HTML
- **Data Binding :** It creates a two-way data-binding between the select element and the orderProp model.
- **Filters :** It provides the filter to format data.
- **Services :** It stores and shares data across the application.
- **Routing :** It is used to build a single page application.
- **Dependency Injection :** It specifies a design pattern in which components are given their dependencies instead of hardcoding them within the component.
- **Testing :** It is easy to test any of the AngularJS components through unit testing and end-to-end testing.

3) How MVC is represented in AngularJS?

Model **V**iew **C**ontroller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

- **Model :** It is the lowest level of the pattern responsible for maintaining data.
- **View :** It is responsible for displaying all or a portion of the data to the user.
- **Controller :** It is a software Code that controls the interactions between the Model and View.



- **Example:** Here we create a view with a customerInfoModel model and a CustomerInfoCtrl controller:

```
<html ng-app="angularMvc">
<head>
    <title>Customar information</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>
    <script>
        //Here we define customer model
        var customerInfoModel = {
            name: "Sonia Yeasmin",
            address: "Dhaka, Bangladesh",
        };


        // Here we define main app module
        var appModule = angular.module("angularMvc", []);

                // Here we define controller and $scope object as function argument
        appModule.controller("CustomerInfoCtrl", function ($scope) {
            $scope.customerInfo = customerInfoModel;
        });

    </script>
```

```
</head> <!--Here we bind model data to the view through controller-->
<body ng-controller="CustomerInfoCtrl">
    Customer Name: <input type = "text" ng-model = "customerInfo.name">
    Address : <input type="text" ng-model = "customerInfo.address">
    <br>
    <div>
        <p>Two way binding of Customer Name and Address</p><br>
        <P>Customer Name: {{customerInfo.name}}</P>
        <P>Address: {{customerInfo.address}}</P>
    </div>
</body></html>
```

- Above example we can see how customer **model** is blinded to **view** through **controller**. Through **$scope** object of AngularJS we implement two-way binding and here controller define visibility level of **$scope** object in the view, in this case the **<body>** element of the view.

4) What are the advantages of AngularJS?

- AngularJS provides capability to create Single Page Application in a very clean and maintainable way.
- AngularJS provides data binding capability to HTML thus giving user a rich and responsive experience.
- AngularJS code is unit testable.
- AngularJS uses dependency injection and make use of separation of concerns.
- AngularJS provides reusable components.
- With AngularJS, developer writes less code and gets more functionality.
- In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.
- AngularJS applications can run on all major browsers and smart phones including Android and iOS based phones/tablets.

5) What are the disadvantages of AngularJS?

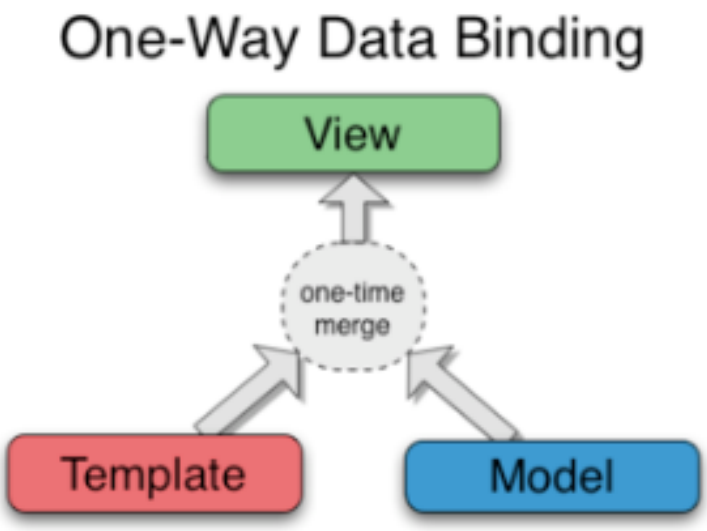
- **Not Secure** – Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.

- **Not degradable** – If your application user disables JavaScript then user will just see the basic page and nothing more.

6) What is data binding in AngularJS?

Data binding is the automatic synchronization of data between model and view components. ng-model directive is used in data binding.

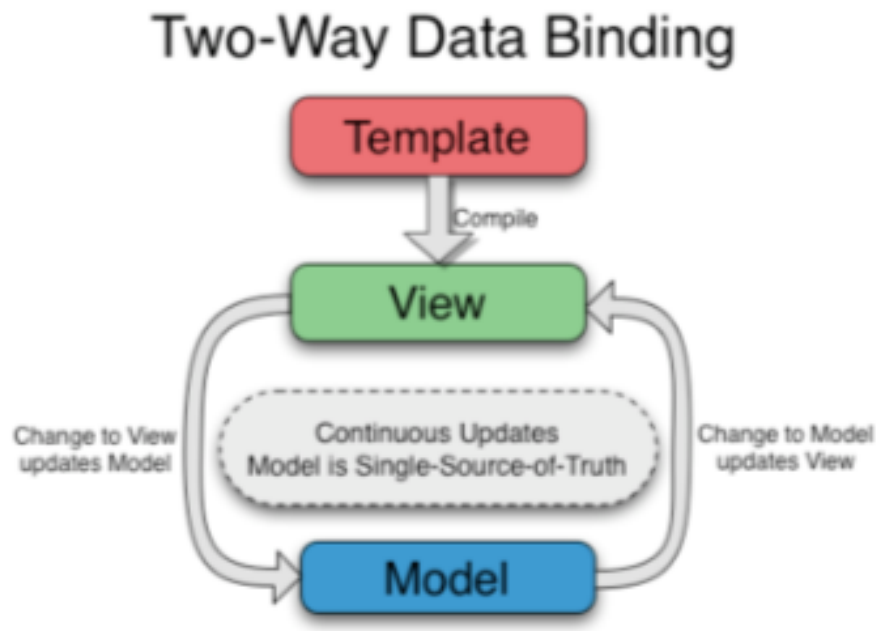
- **One-Way Data Binding :** The one-way data binding is an approach where a value is taken from the data model and inserted into an HTML element. There is no way to update model from view. It is used in classical template systems. These systems bind data in only one direction:



- **Example:**

```
<body>
<div ng-app="" ng-init="firstName='Ajeet'">
<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="firstName"></p>
<p>You wrote: {{ firstName }}</p>
</div>
</body>
```

- **Two-Way Data Binding :** Data-binding in Angular apps is the automatic synchronization of data between the model and view components. Data binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. If the model is changed, the view reflects the change and vice versa.



- **Example:**

```
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script
    <script>
        var mainModule = angular.module("appModule",[]);
        mainModule.controller("CtrlDataBinding", function ($scope) {
            $scope.customerInfo = {};
            $scope.customerInfo.name = "Raid Islam";
            $scope.customerInfo.address = "Dhaka, Bangladesh"
        })
    </script>
</head>

    <body ng-controller="CtrlDataBinding">
        <h1>Customer Information</h1>
        <!--One-way data binding-->
        <strong>Name:</strong> {{customerInfo.name}}<br />
        <strong>Address:</strong> <span ng-bind="customerInfo.address"></span><br />

        <br />
        <!--Two-way data binding -->
        <label>We can set customer name: <input type="text" ng-model="customerInfo.name"/></lab
        <label>We can set customer Address: <input type="text" ng-model="customerInfo.address"/
    </body>
</html>
```

7) What Is The Difference Between One-Way Binding And Two-Way Binding?

The main difference between one-way binding and two-way binding is as follows.

- In one-way binding, the scope variable in the HTML gets initialized with the first value its model specifies.
- In two-way binding, the scope variable will change its value whenever the model gets a different value.

8) What is scope in AngularJS?

Scope is a built-in service (or JavaScript object ***$scope***) of AngularJS framework. Through **$scope**, controller bind model data to their views. In AngularJS scope define the relationship between controllers and views. There are two ways to use ***$scope*** within controller.

- We can define data(model object).
- We can define behaviors which are JavaScript functions.

For example here we will create two ***$scope*** object with initial data and function and also

show there visibility label in DOM hierarchy through two different controllers.

```
<head>
    <title>Scope Example</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>
    <script>
        var appModule = angular.module("scopeExample", []);

        appModule.controller("CtrlFirst", function ($scope) {
            $scope.data = "initial date : 2016-09-10";
            $scope.getGMTDate = function () {
                $scope.data = new Date();
            }
        });

        appModule.controller("CtrlSec", function ($scope) {
            $scope.data = "John Abraham";
            $scope.inverseCase = function () {
                $scope.data = $scope.data.toUpperCase();
            };
        });
    </script>

</head>
```

```
<body>
<div id="first" ng-controller="CtrlFirst">
    <h4>First Controller</h4>
        <button type="button" ng-click=" getGMTDate()">
                Get GMT Date
        </button>
    <input ng-model="data">
</div>

<div id="second" ng-controller="CtrlSec">
    <h4>Second Controller</h4>
        <button type="button" ng-click="inverseCase()">
                Upper Case
        </button>
    <input ng-model="data">
</div>

</body>
```

- To use **$scope** object in controller first we pass the ***$scope*** object to controller's factory function. As seen here we define two controller **CtrlFirst** and **CtrlSec**. **CtrlFirst** controller's ***$scope*** has a data property with initial value **"initial date : 2016-09-10"** and a behavior define by getGMTDate**()**function.

- Similarly ***CtrlSec*** contoller's ***$scope*** has a data property with initial value **"John Abraham"** and **inverseCase()** function.

9) What is Scope Inheritance?

Scope is controller-specific. If we define nested controllers, then the child controller inherits the scope of its parent controller.

```
<body>
   <h2>AngularJS Sample Application</h2>

   <div ng-app = "mainApp" ng-controller = "shapeController">
      <p>{{message}} <br/> {{type}} </p>

      <div ng-controller = "circleController">
         <p>{{message}} <br/> {{type}} </p>
      </div>

      <div ng-controller = "squareController">
         <p>{{message}} <br/> {{type}} </p>
      </div>

   </div>
   <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/
   </script>
```

```
<script>
   var mainApp = angular.module("mainApp", []);

   mainApp.controller("shapeController", function($scope) {
      $scope.message = "In shape controller";
      $scope.type = "Shape";
   });
   mainApp.controller("circleController", function($scope) {
      $scope.message = "In circle controller";
   });
   mainApp.controller("squareController", function($scope) {
      $scope.message = "In square controller";
      $scope.type = "Square";
   });

</script>

/body>
```

The following important points are considered in above example:

- We assign values to the models in shapeController.

- We override message in child controller named *circleController*. When message is used within the module of controller named *circleController*, the overridden message is used.

10) What is scope hierarchy in AngularJS?

Scopes are controllers specific. If we define 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.

- We've set values to models in shapeController.

- We've overridden message in child controller circleController. When "message" is used within module of controller circleController, the overridden message will be used.

11) What are the controllers in AngularJS?

In AngularJS Controller is special function that created from factory function provided by AngularJS framework. The main function of a controller is to pass data and behavior of ***$scope*** object between view and *model* (which is JavaScript object that is blinded to ***$scope*** object). Through controller we also define the visibility of ***$scope*** object in view. It is simply act a glue between the view and the ***$scope's*** *model*.

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>
<script>

    // Here we define the main application module
    var mainApp = angular.module("mainAppModule", []);

        // create ExampleController controller from factory method(controller) of AngularJS
        mainApp.controller("ExampleController",function ($scope) {
        //Here we define the customer info model
        $scope.customerInfo={};

        //define property to customerInfo model and initialized their value
        $scope.customerInfo.name="Rocky";
        $scope.customerInfo.address="Dhaka Bangladesh";
        $scope.customerInfo.monthlyExpense=0;


            //Here we define a behavior of customer info model
            $scope.customerInfo.getCustomerCategory = function (expense) {
                if(expense>5000){
                    //assign expense value to monthlyExpense property
                    $scope.customerInfo.monthlyExpense=expense;
                    alert("VIP Customer");
                }
                else{
                    //assign expense value to monthlyExpense property
                    $scope.customerInfo.monthlyExpense=expense;
                    alert("Normal Customer")
                }
            }
      });

  </script>
```

```
</head>
<body ng-controller="ExampleController">
<p>Customer Name: </p> <input  type="text" ng-model="customerInfo.name">
<p>Customer Address: </p> <input  type="text" ng-model="customerInfo.address">
<br><br>
Select Monthly expense:
    <select ng-model="customerInfo.monthlyExpense"
            ng-change="customerInfo.getCustomerCategory(customerInfo.monthlyExpense)">
        <option value="0">--Select--</option>
        <option ng-selected="customerInfo.monthlyExpense == '5000' "value="5000">VIP</option>
        <option ng-selected="customerInfo.monthlyExpense == '4000' "value="4000">NORMAL</option>
    </select>
<br>
<p>Monthly customer expense</p>
<span ng-bind="customerInfo.monthlyExpense"></span>
</body>
```

12) What are the services in AngularJS?

AngularJS come with several built-in services. For example $https: service is used to make XMLHttpRequests (Ajax calls). Services are singleton objects which are instantiated only once in app.

13) What Is Factory Method?

Using factory method, we first define a factory and then assign method to it:

```
var mainApp = angular.module("mainApp", []);
mainApp.factory('MathService', function() {
   var factory = {};

   factory.multiply = function(a, b) {
      return a * b
   }

   return factory;
});
```

14) What Is Service Method?

Using service method, we define a service and then assign method to it. We've also injected an already available service to it:

```
mainApp.service('CalcService', function(MathService) {
   this.square = function(a) {
      return MathService.multiply(a,a);
   }
});
```

15) What Is A Service?

Services are JavaScript functions and are responsible to do specific tasks only. Each service is responsible for a specific task for example, $http 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:

```
<body>
   <h2>AngularJS Sample Application</h2>

   <div ng-app = "mainApp" ng-controller = "CalcController">
      <p>Enter a number: <input type = "number" ng-model = "number" /></p>
      <button ng-click = "square()">X<sup>2</sup></button>
      <p>Result: {{result}}</p>
   </div>

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

      mainApp.factory('MathService', function() {
         var factory = {};

         factory.multiply = function(a, b) {
            return a * b
         }
         return factory;
      });
      mainApp.service('CalcService', function(MathService) {
         this.square = function(a) {
            return MathService.multiply(a,a);
         }
      });
      mainApp.controller('CalcController', function($scope, CalcService) {
         $scope.square = function() {
            $scope.result = CalcService.square($scope.number);
         }
      });
   </script>

</body>
```

16) What Are The Differences Between Service And Factory Methods?

Factory method is used to define a factory which can later be used to create services as and when required whereas service method is used to create a service whose purpose is to do some defined task.

17) What are the filters in AngularJS?

Filters select a subset of items from an array and return a new array. Filters are used to show filtered items from a list of items based on defined criteria.

```
<div ng-app="myApp" ng-controller="personCtrl">
<p>The name is {{ firstName | uppercase }}</p>
</div>
<script>
angular.module('myApp', []).controller('personCtrl', function($scope) {
   $scope.firstName = "Sonoo",
   $scope.lastName = "Jaiswal"
});
</script>
```

18) Explain Currency Filter?

Currency filter formats text in a currency format. In below example, we've added currency filter to an expression returning number using pipe character. Here we've added currency filter to print fees using currency format:

```
Enter fees: <input type = "text" ng-model = "student.fees">
fees: {{student.fees | currency}}
```

19) Explain Filter Filter?

filter filter is used to filter the array to a subset of it based on provided criteria. In below example, to display only required subjects, we've used subjectName as filter:

```
Enter subject: <input type = "text" ng-model = "subjectName">
Subject:
<ul>
   <li ng-repeat = "subject in student.subjects | filter: subjectName">
      {{ subject.name + ', marks:' + subject.marks }}
   </li>
</ul>
```

20) Explain Uppercase Filter?

Uppercase filter converts a text to upper case text. In below example, we've added uppercase filter to an expression using pipe character. Here we've added uppercase filter to print student name in all capital letters:

```
Enter first name:<input type = "text" ng-model = "student.firstName">
Enter last name: <input type = "text" ng-model = "student.lastName">
Name in Upper Case: {{student.fullName() | uppercase}}
```

21) Explain Lowercase Filter?

Lowercase filter converts a text to lower case text. In below example, we've added lowercase filter to an expression using pipe character. Here we've added lowercase filter to print student name in all lowercase letters:

```
Enter first name:<input type = "text" ng-model = "student.firstName">
Enter last name: <input type = "text" ng-model = "student.lastName">
Name in Upper Case: {{student.fullName() | lowercase}}
```

22) Explain Order By Filter?

orderby filter orders the array based on provided criteria. In below example, to order subjects by marks, we've used orderBy marks:

```
Subject:
<ul>
   <li ng-repeat = "subject in student.subjects | orderBy:'marks'">
      {{ subject.name + ', marks:' + subject.marks }}
   </li>
</ul>
```

23) Explain directives in AngularJS?

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 ng–bind, ng–model etc to perform most of the task that developers have to do.

```
<div ng-app = "" ng-init = "countries = [{locale:'en-IND',name:'India'}, {locale:'en-
PAK',name:'Pakistan'}, {locale:'en-AUS',name:'Australia'}]">
      <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>
```

24) On Which Types Of Component Can We Create A Custom Directive?

Angular JS provides support to create custom directives for following type of elements.

- **Element directives :** Directive activates when a matching element is encountered:

```
<body ng-app="myApp">

<w3-test-directive></w3-test-directive>

<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
    return {
        template : "<h1>Made by a directive!</h1>"
    };
});
</script>
```

- **Attribute :** Directive activates when a matching attribute is encountered:

```
<body ng-app="myApp">

<div w3-test-directive></div>

<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
    return {
        template : "<h1>Made by a directive!</h1>"
    };
});
</script>
```

- **CSS :** Directive activates when a matching css style is encountered:

```
<body ng-app="myApp">

<div class="w3-test-directive"></div>

<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
    return {
        restrict : "C",
        template : "<h1>Made by a directive!</h1>"
    };
});
</script>
```

- **Comment :** Directive activates when a matching comment is encountered:

```
<body ng-app="myApp">

<!-- directive: w3-test-directive -->

<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
    return {
        restrict : "M",
        replace : true,
        template : "<h1>Made by a directive!</h1>"
    };
});
</script>
```

25) Is Angular Js Extensible?

- Yes! In AngularJS we can create custom directive to extend AngularJS existing functionalities.

- Custom directives are used in AngularJS to extend the functionality of HTML. Custom directives are defined using “directive” function. A custom directive simply replaces the element for which it is activated.

- AngularJS application during bootstrap finds the matching elements and do one time activity using its compile() method of the custom directive then process the element using link() method of the custom directive based on the scope of the directive.

```
   <body>
      <h2>AngularJS Sample Application</h2>

      <div ng-app = "mainApp" ng-controller = "StudentController">
         <student name = "Mahesh"></student><br/>
         <student name = "Piyush"></student>
      </div>

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

```
<script>
   var mainApp = angular.module("mainApp", []);

   mainApp.directive('student', function() {
      var directive = {};
      directive.restrict = 'E';
      directive.template = "Student: <b>{{student.name}}</b> ,
         Roll No: <b>{{student.rollno}}</b>";

      directive.scope = {
         student : "=name"
      }
      directive.compile = function(element, attributes) {
         element.css("border", "1px solid #cccccc");
```

```
      var linkFunction = function($scope, element, attributes) {
         element.html("Student: <b>"+$scope.student.name +"</b> ,
            Roll No: <b>"+$scope.student.rollno+"</b><br/>");
         element.css("background-color", "#ff00ff");
      }
      return linkFunction;
   }

   return directive;
```

```
      });
      mainApp.controller('StudentController', function($scope) {
         $scope.Mahesh = {};
         $scope.Mahesh.name = "Mahesh Parashar";
         $scope.Mahesh.rollno  = 1;

         $scope.Piyush = {};
         $scope.Piyush.name = "Piyush Parashar";
         $scope.Piyush.rollno  = 2;
      });
   </script>

</body>
```

26) Explain templates in AngularJS? How would we programmatically change the template of a directive?

- In AngularJS templates allows us to including external HTML into our currently loaded page in browser. The ***ngInclude*** directive is the simplest way to utilize client-side templating. This directive enables us to replace the associated Document Object

Model (DOM) element's inner HTML with a given template's HTML.

- For example below the ***customerInfoTable.html*** file, which will act like template and will be included in the main page with the help of ***ngInclude*** directive:

Content of **customerInfoTable.html** file:

```
<table class="table">
    <thead>
    <tr>
        <th>#</th>
        <th>Customer Name</th>
        <th>Address</th>
        <th>Income</th>
    </tr>
    </thead>
    <tr ng-repeat="customer in customerInfoList" ng-class="$odd ? 'odd' : 'even'">
        <td>{{$index + 1}}</td>
        <td>{{customer.customerName}}</td>
        <td>{{customer.address}}</td>
        <td>{{customer.income}}</td>
    </tr>
</table>
```

Content of **ngIncludeExample.html** file:

```
<html ng-app="mainModuleOfTemplate">
<head>
    <title>Example of Templating</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>
    <!-- CDN Link of Latest compiled and minified CSS  -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.m

    <script>
        var mainModuleOfTemplate=angular.module("mainModuleOfTemplate", []);
        mainModuleOfTemplate.controller("CrtCustomerInfo", function ($scope) {
                    $scope.customerInfoList = [
                        { customerName: "Tuli", address: "Dhaka Bangladesh", income: 45000 },
                        { customerName: "Rana", address: "Dhaka Bangladesh",  income: 60000},
                        { customerName: "Gomas", address:  "Dhaka Bangladesh", income: 80000},
                        { customerName: "Soton", address:  "Dhaka Bangladesh",  income: 25000}];
                });
    </script>
</head>
<body>
<div class="panel" ng-controller="CrtCustomerInfo">
    <h3 class="panel-header">Customer Information</h3>
    <ng-include src="'customerInfoTable.html'"></ng-include>
</div>
</body>
```

- In the controller of **ngIncludeExample.html** file we define a ***customerInfoList*** which contains some customer information and we represent this information to the view through help of ***ngInculde***directive by loading the **customerInfoTable.html** file. Furthermore, the ***ngInclude*** directive caches templates by their URL, so a given template is only loaded from the server once; and

it also support lazy loading.

- We can programmatically change the template of a directive by using the advanced compile and transclude settings. AngularJS directives' transclude setting and its corresponding **ngTransclude** directive exist to reference external HTML code from within our directive's HTML template. In other words, transclusion allows us to parameterize our directive's template, enabling us to modify some HTML in the template based on our needs.

- For example consider the html code below which dynamically change template of a directive:

```
<html ng-app= appModule >
<head>
    <title>Transclusion  Example</title>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
    </script>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js">
    </script>
    <script>
        var module = angular.module('appModule', []);
        module.directive('ngGreeting', function() {
            return {
                restrict: 'E',
                transclude: true,
                scope: {},
                template:
                    'This is example of dynamic template loading in directive ' +
                    '<span ng-transclude></span>',
            };

        });
    </script>
</head>

<body ng-init="initialValue = 'Welcome to the world ng-transclude of angularjs';">
    <ng-greeting>
        {{ initialValue }}
    </ng-greeting>
</body>
</html>
```

- As seen from the above code now we have a directive with a parameterized template. AngularJS update any HTML we put into the directive element into the directive's template.

27) Explain Ng-include Directive?

- 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>
```

Example:

```
<body>
   <h2>AngularJS Sample Application</h2>

   <div ng-app = "mainApp" ng-controller = "studentController">
      <div ng-include = "'/angularjs/src/include/main.htm'"></div>
      <div ng-include = "'/angularjs/src/include/subjects.htm'"></div>
   </div>

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

      mainApp.controller('studentController', function($scope) {
         $scope.student = {
            firstName: "Mahesh",
            lastName: "Parashar",
            fees:500,
```

```
            subjects:[
               {name:'Physics',marks:70},
               {name:'Chemistry',marks:80},
               {name:'Math',marks:65},
               {name:'English',marks:75},
               {name:'Hindi',marks:67}
            ],
            fullName: function() {
               var studentObject;
               studentObject = $scope.student;
               return studentObject.firstName + " " + studentObject.lastName;
            }
         };
      });
   </script>

</body>
```

main.htm

```
<table border = "0">
   <tr>
      <td>Enter first name:</td>
      <td><input type = "text" ng-model = "student.firstName"></td>
   </tr>

   <tr>
      <td>Enter last name: </td>
      <td><input type = "text" ng-model = "student.lastName"></td>
   </tr>

   <tr>
      <td>Name: </td>
      <td>{{student.fullName()}}</td>
   </tr>
</table>
```

subjects.htm

```
<p>Subjects:</p>
<table>
   <tr>
      <th>Name</th>
      <th>Marks</th>
   </tr>

   <tr ng-repeat = "subject in student.subjects">
      <td>{{ subject.name }}</td>
      <td>{{ subject.marks }}</td>
   </tr>
</table>
```

28) What is routing in AngularJS?

It is concept of switching views. AngularJS based controller decides which view to render based on the business logic. Example:

```
<script>
    var appModule = angular.module("routMechanism", ['ngRoute']);

    appModule.config(['$routeProvider',function($routeProvider) {
            $routeProvider.when('/customerinfo', {
                templateUrl: 'customerinfo.html',
                controller:  'CustomerInfoController'
            });
            $routeProvider.when('/productinfo', {
                templateUrl: 'productinfo.html',
                controller: 'ProductInfoController'
            });
            $routeProvider.when('/inventory', {
                templateUrl: 'inventory.html',
                controller: 'InventoryController'
            });
            $routeProvider.otherwise({
                templateUrl: 'index.html'
            });
        }]);
```

```
    appModule.controller("CustomerInfoController", function($scope) {
    });
    appModule.controller("ProductInfoController", function($scope) {

    });
    appModule.controller("InventoryController", function($scope) {

    });

</script>
```

```
<head>
    <title>AngularJS Routing Mechanism</title>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>
     <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.28//angular-route.min.js">
<body ng-app="routMechanism">

<a href="#/customerinfo">Customer Information</a><br/>
<a href="#/productinfo">Product Information</a><br/>
<a href="#/inventory">Inventory</a><br/>

<div ng-view></div> <!-router load the requested page content into this div element-->
```

- Here we load **angular.js** and **angular-route.js** script file. In the html **<body>** element we define ***ng-app*** module name that is ***routMechanism***. Also here we include three hyper link through **<a>**which are related to *customerinfo.html*, *productinfo.html* and *inventory.html* page.

- Next we include the **ng-view** directive inside the ** ** element where the page

content are loaded. In the main app module ***(appModule)*** we inject the **ngRoute** dependency. To set up our routes we call the ***config*** method on the ***appModule*** object. The **config** method takes a function as its argument which is executed when the module is loaded but before the application is executed. The ***$routeProvider*** declared as a dependency to the function that is passed to the ***config*** method.

- The ***$routeProvider*** is configured by calling the ***when()*** and ***otherwise()*** function. The ***when()***function takes two parameter first one is the route path and the second one is a JavaScript object. The route path is matched against the part of the given URL after the # when the application is loaded. The first parameter of ***when()*** function has is a JavaScript object which contains two properties one is the ***templateUrl*** through which we set which html page that the AngularJS should load and show inside the <**div ng-view**></**div**> element, and through ***contoller*** property we load the specific controller for that page.

- The ***otherwise()*** funciton execute when no route paths matchs with given url(In this case it load the index.html page).

29) What is deep linking in AngularJS?

- Deep linking is process of encoding page state, such as the state of radio button, check box or the window's current scroll position in URL. In AngularJS ***$routeProvider*** service provides a convenient interface for implementing deep linking by modifying the current URL without reloading the page.

- For example, consider the URL: http://www.google.com/foo?bar=baz#qux. The three portions of a URL that we need to be familiar with are the path: **/foo**; the query string after **'?'** symbol: **bar=baz**; and the hash string after **'#'** symbol: **qux**. The path and query string communicate with the server to locate precise resource we are looking for. Changing the path or the query string triggers a page reload in modern browsers. However, the browser does not send the hash portion to the server; thus, we can modify the hash component without triggering a page reload. The hash portion is typically used for deep linking functionality. For example consider the

following html page:

```
<!DOCTYPE html>
<html ng-app="deepModule">
<head>
    <!--cnd provider-->
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.6/angular.min.js"></script>
    <!--cnd provider-->
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.28//angular-route.min.js"><
    <script>
       var appModule = angular.module('deepModule', ['ngRoute']);
            /*define routing mechanism*/
            appModule.config( function($routeProvider) {
                    $routeProvider
                            .when('/pagefirst', {template: '<div><h2>Child page</h2><br>Content
                            .when('/pagesecond', {template: '<div><h2>Child page</h2><br>Content
                });
    </script>
```

```
    <title>Angular JS deep linking example</title>
</head>
<body>
<h1>Parent page</h1>
    <div>
        <a href="#/pagefirst'">First</a> | <a href="#/pagesecond'">Second</a>
        <hr/>
        <div ng-view>

        </div>
    </div>
</body>
</html>
```

- Here the content of URL before # remain same regardless of whether we clicking hyperlink **<a>** First or Second. Only portion of URL after # get changed and no page reloading happening; only portion of the page that related to hyperlink get updated.

30) Explain Angular Js Boot Process?

There are two ways to initialize AngularJS to our web application:

1. Initialized AngularJS through **DOMContentLoaded** event of the Browser.

2. Manually Initialized AngularJS by calling ***angular.bootstrap()*** methods of AngularJS framework.

- **Initialized AngularJS through DOMContentLoaded event of the Browser :** To initialized AngularJS through **DOMContentLoaded** event of the browser, we need to add ***"ng-app"*** directive to any HTML element and load

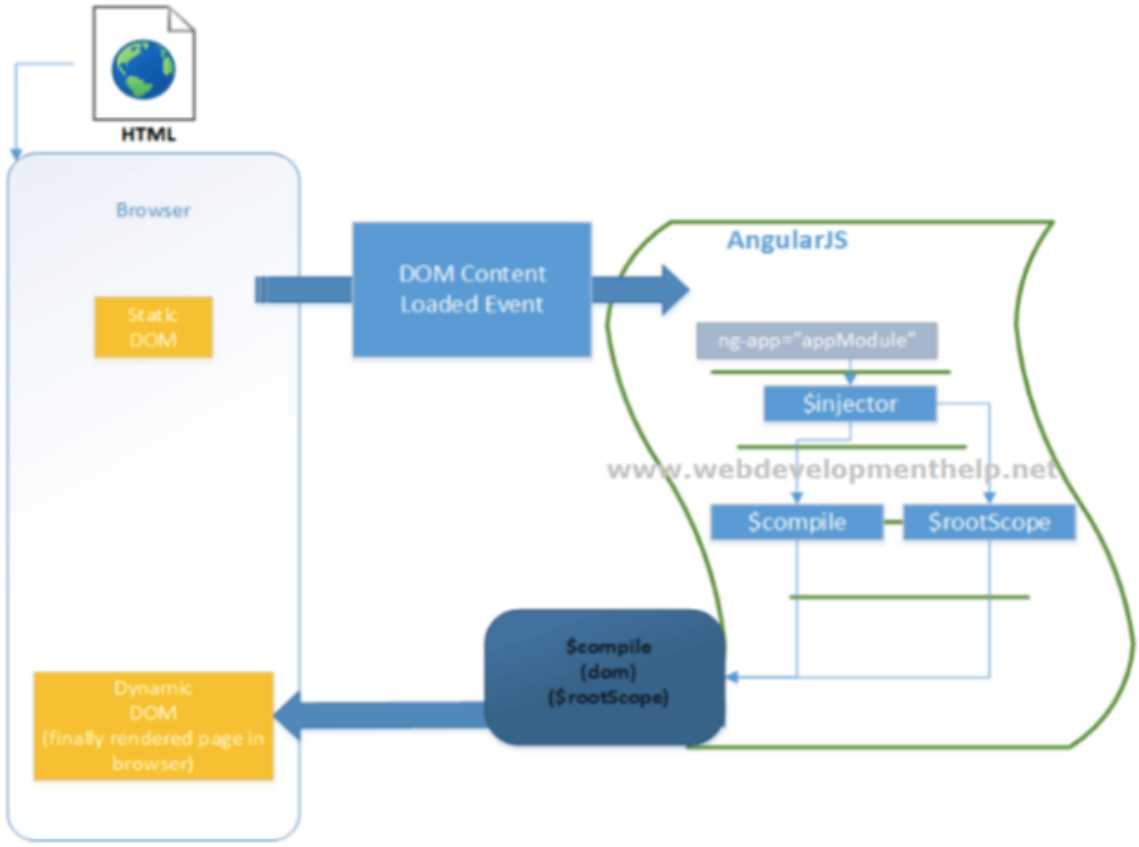
the *angular.js* script. For example:

```
<html>
 <body>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>
      <div id="ngScope" ng-app="appModule">
          Do you necessary stuff with angular JS.
      </div>
 </body>
```

When we load this page in browser AngularJS initializes automatically when **DOMContentLoaded** event is fired that is when the initial HTML document has been completely loaded and parsed (any JavaScript in this case angular.js), without waiting for stylesheets, images, and subframes to finish loading, Then angular looks for the ngApp(i.e. ***ng-app***) directive in this case div HTML element with id ***ngScope***. This ngApp directive define our application root and the HTML element(in this case div with id ***ngScope***) define the scope of the ngApp directive. If ngApp directive is found then Angular will do the following:

- Load any module associated with the directive.
- Create required application injector.
- Compile the DOM that contain ng-app directive as the starting point of the compilation. This allows us to consider only a portion of the DOM (in this case div element with ngScope id) as an AngularJS application.

Below diagram show the overview of AngularJS bootstrap process:



- Manually Initialized AngularJS by calling *angular.bootstrap()* methods of AngularJS framework **:** Through manual Initialization process we can have more control over the initialization process. In this case we need script loaders to define main app module and then as pre application requirement we can define controllers, services, directives, etc. via main app module:

```
<script>
    // first build custom main app module
    var customAppModule = angular.module('appModule', []);
    //add controller to the custom module
        customAppModule.controller('CtrlBoot',function ($scope) {
                $scope.name = 'World';
            });

        // Here we load the custom module to this page
        angular.element(document).ready(function() {
        angular.bootstrap(document, ['appModule']);
    });
</script>

<div ng-controller="CtrlBoot">
    Hello {{name}}!
</div>
```

Here we first create custom app module (***appModule***) then create the controller through this module. Finally we load custom app module to the html DOM object by calling the **angular**.*bootstrap()* function. Note that here we pass the custom module name as second parameter of the method and must create this custom modules before we inject this as a second parameter of **angular**.*bootstrap()* function. To load controllers, services, directives, etc. we must define those before calling **angular.*bootstrap()*** function. *We should not use the ng-app directive when we manually bootstrapping AngularJS application.*

31) Which Are The Core Directives Of Angular Js?

Following are the three core directives of AngularJS.

- •**ng-app :** This directive defines and links an AngularJS application to HTML.
- •**ng-model :** This directive binds the values of AngularJS application data to HTML input controls.
- •**ng-bind :** This directive binds the AngularJS Application data to HTML tags.

32) Explain Ng-model Directive?

ng-model directive binds the values of AngularJS application data to HTML input controls.

It creates a model variable which can be used with the html page and within the container control(for example, div) having ng-app directive:

```
<div ng-app="myApp" ng-controller="myCtrl">
    <input ng-model="name">
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.name = "John Doe";
});
</script>
```

33) Explain Ng-bind Directive?

ng-bind directive binds the AngularJS Application data to HTML tags. ng-bind updates the model created by ng-model directive to be displayed in the html tag whenever user input something in the control or updates the html control's data when model data is updated by controller:

```
<div ng-app="" ng-init="myText='Hello World!'">

<p ng-bind="myText"></p>

</div>
```

34) How Angular Js Integrates With Html?

AngularJS being a pure javaScript based library integrates easily with HTML.

- **Step-1 :** Include angularjs javascript libray in the html page:

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

- **Step-2 :** Point to AngularJS app. Next we tell what part of the HTML contains the AngularJS app. This done by adding the ng-app attribute to the root HTML element of the AngularJS app. You can either add it to html element or body element:

```
<body ng-app = "myapp">
</body>
```

35) Explain The Directives Ng-Repeat, Ng-Switch And Ng-If?

- **Ng-repeat :** ng-repeat directive repeats html elements for each item in a collection:

```
<body ng-app="myApp" ng-controller="myCtrl">

<h1 ng-repeat="x in records">{{x}}</h1>

<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
    $scope.records = [
        "Alfreds Futterkiste",
        "Berglunds snabbköp",
        "Centro comercial Moctezuma",
        "Ernst Handel",
    ]
});
</script>

</body>
```

- **Ng-switch :** The AngularJS ng-switch directive facilitates you to hide/show HTML elements according to an expression. Child elements with the ng-switch-when directive will be displayed if it gets a match, otherwise the element, and its children will be removed. If you want to define a default section you can use ng-switch-default directive.

```
<body ng-app="">
Choose your favorite topic:
<select ng-model="myVar">
  <option value="animals">Zoology
  <option value="tuts">Tutorials
  <option value="cars">Cars
  <option value="bikes">Bikes
</select>
<hr>
```

```
<div ng-switch="myVar">
 <div ng-switch-when="animals">
   <h1>Zoology</h1>
   <p>Welcome to a world of zoology.</p>
 </div>
 <div ng-switch-when="tuts">
   <h1>Tutorials</h1>
   <p>Learn from examples.</p>
 </div>
 <div ng-switch-when="cars">
   <h1>Cars</h1>
   <p>Read about cars.</p>
 </div>
 <div ng-switch-when="bikes">
   <h1>Cars</h1>
   <p>Read about bikes.</p>
</div>

 <div ng-switch-default>
   <h1>Switch</h1>
   <p>Select topic from the dropdown, to switch the content of this DIV.</p>
 </div>
</div>
<hr>
<p>The ng-switch directive hides and shows HTML sections depending on a certain value.</p>
</body>
```

- **Ng-if :** The AngularJS ng-if directive is used to remove the HTML elements if the expression is set to false. If the if element is set to true, a copy of the element is added in the DOM. ngIf is different from ngShow and ngHide which show and hide the elements while ngIf completely removes and recreates the element in the DOM rather than changing its visibility. It is supported by all HTML elements.

```
<body ng-app="">
Keep HTML: <input type="checkbox" ng-model="myVar" ng-init="myVar = true">
<div ng-if="myVar">
<h1>Welcome to JavaTpoint!</h1>
<p>A solution of all technologies..</p>
<hr>
</div>
<p>If you uncheck the checkbox then DIV element will be removed.</p>
<p>If you check the checkbox then DIV element will return.</p>
</body>
```

36) What Are Angular Js Expressions?

Expressions are used to bind application data to html. Expressions are written inside double braces like {{ expression}}. Expressions behave in same way as ng-bind directives. AngularJS application expressions are pure JavaScript expressions and outputs the data where they are used:

```
   <body>
      <h1>Sample Application</h1>

      <div ng-app = "" ng-init = "quantity = 1;cost = 30;
         student = {firstname:'Mahesh',lastname:'Parashar',rollno:101};
         marks = [80,90,75,73,60]">
         <p>Hello {{student.firstname + " " + student.lastname}}!</p>
         <p>Expense on Books : {{cost * quantity}} Rs</p>
         <p>Roll No: {{student.rollno}}</p>
         <p>Marks(Math): {{marks[3]}}</p>
      </div>

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

   </body>
```

37) What Directives Are Used To Show And Hide HTML Elements In AngularJS?

The directives used to show and hide HTML elements in the AngularJS are <ng-show> and <ng-hide>

- **Ng-show :** ng-show directive shows a given control. In below example, we've added ng-show attribute to a HTML button and pass it a model. Then we've attached the model to a checkbox and can see the variation:

```
<input type = "checkbox" ng-model = "showHide1">Show Button
<button ng-show = "showHide1">Click Me!</button>
```

- **Ng-hide :** ng-hide directive hides a given control. In below example, we've added ng-hide attribute to a HTML button and pass it a model. Then we've attached the model to a checkbox and can see the variation:

```
<input type = "checkbox" ng-model = "showHide2">Hide Button
<button ng-hide = "showHide2">Click Me!</button>
```

- **Example :**

```
<div ng-hide="true">if ng-hide is true then show contents of this div.</div>
<div ng-show="true">if ng-show is true then show contents of this div. </div>


<!DOCTYPE html>
<html>
<head>
  <script src="http://code.angularjs.org/1.2.0-rc.2/angular.js"></script>
  <script>
    function mainController($scope) {
      $scope.display = {
        ngShow: true,
        ngHide: true,
        showContent: 'This is ng-show and ng-hide in angularjs contents'
      };
    }
  </script>
</head>
```

```
</head>
<body ng-app>
  <div ng-controller="mainController">
    <h2>ng-show and ng-hide in angularjs</h2>
    <div>
      <div ng-show="display.ngShow">
        ng-Show : {{display.ngShow}} , {{display.showContent}}
      </div>

      <div ng-hide="display.ngHide">
        {{display.ngHide}} : {{display.hideContent}}
      </div>
    </div>
  </div>
</body>
</html>
```

38) Explain Ng-click Directive?

- ng-click directive represents a AngularJS click event. In below example, we've added ng-click attribute to a HTML button and added an expression to updated a model. Then we can see the variation:

```
<button ng-click="count = count + 1" ng-init="count=0">OK</button>
<p>The button has been clicked <strong>{{count}} </strong>times.</p>
```

- **Example:**

```
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body ng-app="myApp">
<div ng-controller="myCtrl">
 <p>Click the button to execute a function:</p>
 <button ng-click="myFunc()">OK</button>
 <p>The button has been clicked <strong>{{count}}</strong> times.</p>
</div>
<script>
```

```
angular.module('myApp', [])
  .controller('myCtrl', ['$scope', function($scope) {
    $scope.count = 0;
    $scope.myFunc = function() {
      $scope.count++;
    };
  }]);
</script>
</body>
</html>
```

39) Explain Ng-disabled Directive?

- ng-disabled directive disables a given control. In below example, we've added ng-disabled attribute to a HTML button and pass it a model. Then we've attached the model to an checkbox and can see the variation:

```
<input type = "checkbox" ng-model = "enableDisableButton">Disable Button
<button ng-disabled = "enableDisableButton">Click Me!</button>
```

- **Example:**

```
<!DOCTYPE html>
<html>
<head lang="en">
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.4
<script>
    var $scope;
    var app = angular.module('myapp', []);
    function myControl($scope) {
    }
</script>
</head>
```

```
</head>
<body>
    <div ng-app="myapp">
    <div ng-controller="myControl">
        First Name :  <input type="text" ng-model="fName" /><br>
        Last Name :  <input type="text" ng-model="lName" /><br>
        <button ng-disabled="!fName || !lName">Submit</button>
    </div>
</div>
</body>
```

40) How Angular.module Works?

- angular.module is used to create AngularJS modules along with its dependent modules. Consider the following example:

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

- Here we've declared an application mainApp module using angular.module function. We've passed an empty array to it. This array generally contains dependent modules declared earlier.

41) How To Validate Data In Angular Js?

AngularJS enriches form filling and validation. We can use $dirty and $invalid flags to do the validations in seamless way. Use novalidate with a form declaration to disable any browser specific validation. 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.

Example:

```
<body>

  <h2>AngularJS Sample Application</h2>
  <div ng-app = "mainApp" ng-controller = "studentController">

    <form name = "studentForm" novalidate>
      <table border = "0">
        <tr>
          <td>Enter first name:</td>
          <td><input name = "firstname" type = "text" ng-model = "firstName" required>
            <span style = "color:red" ng-
show = "studentForm.firstname.$dirty && studentForm.firstname.$invalid">
              <span ng-show = "studentForm.firstname.$error.required">First Name is required.</span>
            </span>
          </td>
        </tr>

        <tr>
          <td>Enter last name: </td>
          <td><input name = "lastname"  type = "text" ng-model = "lastName" required>
            <span style = "color:red" ng-
show = "studentForm.lastname.$dirty && studentForm.lastname.$invalid">
              <span ng-show = "studentForm.lastname.$error.required">Last Name is required.</span>
            </span>
          </td>
        </tr>

        <tr>
          <td>Email: </td><td><input name = "email" type = "email" ng-
model = "email" length = "100" required>
            <span style = "color:red" ng-show = "studentForm.email.$dirty && studentForm.email.$invalid">
              <span ng-show = "studentForm.email.$error.required">Email is required.</span>
              <span ng-show = "studentForm.email.$error.email">Invalid email address.</span>
            </span>
          </td>
        </tr>
```

```
        <tr>
          <td>
            <button ng-click = "reset()">Reset</button>
          </td>
          <td>
            <button ng-disabled = "studentForm.firstname.$dirty &&
              studentForm.firstname.$invalid || studentForm.lastname.$dirty &&
              studentForm.lastname.$invalid || studentForm.email.$dirty &&
              studentForm.email.$invalid" ng-click="submit()">Submit</button>
          </td>
        </tr>
    </table>

    </form>
  </div>
   <script>
    var mainApp = angular.module("mainApp", []);
    mainApp.controller('studentController', function($scope) {
      $scope.reset = function(){
        $scope.firstName = "Sonoo";
        $scope.lastName = "Jaiswal";
        $scope.email = "sonoojaiswal@javatpoint.com";
      }
       $scope.reset();
    });
  </script>
</body>
```

42) Mention what are the styling form that ngModel adds to CSS classes?

ngModel adds these CSS classes to allow styling of form as well as control:

- ng-untouched The field has not been touched yet.

- ng-touched The field has been touched.
- ng-pristine The field has not been modified yet.
- ng-dirty The field has been modified.
- ng-valid The field content is valid.
- ng-invalid The field content is not valid.

Example:

```
<div ng-controller="ExampleController">
  <form novalidate class="css-form">
    <label>Name: <input type="text" ng-model="user.name" required /></label><br />
    <label>E-mail: <input type="email" ng-model="user.email" required /></label><br />
    Gender: <label><input type="radio" ng-model="user.gender" value="male" />male</label>
    <label><input type="radio" ng-model="user.gender" value="female" />female</label><br />
    <input type="button" ng-click="reset()" value="Reset" />
    <input type="submit" ng-click="update(user)" value="Save" />
  </form>
  <pre>user = {{user | json}}</pre>
  <pre>master = {{master | json}}</pre>
</div>
```

```
<style type="text/css">
  .css-form input.ng-invalid.ng-touched {
    background-color: #FA787E;
  }

  .css-form input.ng-valid.ng-touched {
    background-color: #78FA89;
  }
</style>
```

```
<script>
  angular.module('formExample', [])
    .controller('ExampleController', ['$scope', function($scope) {
      $scope.master = {};

      $scope.update = function(user) {
        $scope.master = angular.copy(user);
      };

      $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
      };

      $scope.reset();
    }]);
</script>
```

43) How To Make An Ajax Call Using Angular Js?

AngularJS provides $http control which works as a service to make ajax call 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, $http can be used to get the data from server in the following manner:

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
  $http({
    method : "GET",
      url : "welcome.htm"
  }).then(function mySuccess(response) {
    $scope.myWelcome = response.data;
  }, function myError(response) {
    $scope.myWelcome = response.statusText;
  });
});
```

44) What Is Use Of $routeprovider In Angular Js?

$routeProvider is the key service which set the configuration of urls, maps them with the corresponding html page or ng-template, and attaches a controller with the same.

```
app.config(function($routeProvider) {
  $routeProvider
  .when("/", {
    templateUrl : "main.htm"
  })
  .when("/red", {
    templateUrl : "red.htm"
  })
  .when("/green", {
    templateUrl : "green.htm"
  })
  .when("/blue", {
    templateUrl : "blue.htm"
  });
});
```

45) What Is $rootscope?

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. $rootScope is the parent of all of the scope variables.

```
<body ng-app="myApp">

<p>The rootScope's favorite color:</p>
<h1>{{color}}</h1>

<div ng-controller="myCtrl">
  <p>The scope of the controller's favorite color:</p>
  <h1>{{color}}</h1>
</div>

<p>The rootScope's favorite color is still:</p>
<h1>{{color}}</h1>

<script>
var app = angular.module('myApp', []);
app.run(function($rootScope) {
  $rootScope.color = 'blue';
});
app.controller('myCtrl', function($scope) {
  $scope.color = "red";
});
</script>
</body>
```

46) What Is Value?

Value is a simple JavaScript object, which is required to pass values to the controller during config phase (config phase is when AngularJS bootstraps itself):

```
//define a module
var mainApp = angular.module("mainApp", []);

//create a value object as "defaultInput" and pass it a data.
mainApp.value("defaultInput", 5);
...

//inject the value in the controller using its name "defaultInput"
mainApp.controller('CalcController', function($scope, CalcService, defaultInput)
{
   $scope.number = defaultInput;
   $scope.result = CalcService.square($scope.number);

   $scope.square = function() {
      $scope.result = CalcService.square($scope.number);
   }
});
```

47) What Is Provider?

Provider is used by AngularJS internally to create services, factory, etc. during the config phase. The following script can be used to create MathService that we created earlier. Provider is a special factory method with get() method which is used to return the value/ service/factory:

```
//define a module
var mainApp = angular.module("mainApp", []);
...

//create a service using provider which defines a method square to return square
of a number.
mainApp.config(function($provide) {
   $provide.provider('MathService', function() {
      this.$get = function() {
         var factory = {};

         factory.multiply = function(a, b) {
            return a * b;
         }
         return factory;
      };
   });
});
```

48) What Is Constant?

Constants are used to pass values at the config phase considering the fact that value cannot be used during the config phase.

```
mainApp.constant("configParam", "constant value");
```

49) Which Components Can Be Injected As A Dependency In Angular Js?

Angular JS 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

Example:

```
<body>
  <h2>AngularJS Sample Application</h2>

  <div ng-app = "mainApp" ng-controller = "CalcController">
    <p>Enter a number: <input type = "number" ng-model = "number" /></p>
    <button ng-click = "square()">X<sup>2</sup></button>
    <p>Result: {{result}}</p>
  </div>
```

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

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

  mainApp.config(function($provide) {
    $provide.provider('MathService', function() {
      this.$get = function() {
        var factory = {};

        factory.multiply = function(a, b) {
          return a * b;
        }
        return factory;
      };
    });
  });

  mainApp.value("defaultInput", 10);

  mainApp.factory('MathService', function() {
    var factory = {};

    factory.multiply = function(a, b) {
      return a * b;
    }
    return factory;
  });


  mainApp.service('CalcService', function(MathService){
    this.square = function(a) {
      return MathService.multiply(a,a);
    }
  });
```

```
mainApp.controller('CalcController', function($scope, CalcService, defaultInput) {
  $scope.number = defaultInput;
  $scope.result = CalcService.square($scope.number);

  $scope.square = function() {
    $scope.result = CalcService.square($scope.number);
  }
});
 </script>
</body>
```

50) When To Use Factory?

It is just a collection of functions, like a class. Hence, it can be instantiated in different controllers when you are using it with a constructor function.

51) How Do You Exchange Data Among Different Modules Of Your Angular JS Application?

There are a no. of ways in Angular to share data among modules. A few of them are as follows:

- The most common method is to create an Angular service to hold the data and dispatch it across the modules.
- Angular has a matured event system which provides $broadcast(), $emit() and $on() methods to raise events and pass data among the controllers.
- We can also use $parent, $nextSibling, and $ controllerAs to directly access the controllers.
- Variables defined at the root scope level ($rootScope) are available to the controller scope via prototypical inheritance. But they behave like globals and hard to maintain.

52) How Would You Use An Angular Service To Pass Data Between Controllers?

Explain With Examples?

- Using a service is the best practice in Angular to share data between controllers. Here is a step by step example to demonstrate data transfer.

- We can prepare the data service provider in the following manner:

```
app.service('dataService', function() {

var dataSet = [];



var addData = function(newData) {

dataSet.push(newData);

};
```

```
var getData = function(){

return dataSet;

};



return {

addData: addData,

getData: getData

};

});
```

- Now, we'll inject the service dependency into the controllers.
- Say, we have two controllers – pushController and popController.
- The first one will add data by using the data service provider's addData method. And the latter will fetch this data using the service provider's getData method:

```
app.controller('pushController', function($scope,
dataService) {

$scope.callToAddToProductList = function(currObj){

dataService.addData(currObj);

};

});



app.controller('popController', function($scope,
dataService) {

$scope.dataSet = dataService.getData();

});
```

53) What Would You Do To Limit A Scope Variable To Have One-Time Binding?

By prefixing the "::" operator to the scope variable. It'll make sure the candidate is aware of the available variable bindings in AngularJS.

54) Which Angular Directive Would You Use To Hide An Element From The DOM Without Modifying Its Style?

It is the conditional ngIf Directive which we can apply to an element. Whenever the condition becomes false, the ngIf Directive removes it from the DOM.

55) What Is The Difference Between The $Watch, $Digest, And $Apply?

In AngularJS $scope object is having different functions like $watch(), $digest() and $apply() and we will call these functions as central functions. The AngularJS central functions $watch(), $digest(), and $apply() are used to bind data to variables in view and observe changes happening in variables.

A) $Watch() –

The use of this function is to observe changes in a variable on the $scope. It triggers a function call when the value of that variable changes. It accepts three parameters: expression, listener, and equality object. Here, listener and equality objects are optional parameters:

```
<script>

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

var myController = myapp.controller("myController",
function

($scope) {

$scope.name = 'dotnet-tricks.com';

$scope.counter = 0;

//watching change in name value
```

```
$scope.$watch('name', function (newValue, oldValue)
{

$scope.counter = $scope.counter + 1;

});

});

</script>

</head>
```

```
<body ng-app="myapp" ng-
controller="myController">

<input type="text" ng-model="name" />

<br /><br />

Counter: {{counter}}

</body>

</html>
```

B) $Digest() –

This function iterates through all the watch list items in the $scope object, and its child objects (if it has any). When $digest() iterates over the watches, it checks if the value of the expression has changed or not. If the value has changed, AngularJS calls the listener with the

new value and the old value. The $digest() function is called whenever AngularJS thinks it is necessary. For example, after a button click, or after an AJAX call. You may have some cases where AngularJS does not call the $digest() function for you. In that case, you have to call it yourself:

```
<script src="lib/jquery-1.11.1.js"></script>

<script src="lib/angular.js"></script>

</head>

<body ng-app="app">

<div ng-controller="Ctrl">

<button class="digest">Digest my scope!</button>

<br />
```

```
<h2>obj value : {{obj.value}}</h2>

</div>

<script>

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

app.controller('Ctrl', function ($scope) {

$scope.obj = { value: 1 };

$('.digest').click(function () {

console.log("digest clicked!");

console.log($scope.obj.value++);
```

```
//update value

$scope.$digest();

});

});

</script>

</body>
```

$Apply() –

AngularJS automatically updates the model changes which are inside AngularJS context. When you apply changes to any model, that lies outside of the Angular context (like browser DOM events, setTimeout, XHR or third party libraries), then you need to inform the Angular about the changes by calling $apply() manually. When the $apply() function call finishes, AngularJS calls $digest() internally, to update all data bindings.

Following are the key differences between $apply() and $digest().

- Its use is to update the model properties forcibly.
- The $digest() method evaluates the watchers for the current scope. However, the $apply() method is used to evaluate watchers for root scope, that means it's for all scopes.

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

var myController = myapp.controller("myController",
function

($scope) {

$scope.datetime = new Date();

$scope.updateTime = function () {

$scope.datetime = new Date();

}
```

```
//outside angular
context        document.getElementById("updateTimeButton").addEventListener('click',
function () {

//update the value

$scope.$apply(function () {

console.log("update time clicked");

$scope.datetime = new Date();

console.log($scope.datetime);

});

});

});

</script>

</head>

<body ng-app="myapp" ng-
controller="myController">
```

```
<button ng-click="updateTime()">Update time – ng-
click</button>

<button id="updateTimeButton">Update
time</button>

<br />

{{datetime | date:'yyyy-MM-dd HH:mm:ss'}}

</body>
```

56) Explain what is linking function and type of linking function?

Link combines the directives with a scope and produces a live view. For registering DOM listeners as well as updating the DOM, link function is responsible. After the template is cloned it is executed.

- Pre-linking function: Pre-linking function is executed before the child elements are linked. It is not considered as the safe way for DOM transformation.
- Post linking function: Post linking function is executed after the child elements are linked. It is safe to do DOM transformation by post-linking function.

57) Explain what is injector?

An injector is a service locator. It is used to retrieve object instances as defined by the provider, instantiate types, invoke methods and load modules. There is a single injector per Angular application, it helps to look up an object instance by its name.

58) Explain what is the difference between link and compile in Angular.js?

- Compile function: It is used for template DOM Manipulation and collect all of the

directives.

- Link function: It is used for registering DOM listeners as well as instance DOM manipulation. It is executed once the template has been cloned.

59) Who created Angular JS ?

Initially, it was developed by Misko Hevery and Adam Abrons. Currently, it is being developed & Maintained by Google.

60) What Is Internationalization?

Internationalization is a way to show locale specific information on a website. For example, display content of a website in English language in United States and in Danish in France.

61) How To Implement Internationalization In Angular Js?

- Angular JS supports inbuilt internationalization for three types of filters currency, date and numbers. We only need to incorporate corresponding js according to locale of the country. By default it handles the locale of the browser. For example, to use Danish locale, use following script:

```
<script src = "https://code.angularjs.org/1.2.5/i18n/angular-locale_da-dk.js">
</script>
```

- **Example:**

```
   <body>
      <h2>AngularJS Sample Application</h2>

      <div ng-app = "mainApp" ng-controller = "StudentController">
         {{fees | currency }}  <br/><br/>
         {{admissiondate | date }}  <br/><br/>
         {{rollno | number }}
      </div>

      <script src =
"https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
      </script>
      <script src = "https://code.angularjs.org/1.3.14/i18n/angular-locale_da-
dk.js">
      </script>
```

```
       <script>
          var mainApp = angular.module("mainApp", []);

          mainApp.controller('StudentController', function($scope) {
             $scope.fees = 100;
             $scope.admissiondate  = new Date();
             $scope.rollno = 123.45;
          });
       </script>

    </body>
```