\*\*\*\*\*\*\*\*\*\*ANGULARJS INTERVIEW’S MATERIALS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Top 153 AngularJS Interview Questions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1) What is AngularJS?**

AngularJS is a javascript framework i.e. used for creating single web page applications.  It allows you to use HTML as your template language and enables you to extend HTML’s syntax to express your application’s components clearly. It follows MVC (Model View Controller) pattern. It is open source, cross browser compliant and easy to maintain.

OR

1. AngularJS is an open-source JavaScript framework developed by Google.

2. It helps you create single-page applications or one-page web applications that only require HTML, CSS, and JavaScript on the client side.

3. It is based on MV-\* pattern and allow you to build well structured, easily testable, and maintainable front-end applications.

4. AngularJS is not based on JQuery to perform its operations. In spite of using ASP.NET Web from, ASP.NET MVC, PHP, JSP, Ruby on Rails for web development, you can do your complete web development by using most powerful and adaptive JavaScript Framework AngularJS.

OR

AngularJS is open source client side MV\* (Model – View – Whatever) framework for creating dynamic web applications. It gives life to your static HTML and makes it dynamic with its magic. It extends HTML using directives, expression and data binding techniques to define a powerful HTML template.

OR

AngularJSis an open-source JavaScript framework, maintained by Google, that assists with running single-page applications. Its goal is to augment browser-based applications with model–view–controller capability, in an effort to make both development and testing easier.

OR

*“AngularJS is a JavaScript framework which simplifies binding JavaScript objects with HTML UI elements.”*

Let us try to understand the above definition with simple sample code.

Below is a simple “Customer” function with “CustomerName” property. We have also created an object called as “Cust” which is of “Customer” class type.

function Customer()

{

this.CustomerName = "AngularInterview";

}

var Cust = new Customer();

Now let us say the above customer object we want to bind to a HTML text box called as “TxtCustomerName”. In other words when we change something in the HTML text box the customer object should get updated and when something is changed internally in the customer object the UI should get updated.

<input type=text id="TxtCustomerName" onchange="UitoObject()"/>

So in order to achieve this communication between UI to object developers end up writing functions as shown below. “UitoObject” function takes data from UI and sets it to the object while the other function “ObjecttoUI” takes data from the object and sets it to UI.

function UitoObject()

{

Cust.CustomerName = $("#TxtCustomerName").val();

}

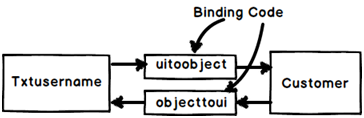
function ObjecttoUi()

{

$("#TxtCustomerName").val(Cust.CustomerName);

}

So if we analyze the above code visually it looks something as shown below. Your both functions are nothing but binding code logic which transfers data from UI to object and vice versa.



Now the same above code can be written in Angular as shown below. The javascript class is attached to a HTML parent div tag using “ng-controller” directive and the properties are binded directly to the text box using “ng-model” declarative.

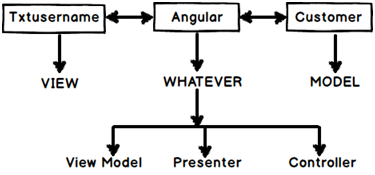
So now whatever you type in the textbox updates the “Customer” object and when the “Customer” object gets updated it also updates the UI.

<div ng-controller="Customer">

<input type=text id="txtCustomerName" ng-model="CustomerName"/>

</div>

In short if you now analyze the above code visually you end up with something as shown in the below figure.You have the VIEW which is in HTML, your MODEL objects which are javascript functions and the binding code in Angular.



Now that binding code have different vocabularies.

* Some developers called it “ViewModel” because it connects the “Model” and the “View” .
* Some call it “Presenter” because this logic is nothing but presentation logic.
* Some term it has “Controller” because it controls how the view and the model will communicate.

To avoid this vocabulary confusion Angular team has termed this code as “Whatever”. It’s that “Whatever” code which binds the UI and the Model. That’s why you will hear lot of developers saying Angular implements “MVW”

**2) Why to use AngularJs?**

1. It is based on MVC pattern which helps you to organize your web apps or web application properly.

2. It extends HTML by attaching directives to your HTML markup with new attribute or tags and expressions in order to define very powerful templates.

3. It also allows you create your own directives, marking reusable components that fill your needs and abstract your DOM manipulation logic.

4. It supports two-way data binding i.e. connects your HTML (views) to your JavaScript objects (models) seamlessly. In this way any change in model will update the view and vice versa without any DOM manipulation or event handling .

5. It encapsulates the behavior of your application in controllers which are instantiated with help of dependency injection.

6. It supports services that can be injected into your controllers to use some utility code to fulfil your need. For example, it provides $http service to communicate with REST service.

OR

AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop. 

* MVC implementation is done right.
* It extends HTML using directives, expression and data binding techniques to define a powerful HTML template.
* Two way data-binding, form validations, routing supports, inbuilt services.
* REST friendly.
* Dependency injection support.
* It helps you to structure and test your JavaScript code.

**3) How does the AngularJS framework initialize itself?**

The AngularJS has a self-executing anonymous function present in angular.js code, which does the initialization of AngularJS. Here is how below it looks,

(function(window, document, undefined) {

<!--

here goes entire AngularJS code

including functions, services, providers etc related code goes here

-->

if (window.angular.bootstrap) {

//AngularJS is already loaded, so we can return here...

console.log('WARNING: Tried to load angular more than once.');

return;

}

//try to bind to jquery now so that one can write

angular.element().read()

//but we will rebind on bootstrap again.

bindJQuery();

publishExternalAPI(angular);

jqLite(document).ready(function() {

angularInit(document, bootstrap);

});

})(window, document);

Above function first check if Angular has already been setup. If it has, we return here to prevent Angular from trying to initialize itself a second time. And then it binds jQuery (if present) and publish external API. And finally angularInit() method does the trick for initialization of AngularJS.

**4) Explain what are the key features of AngularJS ?**

The key features of AngularJS are

* Scope

The job of the Scope is to detect changes to model objects and create an execution context for expressions. There is one root scope for the application (ng-app) with hierarchical children scopes. It marshals the model to the view and forwards events to the controller.

* Controller

The Controller is responsible for construction of the model and connects it to the view (HTML). The scope sits between the controller and the view. Controllers should be straightforward and simply contain the business logic needed for a view. Generally you want thin controllers and rich services. Controllers can be nested and handle inheritance. The big difference in AngularJS from the other JavaScript frameworks is there is no DOM manipulation in controllers. It is something to unlearn when developing in AngularJS.

* Model

In AngularJS, a Model is simply a JavaScript object. No need to extend anything or create any structure. This allows for nested models  - something that Backbone doesn’t do out-of-the-box.

* View

The View is based on DOM objects, not on strings. The view is the HTML. HTML is declarative – well suited for UI design. The View should not contain any functional behavior. The flexibility here is to allow for multiple views per Controller.

* Services

The Services in AngularJS are singletons that perform common tasks for web applications. If you need to share common functionality between Controllers, then use Services. Built-in AngularJS, Services start with a $. There are several ways to build a service: Service API, Factory API, or the $provide API.

* Data Binding

Data Binding in AngularJS is a two-way binding between the View and the Model. Automatic synchronizing between views and data models makes this really easy (and straightforward) to use. Updating the model is reflected in View without any explicit JavaScript code to bind them together, or to add event listeners to reflect data changes.

* Directives

Now this is cool. AngularJS allows you to use Directives to transform the DOM or to create new behavior. A directive allows you to extend the HTML vocabulary in a declarative fashion. The ‘ng’ prefix stands for built-in AngularJS directives. The App (ng-app), Model (ng-model), the Controller (ng-controller), etc. are built into the framework. AngularJS allows for building your own directives. Building directives is not extremely difficult, but not easy either. There are different things that can be done with them. Please check out AngularJS’s documentation on directives.

* Filters

The Filters in AngularJS perform data transformation. They can be used to do formatting (like I did in my Directives example with padding zeros), or they can be used to do filter results (think search).

* Testable

Testing is a big concern for enterprise applications. There are several different ways to write and run tests against JavaScript code, thus against AngularJS. The developers at AngularJS advocate using Jasmine tests ran using Testacular. I have found this method of testing very straightforward and, while writing tests may not be the most enjoyable, it is just as importable as any other piece of developing an application.

* Validation

AngularJS has some built-in validation around HTML5 input variables (text, number, URL, email, radio, checkbox) and some directives (required, pattern, minlength, maxlength, min, max). If you want to create your own validation, it is just as simple as creating a directive to perform your validation.

* Modules
* Templates
* Expressions
* Routing
* Dependency Injection
* Testing

**5) Explain what is scope in AngularJS ?**

Scope refers to the application model, it acts like glue between application controller and the view.  Scopes are arranged in hierarchical structure and impersonate the DOM ( Document Object Model) structure of the application.  It can watch expressions and propagate events.

OR

Scope is an object that represents **application model**. Each AngularJS application can have only **one root scope but can have multiple child scopes.**

OR

A scope is an object that ties a view (a DOM element) to the controller. In the MVC framework, scope object is your model. In other words, it is just a JavaScript object, which is used for communication between controller and view.

OR

Scope is an object that refers to the application model. It is the glue between application controller and the view. Both the controllers and directives have reference to the scope, but not with each other. It is an execution context for expressions and arranged in hierarchical structure. Scopes can watch expressions and propagate events.

## 6) Explain $scope in Angular?

“$scope” is an object instance of a controller. “$scope” object instance get’s created when “ng-controller” directive is encountered.

For example in the below code snippet we have two controllers “Function1” and “Function2”. In both the controllers we have a “ControllerName” variable.

function Function1($scope)

{

$scope.ControllerName = "Function1";

}

function Function2($scope)

{

$scope.ControllerName = "Function2";

}

Now to attach the above controllers to HTML UI we need to use “ng-controller” directive. For instance you can see in the below code snippet how “ng-controller” directive attaches “function1” with “div1” tag and “function2” with “div2” tag.

<div id="div1" ng-controller="Function1">

Instance of {{ControllerName}} created

</div>

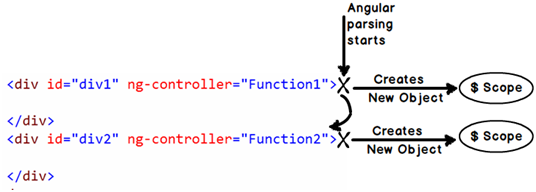
<div id="div2" ng-controller="Function2">

Instance of {{ControllerName}} created

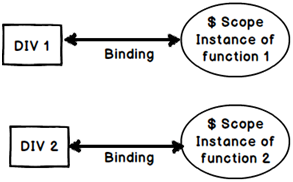
</div>

So this is what happens internally. Once the HTML DOM is created Angular parser starts running on the DOM and following are the sequence of events:-

* The parser first finds “ng-controller” directive which is pointing to “Function1”. He creates a new instance of “$scope” object and connects to the “div1” UI.
* The parser then starts moving ahead and encounters one more “ng-controller” directive which is pointing to “Function2”. He creates a new instance of “$scope” object and connects to the “div2” UI.



Now once the instances are created, below is a graphical representation of the same. So the “DIV1” HTML UI is binded with “function1” $scope instance and the “DIV2” HTML UI is binded with “function2” $scope instance. In other words now anything changes in the $scope object the UI will be updated and any change in the UI will update the respective $scope object.



**7) What is the difference between $scope and scope in AngularJS?**

$scope is used while defining a controller (see previous question) where scope is used while creating a link function for custom directive. The common part is that they both refers to scope object in AngularJS, but the difference is that $scope uses dependency injection where scope doesn't. When the arguments are passed-in via dependency injection, their position doesn't matter. So for example, a controller defined as ($scope as first parameter)

myApp.controller('MyController', ['$scope', function($scope, $http) {

OR ($scope is second parameter)

myApp.controller('MyController', ['$scope', function($http, $scope) {

In both the case, the postion of $scope doesn't matter to AngularJS. Because AngularJS uses the argument name to get something out of the dependency-injection container and later use it.  
  
But in case of link function, the position of scope does matter because it doesn't uses DI. The very first parameter has to be scope and that's what AngularJS expects.

app.directive("myDirective", function() {

return {

scope: {};

link: function(scope, element, attrs) {

// code goes here.

}

};

});

However you can name it anything of your choice. In below code, foo is your scope object.

link: function(foo, bar, baz) {

// code goes here.

}

**8) 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.

OR

The $rootScope is the top-most scope. An app can have only one $rootScope which will be shared among all the components of an app. Hence it acts like a global variable. All other $scopes are children of the $rootScope. Since $rootScope is a global, which means that anything you add here, automatically becomes available in $scope in all controller. To add something in $rootScope, you need to use app.run function which ensures that it will run prior to the rest of the app. You may say that "run" function is like "main" method of angular app.

app.run(function($rootScope) {

$rootScope.name = "AngularJS";

});

And then you can use in your view. (via expression)

<body ng-app="myApp">

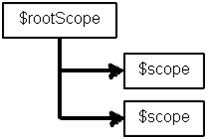
<h1>Hello {{ name }}!</h1>

</body>

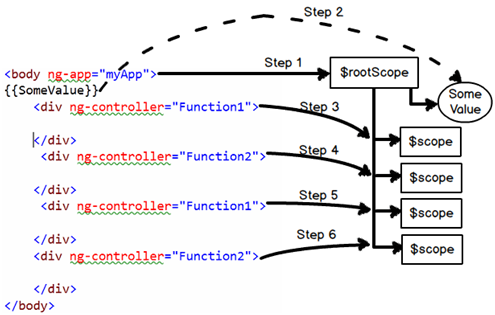
**9) What is $rootscope and how it is useful?**   
$rootScope is just like a global variable which has the top most scope associated. Any app can have only one specific $rootScope which can be shared among all the components for that app.   
Rest all $scopes will act as a children to the $rootScope. Since it is global any changes made to this will reflect in $scope of all other controllers. To add any thing in $rootScope we have to invoke app.run function. So it will ensure to run before to all other functionalities of the app.   
app. run(function($rootScope) {   
$rootScope.name = “Angu1arDotNetFunda”;  
}):   
The same can be used to print in this way   
<body ng-app=“myApp”>  
<hl>Hello {{ name }}!</hl>   
</body>

## 10) What is “$rootScope” and how is it related with “$scope”?

“$rootScope” is a parent object of all “$scope” angular objects created in a web page.



Let us understand how Angular does the same internally. Below is a simple Angular code which has multiple “DIV” tags and every tag is attached to a controller. So let us understand step by step how angular will parse this and how the “$rootScope” and “$scope” hierarchy is created.



The Browser first loads the above HTML page and creates a DOM (Document object model) and Angular runs over the DOM.Below are the steps how Angular creates the rootscope and scope objects.

* Step 1:- Angular parser first encounters the “ng-app” directive and creates a “$rootScope” object in memory.
* Step 2:- Angular parser moves ahead and finds the expression {{SomeValue}}. It creates a variable
* Step 3:- Parser then finds the first “DIV” tag with “ng-controller” directive which is pointing to “Function1” controller. Looking at the “ng-controller” directive it creates a “$scope” object instance for “Function1” controller. This object it then attaches to “$rootScope” object.
* Step 4:- Step 3 is then repeated by the parser every time it finds a “ng-controller” directive tag. Step 5 and Step 6 is the repetition of Step 3.

If you want to test the above fundamentals you can run the below sample Angular code. In the below sample code we have created controllers “Function1” and “Function2”. We have two counter variables one at the root scope level and other at the local controller level.

<script language="javascript">

function Function1($scope, $rootScope)

{

$rootScope.Counter = (($rootScope.Counter || 0) + 1);

$scope.Counter = $rootScope.Counter;

$scope.ControllerName = "Function1";

}

function Function2($scope, $rootScope)

{

$rootScope.Counter = (($rootScope.Counter || 0) + 1);

$scope.ControllerName = "Function2";

}

var app = angular.module("myApp", []); *// creating a APP*

app.controller("Function1", Function1); *// Registering the VM*

app.controller("Function2", Function2);

</script>

Below is the HTML code for the same. You can we have attached “Function1” and “Function2” two times with “ng-controller” which means four instances will be created.

<body ng-app="myApp" id=1>

Global value is {{Counter}}<br />

<div ng-controller="Function1">

Child Instance of {{ControllerName}} created :- {{Counter}}

</div><br />

<div ng-controller="Function2">

Child Instance of {{ControllerName}} created :- {{Counter}}

</div><br />

<div ng-controller="Function1">

Child Instance of {{ControllerName}} created :- {{Counter}}

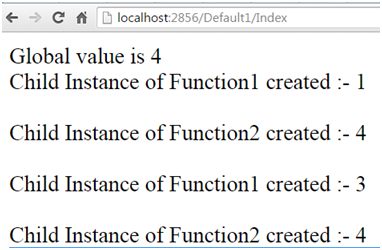
</div><br />

<div ng-controller="Function2">

Child Instance of {{ControllerName}} created :- {{Counter}}

</div><br />

</body>



Above is the output of the code you can see the global variable of root scope has be incremented four times because four instances of $scope have been created inside “$rootScope” object.

**11) Explain what is services in AngularJS ?**

In AngularJS services are the singleton objects or functions that are used for carrying out specific tasks.  It holds some business logic and these function can be called as controllers, directive, filters and so on.

OR

Services are objects that can be used to store and share data across the application. AngularJS offers many built-in services such as $http service i.e. used to make XMLHttpRequests(Ajax calls).

OR

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.

OR

Service helps to implement dependency injection. For instance let’s say we have the below “Customer” class who needs “Logger” object. Now “Logger” object can be of “FileLogger” type or “EventLogger” type.

function Customer($scope,$http, Logger)

{

        $scope.Logger = Logger;

}

So you can use the “service” method of the application and tie up the “EventLogger” object with the “Logger” input parameter of the “Customer” class.

var app = angular.module("myApp", []); *// creating a APP*

app.controller("Customer", Customer); *// Registering the VM*

app.service("Logger", EventLogger); *// Injects a global Event logger object*

So when the controller object is created the “EventLogger” object is injected automatically in the controller class.

## 12) Are Service object instances global or local?

Angular Services create and inject global instances. For example below is a simple “HitCounter” class which has a “Hit” function and this function increments the variable count internally every time you call hit the button.

function HitCounter()

{

       var i = 0;

        this.Hit = function ()

        {

            i++;

            alert(i);

        };

}

This “HitCounter” class object is injected in “MyClass” class as shown in the below code.

function MyClass($scope, HitCounter)

{

$scope.HitCounter = HitCounter;

}

Below code advises the Angular framework to inject “HitCounter” class instance in the “MyClass” class. Read the last line of the below code specially which says to inject the inject the “HitCounter” instance.

var app = angular.module("myApp", []); *// creating a APP*

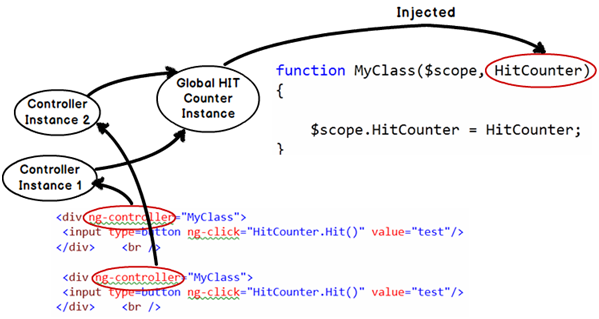
app.controller("MyClass", MyClass); *// Registering the VM*

app.service("HitCounter", HitCounter); *// Injects the object*

Now let’s say that the “Controller” “MyClass” is attached to twodiv tag’s as shown in the below figure.

So two instances of “MyClass” will be created. When the first instance of “MyClass” is created a “HitCounter” object instance is created and injected in to “MyClass” first instance.

When the second instance of “MyClass” is created the same “HitCounter” object instance is injected in to second instance of “MyClass”.  
Again I repeat the same instance is injected in to the second instance, new instances are not created.



If you execute the above code you will see counter values getting incremented even if you are coming through different controller instances.

**13) Explain what is Angular Expression? Explain what is key difference between angular expressions and JavaScript expressions?**

Like JavaScript,  Angular expressions are code snippets that are usually placed in binding such as {{ expression }}

Below are some examples of angular expressions:-

The below expression adds two constant values.

{{1+1}}

The below expression multiplies quantity and cost to get the total value.

The value total cost is {{ quantity \* cost }}

The below expression displays a controller scoped variable.

<div ng-controller="CustomerVM">

The value of Customer code is {{CustomerCode}}

</div>

The value of Customer code is {{CustomerCode}}

AngularJS supports one-time binding expressions.

The key difference between the JavaScript expressions and Angular expressions

* **Context :** In Angular, the expressions are evaluated against a scope object, while the Javascript expressions are evaluated against the global window
* **Forgiving:** In Angular expression evaluation is forgiving to null and undefined, while in Javascript undefined properties generates TypeError or ReferenceError
* **No Control Flow Statements:** Loops, conditionals or exceptions cannot be used in an angular expression
* **Filters:** To format data before displaying it you can use filters
* **No Comma And Void Operators:** You cannot use , (comma) or void in an Angular expression. And You cannot create regular expressions in an Angular expression.

**14) With options on page load how you can initialize a select box ?**

You can initialize a select box with options on page load by using **ng-init** directive

* <div ng-controller = “ apps/dashboard/account ” ng-switch
* On = “! ! accounts” ng-init = “ loadData ( ) ”>

**15) How will you initialize a select box with options on page load?**

Use the ng-init directive.

<div ng-controller="apps/dashboard/account" ng-switch

on="!!accounts" ng-init="loadData()">

**16) How will you add options to a select box?**

Using the ng-options and ng-model directives.

<fieldset>

<dl class="control-group">

<dt>

<label for="cientId">

<h4>Client Id:</h4>

</label>

</dt>

<dd>

<select id="cientId" class="input-xlarge" ng-model="clientId"

ng-options="reportClient.clientId as reportClient.clientId for reportClient in reportClients "

ng-click="getReportParams()" ng-change="getValuationDates()" />

</dd>

</dl>

<dl class="control-group">

<dt>

<label for="valuationDate">

<h4>

Valuation Date <small>(dd/mm/yyyy)</small>

</h4>

</label>

</dt>

<dd>

<select id="valuationDate" class="input-xlarge"

ng-model="valuationDate"

ng-options="reportdate for reportdate in reportDates" />

</dd>

</dl>

</fieldset>

**17) Explain what are directives ? Mention some of the most commonly used directives in AngularJS application ?**

A directive is something that introduces new syntax, they are like markers on DOM element which attaches a special behavior to it. All AngularJS directives start with the word "ng". In any AngularJS application, directives are the most important components.

Some of the commonly used directives are **ng-model, ng-app, ng-bind, ng-repeat , ng-show, ng-controller** etc.

Let's see a simple example of AngularJS directive.

**<input** type="text" id="empName"  ng-model="EmpName"**/>**

OR

Directives are attributes decorated on the HTML elements. All directives start with the word “ng”. As the name says directive it directs Angular what to do.

For example below is a simple “ng-model” directive which tells angular that the HTML textbox “txtCustomerName” has to be binded with the “CustomerName” property.

<input type=text id="txtCustomerName" ng-model="CustomerName"/>

Some of the most commonly used directives are ng-app,ng-controller and ng-repeat.

OR

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.

OR

Directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS to attach a specified behavior to that DOM element or even transform the DOM element and its children. When AngularJS finds the directive at the time of rendering then it attaches the requested behavior to the DOM element. Angular comes with a set of these directives built-in, like ngBind, ngModel, and ngClass.

**18) Mention what are the advantages of using AngularJS ?**

AngularJS has several advantages in web development.

* AngularJS supports MVC pattern
* Can do two ways data binding using AngularJS
* It has per-defined form validations
* It supports both client server communication
* It supports animations
* AngularJS provides reusable components.
* With AngularJS, developer writes less code and gets more functionality
* allows us to create single page application(SPA)
* open source
* cross browser compliant
* its code are unit testable
* AngularJS applications can run on all major browsers and smart phones including Android and iOS based phones/tablets.
* Customize & extensible
* Compatibility
* testing

**19) Explain what Angular JS routes does ?**

Angular js routes enable you to create different URLs for different content in your application.  Different URLs for different content enables user to bookmark URLs to specific content.  Each such bookmarkable URL in AngularJS is called a route

A value in Angular JS is a simple object.  It can be a number, string or JavaScript object.  Values are typically used as configuration injected into factories, services or controllers. A value should be belong to an AngularJS module.

Injecting a value into an AngularJS controller function is done by adding a parameter with the same name as the value

**20)  Explain what is data binding in AngularJS ?**

Automatic synchronization of data between the model and view components is referred as data binding in AngularJS. ng-model directive is used in data binding.

There are two ways for data binding

1. **Data mining in classical template systems(**One way data binding**)**
2. **Data binding in angular templates(**Two way data binding**)**

## 21)  How is the data binding in Angular ?

Its two way binding. So whenever you make changes in one entity the other entity also gets updated.

**22)  What makes AngularJS better ?**

* **Registering Callbacks:** There is no need to register callbacks . This makes your code simple and easy to debug.
* **Control HTML DOM programmatically:**All the application that are created using Angular never have to manipulate the DOM although it can be done if it is required
* **Transfer data to and from the UI:**AngularJS helps to eliminate almost all of the boiler plate like validating the form, displaying validation errors, returning to an internal model and so on which occurs due to flow of marshalling data
* **No initilization code:**With AngularJS you can bootstrap your app easily using services, which auto-injected into your application in Guice like dependency injection style

**23)  Explain what is string interpolation in Angular.js ?**

In Angular.js the compiler during the compilation process matches text and attributes using interpolate service to see if they contains embedded expressions.  As part of normal digest cycle these expressions are updated and registered as watches.

**24)  Mention the steps for the compilation process of HTML happens?**

Compilation of HTML process occurs in following ways

* Using the standard browser API, first the HTML is parsed into DOM
* By using the call to the $compile () method, compilation of the DOM is performed.  The method traverses the DOM and matches the directives.
* Link the template with scope by calling the linking function returned from the previous step

## 25) What kind of naming conventions is used for custom directives?

For angular custom directive the best practice is to follow camel casing and that also with atleast two letter’s. In camel case naming convention we start with a small letter, followed by a capital letter for every word.

Some example of camel cases are “loopCounter” , “isValid” and so on.

So when you register a custom directive it should be with camel case format as shown in the below code “companyCopyRight”.

app.directive('companyCopyRight', function ()

{

return

{

        template: '@CopyRight questpond.com '

};

});

Later when this directive is consumed inside HTML before each capital letter of camel case we need to insert a “-“ as specified in the below code.

<div company-copy-right></div>



If you are making a one letter prefix like “copyright” it’s very much possible that tomorrow if HTML team creates a tag with the same name, it will clash with your custom directive. That’s why angular team recommends camel case which inserts a “-“ in between to avoid further collision with future HTML tag’s.

**26) On which types of component can write create a custom directive?**

**AngularJS provides support to create custom directive for following type of elements.**

* Element directives- Directive activates when a matching element is encountered.
* Attribute- Directive activates when a matching attribute is encountered.
* CSS- Directive activates when a matching css style is encountered.
* Comment- Directive activates when a matching comment is encountered.

**27)  Explain what is directive and Mention what are the different types of Directive or custom directive?**

During compilation process when specific HTML constructs are encountered a behaviour or function is triggered, this function is referred as directive.  It is executed when the compiler encounters it in the DOM.

There are different flavors of Angular directives depending till what level you want to restrict your custom directive.

In other words do you want your custom directive to be applied only on HTML element or only on an attribute or just to CSS etc.

Different types of directives are

* Element directives(E)
* Attribute directives(A)
* CSS class directives(C)
* Comment directives(M)

Below is a simple custom directive implementation at the element level.

myapp.directive('userinfo', function()

{

    var directive = {};

    directive.restrict = 'E';

    directive.template = "User : {{user.firstName}} {{user.lastName}}";

    return directie;

});

The “restrict” property is set to “E” which means that this directive can only be used at element level as shown in the code snippet below.

<userinfo></userinfo>

If you try to use it at an attribute level as shown in the below code it will not work.

<div userinfo></div>

So “E” for element, “A” for attribute, “C” for CSS and “M” for comments.

## 28) What if I want custom directives to be applied on element as well as attributes ?

directive.restrict = 'EA';

**29)  Explain what is linking function and type of linking function?**

Link combines the directives with a scope and produce 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

**30)  Explain what is injector?**

An injector is a service locator.  It is used to retrieve object instances as defined by 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.

**31)  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.

## 32) Explain compile and link phase?

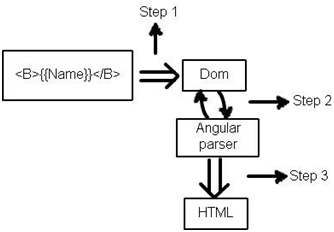
At the heart Angular framework is a parser. A parser which parses the Angular directives and render’s HTML output.

Angular parser works in 3 steps:-

Step 1:- HTML browser parses the HTML and creates a DOM (Document Object Model).

Step 2:- Angular framework runs over this DOM looks at the Angular directives and manipulates the DOM accordingly.

Step 3:- This manipulated is then rendered as HTML in the browser.

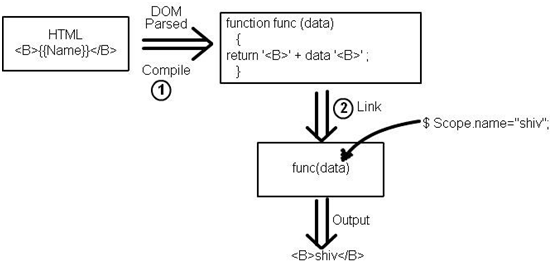


Now the above angular parsing is not so simple as it looks to be. It occurs in two phases “Compile” and “Link”. Firs the compile phase occurs then the link phase.

https://www.codeproject.com/KB/aspnet/891718/angular_parser_.2.png

In compile phase the angular parser starts parsing the DOM and whenever the parser encounters a directive it create a function. These functions are termed as template or compiled functions. In this phase we do not have access to the $scope data.

In the link phase the data i.e. ($scope) is attached to the template function and executed to get the final HTML output.



**33)  Explain what is factory method in AngularJS?**

For creating the directive, factory method is used.  It is invoked only once, when compiler matches the directive for the first time.  By using $injector.invoke the factory method is invoked.

OR

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

}

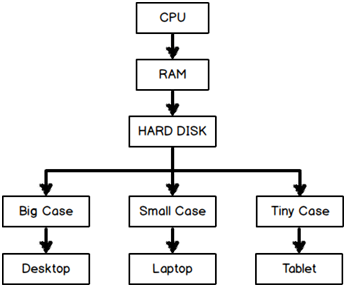
return factory;

});

## 34) What is a Factory in Angular?

“Factory” in real world means a premise where products are manufactured. Let’s take an example of a computer manufacturing firm. Now the company produces different kinds and sizes of computers likelaptops,desktops, tablets etc.

Now the process of manufacturing the computer products are same with slight variation. To manufacture any computer we need processor, RAM and hard disk. But depending on what kind of final case packing is the final product shapes.



That’s what the use of Factory in Angular.

For example see the below code we have a “Customer”, “Phone” and “Address” class.

function Customer()

{

        this.CustomerCode = "1001";

        this.CustomerName = "Shiv";

}

function Phone()

{

        this.PhoneNumber = "";

}

function Address()

{

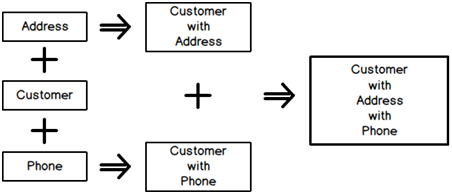
        this.Address1 = "";

        this.Address2 = "";

}

So now we would create different types of “Customer” object types using the combination of “Address” and “Phones” object.

* We would like to combine “Customer” with “Address” and create a “Customer” object which has “Address” collection inside it.
* Or must be we would like to create “Customer” object with “Phone” objects inside it.
* Or must be “Customer” object with both “Phone” and “Address” objects.



In other words we would like to have different permutation and combination to create different types of “Customer” objects.

So let’s start from bottom. Let’s create two factory function’s one which creates “Address” object and the other which creates “Phone” objects.

functionCreateAddress()

{

var add = new Address();

return add;

}

functionCreatePhone()

{

var phone =  new Phone();

return phone;

}

Now let’s create a main factory function which uses the above two small factory functions and gives us all the necessary permutation and combination.

In the below factory you can see we have three functions:-

* “CreateWithAddress” which creates “Customer” with “Address” objects inside it.
* “CreateWithPhone” which creates “Customer” object with “Phone” objects inside it.
* “CreateWithPhoneAddress” which creates “Customer” object with aggregated “Phone” and “Address” objects.

function CreateCustomer() {

return {

CreateWithAddress: function () {

varcust = new Customer();

cust.Address = CreateAddress();

returncust;

            },

CreateWithPhone: function () {

varcust = new Customer();

cust.Phone = {};

cust.Phone = CreatePhone();

returncust;

            }

            ,

CreateWithPhoneAddress: function () {

debugger;

varcust = new Customer();

cust.Phone = CreatePhone();

cust.Address = CreateAddress();

returncust;

            }

        }

    }

Below is a simple “CustomerController” which takes “CustomerFactory” as the input. Depending on “TypeOfCustomer” it creates with “Address” , “Phones” or both of them.

functionCustomerController($scope, Customerfactory)

    {

        $scope.Customer = {};

        $scope.Init = function(TypeofCustomer)

        {

if (TypeofCustomer == "1")

            {

                $scope.Customer = Customerfactory.CreateWithAddress();

            }

if (TypeofCustomer ==  "2")

            {

                $scope.Customer = Customerfactory.CreateWithPhone();

            }

if (TypeofCustomer == "3") {

                $scope.Customer = Customerfactory.CreateWithPhoneAddress();

            }

        }

    }

You also need to tell Angular that the “CreateCustomer” method needs to be passed in the input. For that we need to call the “Factory” method and map the “CreateCustomer” method with the input parameter “CustomerFactory” for dependency injection.

var app = angular.module("myApp", []); *// creating a APP*

app.controller("CustomerController", CustomerController); *// Register the VM*

app.factory("Customerfactory", CreateCustomer);

So if we consume the “CustomerController” in UI , depending on situation it creates different flavors of “Customer” object. You can in the below code we have three different “DIV” tags and depending on the “TypeofCustomer” we are displaying data.



## 35) What is the difference between Factory and Service?

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.

OR

“Factory” and “Service” are different ways of doing DI (Dependency injection) in angular. Please read the previous question to understand what is DI.

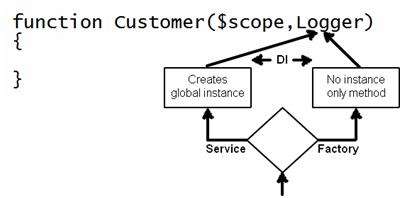
So when we define DI using “service” as shown in the code below. This creates a new GLOBAL instance of the “Logger” object and injects it in to the function.

app.service("Logger", Logger); *// Injects a global object*

When you define DI using a “factory” it does not create a instance. It just passes the method and later the consumer internally has to make calls to the factory for object instances.

app.factory("Customerfactory", CreateCustomer);

Below is a simple image which shows visually how DI process for “Service” is different than “Factory”.



|  |  |  |
| --- | --- | --- |
|  | **Factory** | **Service** |
| Usage | When we want to create different types of objects depending on scenarios. For example depending on scenario we want to create a simple “Customer” object , or “Customer” with “Address” object or “Customer” with “Phone” object. See the previous question for more detailed understanding. | When we have utility or shared functions to be injected like Utility , Logger , Error handler etc. |
| Instance | No Instance created. A method pointer is passed. | Global and Shared instance is created. |

**36)  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- valid
* ng- invalid
* ng-pristine
* ng-dirty

**37)  Mention what are the characteristics of “Scope”?**

* To observer model mutations scopes provide APIs ($watch)
* To propagate any model changes through the system into the view from outside of the Angular realm
* A scope inherits properties from its parent scope,  while providing access to shared model properties, scopes can be nested to isolate application components
* Scope provides context against which expressions are evaluated

**38)  Explain what is DI (Dependency Injection ) and how an object or function can get a hold of its dependencies ?**

DI or Dependency Injection is a software design pattern that deals with how code gets hold of its dependencies.  In order to retrieve elements of the application which is required to be configured when module gets loaded , the operation “config” uses dependency injection.

These are the ways that object uses to hold of its dependencies

* Typically using the new operator, dependency can be created
* By referring to a global variable, dependency can be looked up
* Dependency can be passed into where it is required

## 39) What is dependency injection and how does it work in Angular?

Dependency injection is a process where we inject the dependent objects rather than consumer creating the objects. DI is everywhere in Angular or we can go one step ahead and say Angular cannot work without DI.

For example in the below code “$scope” and “$http” objects are created and injected by the angular framework. The consumer i.e. “CustomerController” does not create these objects himself rather Angular injects these objects.

function CustomerController($scope,$http)

{

*// your consumer would be using the scope and http objects*

}

## 40)  How does DI benefit in Angular?

There are two big benefits of DI: - Decoupling and Testing.

Let’s first start with Decoupling. Consider your application has a logger functionality which helps to log errors , warning etc in some central place. This central place can be a file, event viewer, database etc.

function FileLogger()

{

        this.Log = function () {

            alert("File logger");

        };

}

function EventLogger()

{

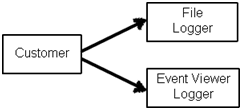
        this.Log = function () {

            alert("Event viewer logger");

        };

}

Now let’s say you have a “Customer” class who wants to use the “Logger” classes. Now which “Logger” class to use depends on configuration.



So the code of “Customer” is something as shown below. So depending on the configuration “Customer” class either creates “FileLogger” or it creates “EventLogger” object.

function Customer($scope, Logger)

{

        $scope.Logger = {};

        if (config.Loggertype = "File")

{

            $scope.Logger = new FileLogger();

        }

        else

{

            $scope.Logger = new EventLogger();

        }

}

But with DI our code becomes something as shown below. The “Customer” class says he is not worried from where the “Logger” object comes and which type of “Logger” objects are needed .He just wants to use the “Logger” object.

function Customer($scope,$http, Logger)

{

        $scope.Logger = Logger;

}

With this approach when a new “Logger” object gets added the “Customer” class does not have to worry about the new changes because the dependent objects are injected by some other system.  
The second benefit of DI is testing. Let’s say you want to test the “Customer” class and you do not have internet connection. So your “$http” object method calls can throw errors. But now you can mock a fake “$http” object and run your customer class offline without errors.The fake object is injected using DI.

**41)  Mention what are the advantages of using Angular.js framework ?**

Advantages of using Angular.js as framework are

* Supports two way data-binding
* Supports MVC pattern
* Support static template and angular template
* Can add custom directive
* Supports REST full services
* Supports form validations
* Support both client and server communication
* Support dependency injection
* Applying Animations
* Event Handlers

**42)  Explain the concept of scope hierarchy?  How many scope can an application have?**

Each angular application consist of one root scope but may have several child scopes. As child controllers and some directives create new child scopes, application can have multiple scopes. When new scopes are formed or created they are added as a children of their parent scope. Similar to DOM, they also creates a hierarchical structure.

OR

Each Angular application has exactly one root scope, but may have several child scopes. The application can have multiple scopes, because child controllers and some directives create new child scopes. When new scopes are created, they are added as children of their parent scope. This creates a hierarchical structure similar to the DOM where they're attached.

When Angular evaluates a bound variable like say {{firstName}}, it first looks at the scope associated with the given element for the firstName property. If no such property is found, it searches the parent scope and so on until the root scope is reached. In JavaScript this behaviour is known as prototypical inheritance, and child scopes prototypically inherit from their parents. The reverse is not true. i.e. the parent can't see it's children's bound properties.

**43)  Explain what is the difference between AngularJS and backbone.js?**

AngularJS combines the functionalities of most of the 3rd party libraries, it supports individual functionalities required to develop HTML5 Apps.  While Backbone.js do their jobs individually.

**44)  Who created Angular JS ?**

Intially it was developed by Misko Hevery and Adam Abrons. Currently it is being developed by Google.

**45) What are the disadvantages of AngularJS?**

1. **JavaScript Dependent**: If end user disables JavaScript, AngularJS will not work.
2. **Not Secured**: It is JavaScript based framework so it is not safe to authenticate user through AngularJS only.
3. Server side authentication and authorization is must to keep an application secure.
4. **Not degradable:** If your application user disables JavaScript then user will just see the basic page and nothing more.

## 46) Is AngularJS dependent on JQuery? Or Do I need Jquery for Angular?

No, you do not need Jquery for Angular. It’s independent of Jquery.

### 47) What IDE's are currently used for the development of AngularJS?

1. Eclipse
2. Visual Studio(2012,13,15 or higher)
3. WebStorm
4. Sublime Text
5. TextMate etc.

### 48) What are controllers in AngularJS?

Controllers are JavaScript functions that are used to provide data and logic to HTML UI. It acts as an interface between Server and HTML UI.

OR

ControllersareJavaScript functions that are bound to a particularscope.   
They are the prime actors in AngularJSframework and carry functions to   
operate on dataand decide whichview is to beupdated to show the updated model based data.

OR

In Angular, a Controller is a JavaScript constructor function. When a Controller is attached to the DOM via the ng-controller directive, Angular will instantiate a new Controller object, using the specified Controller's constructor function. The job of a controller is to pass data from the model, to the view or the view can asks for something from the controller, and the controller turns to the model and takes that thing, and hands it back to the view.

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

myApp.controller('MyController', ['$scope', function($scope) {

$scope.website = 'jquerybyexample.net';

}

]);

And then in your HTML using ng-controller directive,

<div ng-controller="MyController">

<h1>My website address is {{ website }}!</h1>

</div>

### 49) What are the usage of controllers in AngularJS?

AngularJS Controllers are used to:

1. Set up initial state of the $scope object, and
2. Add behavior to the $scope object.

### 50) What is template in AngularJS?

A template consists of HTML, CSS and AngularJS directives that are used to render dynamic view.

OR

Templates are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using partials.

OR

In Angular, templates are written with HTML that contains Angular-specific elements and attributes. Angular combines the template with information from the model and controller to render the dynamic view that a user sees in the browser. In other words, if your HTML page is having some Angular specific elements/attributes it becomes a template in AngularJS.

**51) What are the filters in Angular JS?**   
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.

OR

Filter is used to filter the array to a subset of it based on provided criteria.

### 52) What is the use of filter in AngularJS?

A filter is used to format the value of expression to display the formatted output. AngularJS enables us to write our own filter.

## 53) What are controllers and need of ng-controller and ng-model in Angular?

“Controllers” are simple javascript function which provides data and logic to HTML UI. As the name says controller they control how data flows from the server to HTML UI.

https://www.codeproject.com/KB/aspnet/891718/ajs.4.png

For example below is simple “Customer” controller which provides data via “CustomerName” and “CustomerCode” property and Add/ Update logic to save the data to database.

|  |
| --- |
| Note: - Do not worry too much about the $scope , we will discuss the same in the next question. |

function Customer($scope)

{

        $scope.CustomerName = "Shiv";

        $scope.CustomerCode = "1001";

        $scope.Add = function () {

        }

        $scope.Update = function () {

        }

}

“ng-controller” is a directive.Controllers are attached to the HTML UI by using the “ng-controller” directive tag and the properties of the controller are attached by using “ng-model” directive. For example below is a simple HTML UI which is attached to the “Customer” controller via the “ng-controller” directive and the properties are binded using “ng-model” directive.

<div ng-controller="Customer">

<input type=text id="CustomerName" ng-model="CustomerName"/><br />

<input type=text id="CustomerCode" ng-model="CustomerCode"/>

</div>

## 54) How can we initialize Angular application data?

We can use “ng-init” directive to achieve the same. You can see in the below example we have used “ng-init” directive to initialize the “pi” value.

<body ng-app="myApp" ng-init="pi=3.14">

The value of pi is {{pi}}

</body>

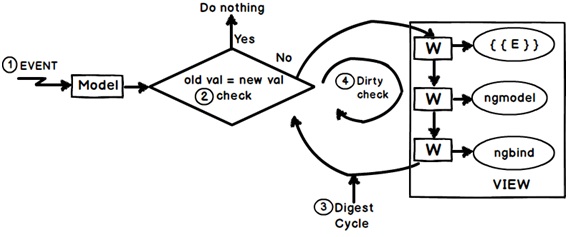
## 55) Explain the concept of digest cycle, watchers and dirty checking?

Angular is a MVW framework. It helps us to bind the model and the view. In other words when any change happens in the model the view gets updated. This updation of the model and the view is done by a loop called as digest cycle.

Digest cycle follows four important steps:-

1. Step 1:- Some kind of event is triggered by the end user like typing (onchange), button click etc and due to this activity model value changes.
2. Step 2:- Angular first checks if the new value and old values are same. If they are same he does not do anything. If they are not it then it invokes the digest cycle.
3. Step 3:- Digest cycle then runs through the scope objects to check which objects are getting affected because of this change. Every object in the scope have watchers. Watchers as the name says it listens whether the model has changed or not. Digest cycle informs the watchers about the model change and then watchers synchronize the view with the model data.
4. Step 4 :- In step 3 watchers update the view and due that update its very much possible that the model changes again. Now due to this model change we have to reevaulate the view again. So the digest loop runs once again to ensure that all things are synched up. This second loop which runs is termed as dirty check loop.

Below is the figure where in we have highlighted all the four steps.



So summarizing definitions for the above three concepts:-

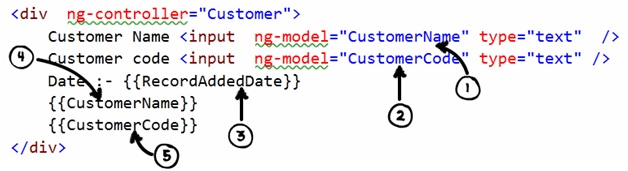
* Digest cycle: - It is a simple loop which updates the model and view.
* Watchers :- They are listeners which are attached to expression and angular directives and fire when the model data changes.
* Dirty check :- This is a extra digest loop which runs to check any cascading left over updates due to the first digest cycle.

## 56) What can be the performance implications of watchers and digest cycle ?

If there lot of unnecessary watchers then digest cycle has to work harder. As per AngularJS team having more than 2000 watchers on Angular screen is a bad practice.

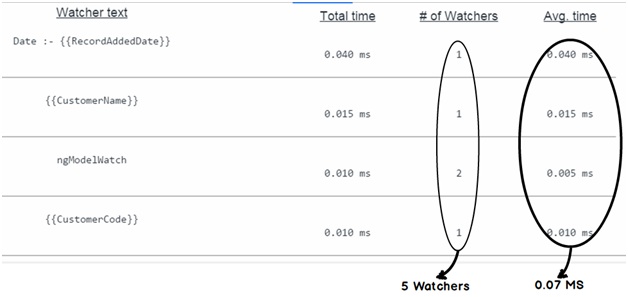
## 57) How can we measureno: of watchers & time spent on digest cycle?

Consider the below simple example where we have two ng-models and three expression. So in all we should have 5 watchers for the below screen



There are lot of great open source tools which help you to figure out the number of watchers , one such tool is the “batarang” tool. It’s a simple Google chrome extension which you can install separately.

Below is a simple snapshot where in we ran the above program , pressed f12 , enabled batarang and below are the results. You can see that he is showing 5 total watchers and for that digest cycle ran for 0.07 MS.



## 58) How can we decrease digest cycle time ?

To decrease digest cycle time you need to decrease the number of watchers. Below are some best practices you can follow to decrease number of watchers :-

* Remove unnecessary watchers.
* Use one time Angular binding. Especially if you see ng-repeat loop apply one time binding.
* Work in batches.
* Cache DOM
* Use Web worker

## 59) Can we force the digest cycle to run manually?

Yes , you can force it to run manually by calling the “$apply()” method.

## 60) How do we make HTTP get and post calls in Angular?

To make HTTP calls we need to use the “$http” service of Angular. In order to use the http services you need to make provide the “$http” as a input in your function parameters as shown in the below code.

function CustomerController($scope,$http)

{

$scope.Add = function()

{

            $http({ method: "GET", url: "http://localhost:8438/SomeMethod"     }).success(function (data, status, headers, config)

{

*// Here goes code after success*

}

}

}

“$http” service API needs atleast three things:-

* First what is the kind of call “POST” or “GET”.
* Second the resource URL on which the action should happen.
* Third we need to define the “success” function which will be executed once we get the response from the server.

$http({ method: "GET", url: "http://localhost:8438/SomeMethod"    }).success(function (data, status, headers, config)

{

*// Here goes code after success*

}

## 61) How do we pass data using HTTP POST in Angular ?

You need to pass data using the “data” keyword in the “$http” service API function. In the below code you can see we have created a javascript object “myData” with “CustomerName” property. This object is passed in the “$http” function using HTTP POST method.

Var myData = {};

myData.CustomerName = "Test";

$http({ method: "POST",

data: myData,

url: "http://www.xyz.com"})

.success(function (data, status, headers, config)

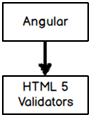
{

*// Here goes code after success*

}

## 62) How are validations implemented in Angular?

Angular leverages HTML 5 validations and new form element types to implement validation.



For instance below is a simple form which has two text boxes. We have used HTML 5 “required” validation attribute and a form element of type “email”.

<form name="frm1" id="frm1" >

Name :- <input type=text name="CustomerName" id="CustomerName" required /> Email :- <input type=email name="Email" id="Email" />

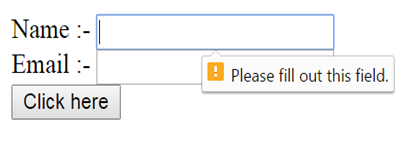
<input type=submit value="Click here"/>

</form>

Below are some example of new form elements introduced in HTML 5 and Angular works with almost all of them :-

* Color.
* Date
* Datetime-local
* Email
* Time
* Url
* Range
* Telephone
* Number
* Search

When you run the above HTML inside a browser which understands HTML 5 , you will see your validations and form types in actions as shown in the below browser screen shot.



Angular leverages HTML 5 validation attributes and new HTML 5 form elements. Now if we want Angular to handle validation we need first stop HTML 5 to do validation. So for that the first step is to specify “novalidate” attribute on the form tag.

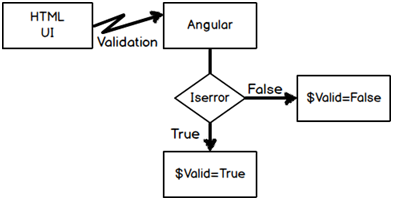
<form name="frm1" novalidate>

-----

</form>

So now the HTML will not fire those validations it will be routed to the Angular engine to further take actions.

In other words when end user fills data in the HTML UI , validation events are routed to Angular framework and depending on scenario Angular sets a field called as “$Valid”. So if the validations are fine it sets it to “True” or else its sets it to “False”.



So you can see in the below code we have attached the angular controller and models to the text boxes. Watch the code of the button it has “ng-disabled” attribute which is set via the “$Valid” property in a NEGATED fashion.

Negated fashion means when there is no error it should enable the button and when there are errors that means it’s false it should disable the button.

<form name="frm1" novalidate>

Name:-<input type=text ng-model="Customer.CustomerName" name="CustomerName" required />

Email :- <input type=email ng-model="Customer.Email" name="Email" />

<input type=submit value="Click here" ng-disabled="!(frm1.$valid)"/>

</form>

***Note*** *:- “Name” is needed for the validations to work.*

## 63) How to check error validation for a specific field?

To check for a specific field you need to use the below DOM code.

!frm1.CustomerName.$valid

## 64) What does SPA (Single page application) mean?

SPA is a concept where rather loading pages from the server by doing post backs we create a single shell page or master page and load the webpages inside that master page.

## 65) How can we implement SPA with Angular?

By using Angular routes.

## 66) How to implement routing in Angular?

Implementing Angular route is a five step process: -

Step 1: - Add the “Angular-route.js” file to your view.

<script src="~/Scripts/angular-route.js"></script>

Step 2: - Inject “ngroute” functionality while creating Angular app object.

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

Step 3: - Configure the route provider.

In route provider we need to define which URL pattern will load which view. For instance in the below code we are saying “Home” loads “Yoursite/Home” view and “Search” loads “YourSite/Search” view.

app.config(['$routeProvider',

function ($routeProvider) {;

$routeProvider.

when('/Home, {

templateUrl: 'Yoursite/Home',

controller: 'HomeController'

}).

when('/Search', {

templateUrl: YourSite/Search',

controller: 'SearchController'

}).

otherwise({

redirectTo: '/'

});

}]);

Step 4: - Define hyperlinks.

Define hyper link with the “#” structure as shown below. So now when user clicks on the below anchor hyperlinks, these actions are forwarded to route provider and router provider loads the view accordingly.

<div>

<a href="#/Home">Home</a><br />

<a href="#/Search"> Search </a><br />

</div>

Step 5: - Define sections where to load the view.

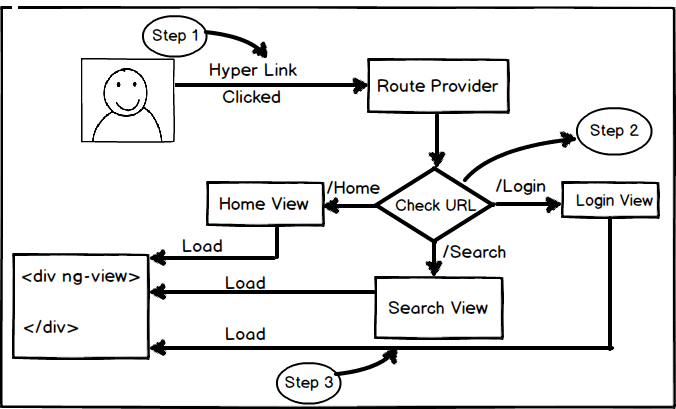
Once the action comes to the router provider it needs a place holder to load views. That’s defined by using the “ng-view” tag on a HTML element. You can see in the below code we have created a “DIV” tag with a place holder. So the view will load in this section.

<div ng-view>

</div>

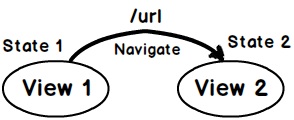
So if we summarize angular routing is a three step process (Below is a visual diagram for the same): -

* Step 1: - End user clicks on a hyperlink or button and generates action.
* Step 2: - This action is routed to the route provider.
* Step 3: - Router provider scans the URL and loads the view in the place holder defined by “ng-view” attribute.



## 67) How to implement SPA using angular-UI route?

Angular UI route helps to implement SPA concept using the concept of STATES. The main goal of SPA is navigating from one view to other view without reloading the main page. Angular UI route visualizes every view as a STATE. When you want to navigate from one view to other view you can either use the STATE names or use URL.



So let’s say we want to navigate from “Home.htm” view to About.htm” view so we can define two states “Home” and “About” and link them to the respective HTML page as shown below.

You can also specify URL by which you can move between these states by using “url” property as shown in the below code.

myApp.config(function ($stateProvider, $urlRouterProvider) {

$stateProvider

.state('Home', {

url: '/HomePage',

templateUrl: 'Home.htm'

})

.state('About', {

url: '/About',

templateUrl: 'About.htm'

})};

Now once the states are defined to we need to use “ui-sref” and if you want to navigate using url provide “url” value in the “href” of the anchor tag.

We also need to provide "<ui-view>" tag to define in which location we want to load the views.

<a ui-sref="About" href="#About">Home</a>

<a href="#Home">About</a>

<ui-view></ui-view>

Below is the complete code if HTML , please ensure you have also referenced of “Angular-UI” js file. You can also see “App.js” file , this file has code which defines the states.

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

<script src="Scripts/angular-ui-router.js" type="text/javascript"></script>

<script src="Scripts/App.js" type="text/javascript"></script>

<body ng-app="myApp">

<a ui-sref="About" href="#About">Home</a>

<a href="#Home">About</a>

<ui-view></ui-view>

</body>

</html>

## 68) Can we load HTML content rather than a full page ?

Yes, you can load simple HTML content by using “template” property as shown in the highlighted code below.

myApp.config(function ($stateProvider, $urlRouterProvider) {

$stateProvider

.state('About', {

url: '/About',

template: '<b>This is About us</b>'

})};

## 69) How can we create controllers and pass parameters in Angular UI route?

To create a controller we need to use “controller” property of the state provider. To specify parameters you can put the parameter name after the url. In the below code you can see ‘Id’ parameter after the url and also you can see how validations are applied on these parameters using regex.

myApp.config(function ($stateProvider, $urlRouterProvider) {

$stateProvider

.state('State1', {

url: '/SomeURL/{Id:[0-9]{4,4}}',

template: '<b>asdsd</b>',

controller: function ($scope, $stateParams) {

alert($stateParams.Id);

}

});

## 70) How to implement nested views using Angular UI route?

First let us understand the concept of nested views. We want to navigate as follows in SPA. From main view we want to navigate to some view and in that view we want to load some other view.

Angular UI Router helps to define nested states. Below is the code of “MainView” in which we have defined one more state “View” and in that we have two child states “View.SubView1” and “View.SubView2” which points to different views.

myApp.config(function ($stateProvider, $urlRouterProvider) {

$stateProvider

.state("View", {

templateUrl: 'View.htm'

})

.state('View.SubView1', {

template: '<b>Sub view 1</b>'

}).state('View.SubView2', {

template: '<b>Sub view 2</b>'

});

});

In the parte view we can now define navigation to child states i.e. “View.SubView1” and “View.SubView2”.

<a ui-sref="View.SubView1" href="#View.SubView1">Sub view 1</a>

<a ui-sref="View.SubView2" href="#View.SubView1 ">Sub view 2</a>

<div ui-view></div>

## 71) How can we create a custom directive in Angular?

Till now we have looked in to predefined Angular directives like “ng-controller”,”ng-model” and so on. But what if we want to create our own custom Angular directive and attach it with HTML elements as shown in the below code.

<div id=footercompany-copy-right></div>

To create a custom directive we need to use the “directive” function to register the directive with angular application. When we call the “register” method of “directive” we need to specify the function which will provide the logic for that directive.

For example in the below code we have created a copy right directive and it returns a copy right text.

*Please note “app” is an angular application object which has been explained in the previous sections.*

app.directive('companyCopyRight', function ()

{

return

{

        template: '@CopyRight questpond.com '

};

});

The above custom directive can be later used in elements as shown in below code.

<div ng-controller="CustomerViewModel">

<div company-copy-right></div>

</div>

## 72) Can I set an Angular directive template to a HTML web page?

Yes, you can set template to page directly by using “templateUrl” property of the directive as shown in the code snippet below.

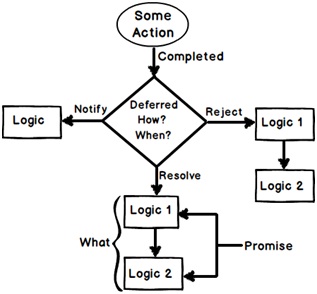
directive.templateUrl = "/templates/footer.html";

## 73) Explain $q service, deferred and promises?

Promises are POST PROCESSING LOGICS which you want to execute after some operation / action is completed. While deferred helps to control how and when those promise logics will execute.

We can think about promises as “WHAT” we want to fire after an operation is completed while deferred controls “WHEN” and “HOW” those promises will execute.

For example after an operation is complete you want to a send a mail, log in to log file and so on. So these operations you will define using promise. And these promise logics will be controlled by deferred.



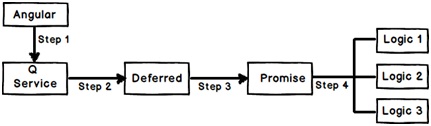
We are thankful to [www.stepbystepschools.net](http://www.stepbystepschools.net/) for the above image.

So once some action completes deferred gives a signal “Resolve”, “Reject” or “Notify” and depending on what kind of signal is sent the appropriate promise logic chain fires.

“$q” is the angular service which provides promises and deferred functionality.

Using promises, deferred and “q” service is a 4 step process:-

* Step 1:- Get the “q” service injected from Angular.
* Step 2 :- Get deferred object from “q” service object.
* Step 3 :- Get Promise object from deferred object.
* Step 4 :- Add logics to the promise object.



Below is the angular code for the above 4 steps.

*// Step 1 :- Get the "q" service*

function SomeClass($scope,$q) {

*// Step 2 :- get deferred from "q" service*

var defer = $q.defer();

*// step 3:- get promise from defer*

var promise = defer.promise;

*// step 4 :- add success and failure logics to promise object*

promise.then(function () {

alert("Logic1 success");

}, function () {

alert("Logic 1 failure");

});

promise.then(function () {

alert("Logic 2 success");

}, function () {

alert("Logic 2 failure");

});

}

So now depending on situations you can signal your promise logics via deferred to either fire the success events or the failure events.

*// This will execute success logics of promise*

defer.resolve();

*// This will execute failure logics of promise*

defer.reject();

**74) What is routing in AngularJs?**

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

**75) What is deep linking in AngularJS?**Deep linking allows you to encode the state of application in the URL so that it can be bookmarked.   
The application can then be restored from the URL to the same state.

**76) Explain AngularJS boot process.** Whenthe page is loaded in the browser, following things happen:   
1. HTML document is loaded into the browser, and evaluated the browser.   
2. AngularJS JavaScript file is loaded: the angular global object is created. Next, JavaScript which registers controller functions is executed.   
3. Next AngularJS scans through the HTML to look for AngularJS apps are views.   
4. Once view is located, it connects that view to the corresponding controller function.

5. Next, AngularJS executes the controller functions.

6. It then renders the views with data from the model populated by the controller.

7. The page gets ready.

OR

When ever we load any Angular is application in the browser the following sequence of steps that happen:   
**step 1:** Just like as any request the HTML document will be loaded will be loaded and evaluated In the browser. After that the Angular Js file which was attached in the application gets loaded.   
**step 2:** Once that JS is being called the Global object is created after that the java script which registers the controller functions are executed.   
**step 3:** After that Angular is will look for Angular Js Apps and Views. Once that view is being traced out it connects that view into the corresponding controller function   
**step 4:** Now Angular is will execute the controller functions. It will render the views with data from mnodel which is populated by the controller.   
**step 5:** Now the page is ready for mhe access.   
This is how the Angular is will control the flow from initial step to the final stage of page load functionality.

**77) What is MVC?**   
Model View Controller 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.

**78) What is MVC Architecture in AngularJS?**

In AngularJS, scope objects are treated as **M**odel. The **V**iew is display of model that is your data. And the model gets initialized within a JavaScript constructor function, called**C**ontroller in AngularJS. Let take a look at below code to understand it better.

<!DOCTYPE html>

<html>

<head>

<script data-require="angular.js@\*" data-semver="1.3.6" src="https://code.angularjs.org/1.3.6/angular.js"></script>

<link rel="stylesheet" href="style.css" />

<script>

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

myApp.controller('MyController', ['$scope',

function($scope) {

$scope.website = 'jquerybyexample.net';

}

]);

</script>

</head>

<body ng-app="myApp">

<div ng-controller="MyController">

<h1>My website address is {{ website }}</h1>;

</div>

</body>

</html>

Here MyController is a Controller and $scope (Model) is populated within Controller. And later on in div element $scope object data is displayed (View).

**79) 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.

**80) Explain ng-app directive.**   
ng-app directive defines and links an AngularJS application to HTML. It also indicates the start of the application.

**81) 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.

**82) Explain ng-controller directive.**   
ng-controller directive tells AngularJS what controller to use with this view.   
AngularJS application mainly relies on controllers to control the flow of data in the application.   
A controller is a JavaScript object containing attributes/properties and functions.   
Each controller accepts $scope as a parameter which refers to the application/module that controller is to control.

**83) How AngularJS integrates with HTML?**   
AngularJS being a pure javaScript based library integrates easily with HTML.   
- Step 1 - Include angularjs javascript library in the html page.   
- 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.

**84) 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.

OR

Angular expressions are JavaScript-like code snippets that are usually placed in bindings such as {{ expression }}. For example, these are valid expressions in Angular:

* {{ 3+4 }}
* {{ a+b }}
* {{ user.name }}
* {{ items[index] }}

**85) How do you make Ajax call using AngularJS?**   
AnguIarJS 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.

**86) Is Angular JS extensible?**   
Yes! In AngularJS we can create custom directive to extend AngularJS existing functionalities.   
Custom directive 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.

**87) What is internationalization?**

Internationalization is a way to show locate specific information on a website.

For example, display content of a website in English language in United States and in Danish in France.

**88) How to implement internationalization in Angular JS?**   
AngularJS 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>

**89) Explain ng-init directive.**   
ng-init directive initializes an AngularJS Application data. It is used to put values to the variables to be used in the application.

**90) Explain ng-repeat directive.**   
ng-repeat directive repeats html elements for each item in a collection.

**91) Explain lowercase filter.**   
Lowercase filter converts a text to lower case text.

**92) Explain uppercase filter.**

Uppercase filter converts a text to upper case text.

**93) Explain currency filter.**   
Currency filter formats text in a currency format.

**94) Explain orderby filter.**orderby filter orders the array based on provided criteria.

**95) Explain ng-disabled directive.**   
ng-disabled directive disables a given control.

**96) Which of the following is true about ng-disabled directive?**

ng-disabled directive disables/enables a given control.

**97) Explain ng-show directive.**   
ng-show directive shows a given control.

**98) Explain ng-hide directive.**   
ng-hide directive hides a given control.

**99) Explain ng-click directive.**

ng-cick directive represents a AngularJS click event.

**100) 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);

}

});

**101) Which components can be injected as a dependency in AngularJS?**AngularJS provides a supreme Dependency Injection mechanism.   
It provides following core components which can be injected into each other as dependencies.   
- value   
- factory   
- service

- provider  
- constant

**102) How angular.module works?**

Angular.module is used to create AngularJS modules along with its dependent modules.

**103) How to validate data in AngularJS?**   
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.

**104) What is use of $routeProvider in AngularJS?**

$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.

**105) What is scope hierarchy in AngularJS?**

Scopes are controller specific.

If we define nested controllers then child controller will inherit the scope of its parent controller.

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

**107) What is provider?**Provider is used by AngularJS internally to create services, factory etc. during config phase(phase during which AngularJS bootstraps itself).   
Below mention script can be used to create MathService that we’ve created earlier.   
Provider is a special factory method with a method get() which is used to return the value/service/factory.   
   
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;

};

});

});

**108) What is constant?**Constants are used to pass values at config phase considering the fact that value cannot be used to be passed during config phase.

mainApp.constant(“configParam”, “constant value”);

**109) How to retrieve the scope of a DOM element in AngularJS?**

To retrieve the scope of a DOM element in AngularJS, we can use following code snippet.

var scope = angular.element(“elementSelector”).scope()

**110) What is jqLite or jQuery lite?**

jQLite is a subset of jQuery that is built directly into AngularJS. jQLite provides you all the useful features of jQuery. In fact it provides you limited features or functions of JQuery.

OR

jqLite is AngularJS built-in subset of jQuery that exposes almost 35 most used functions of jQuery without referencing the jQuery file in the web page.

OR

jqLite is a tiny, API-compatible subset of jQuery that allows AngularJS to manipulate the DOM in a cross-browser compatible way. jqLite implements only the most commonly needed functionality with the goal of having a very small footprint.

**111) How AngularJS is different from other JavaScript Framework?**AngularJS is different from other JavaScript framework in following ways:   
1. AngularJS mark-up lives in the DOM.   
2. AngularJS uses plain old JavaScript objects (POJO).   
3. AngulariS is leverages with Dependency Injection.

**112) How to use jQuery with AngularJs?**By default AngulariS use jQLlte which is the subset of jQuery. If you want to use jQuery then simply load the jQuery library before loading the AngularJS. By doing so, Angular will skip jQLite and will started to use jQuery library.

**113) Is AngularJS a library, framework, plugin or a browser extension?**AngularJS is a first class JavaScript framework which allows you to build well structured, easily testable, and maintainable front-end applications. It is not a library since library provides you limited functionality or has dependencies to other libraries. It is not a plugin or browser extension since it is based on JavaScript and compatible with both desktop and mobile browsers.

OR

AngularJS fits the definition of a framework the best, even though it's much more lightweight than a typical framework and that's why many confuse it with a library.

AngularJS is 100% JavaScript, 100% client side and compatible with both desktop and mobile browsers. So it's definitely not a plugin or some other native browser extension.

**114) What is the size of angular.js file?**The size of the compressed and minified file is < 36KB.

**115) How AngularJs handle the security?**AngulariJS provide following built-in protection from basic security holes:   
1. Prevent HTML injection attacks.   
2. Prevent Cross-site-Scripting (CSS) attacks.   
3. Prevent XSRF protection for server side communication.

4. Basic authentication and authorization etc.

**116) What is Angular Prefixes $ and $$?**To prevent accidental name collisions with your code, Angular prefixes names of public objects with $ and names at private objects with $$. So, do not use the s or $$ prefix in your code.

**117) How to loop elements in Angular Js**   
Below is the procedure to loop through data in Angular JS

<div data-ng-app= “ ” data-ng-init="names=[’DotnetFunda’, ‘TechFunda', ‘SQLFunda', ‘ITFunda’]”>

<b>Loop Example:</b>   
<br />   
<ul>   
<li data-ng-repeat= “x in names”>   
{{ x })   
</1i>   
</ul>   
</div>

**118) How to create an object in Angular JS**   
Below is the way to create an object in Angular JS   
<div ng-app= “ ” ng-init="DnfObj={Str1:‘Dotnet ‘,Str2:‘ Funda’}”>

String Display: <b>{{ Dnfobj.Str2 }}</b></div>

**119) What is the meaning of Directive in Angular Js? List some of them.**As the name suggests Directives are the ones which shows the direction on how the DOM elements to behave.   
Directives are the markers on the DOM element just like an attribute like element name or Css Class which tells the Angular Js HTML compiler to add a specified behavior to the respective DOM element via event listeners which are available.   
Some of the Directives are:   
ng-app : It Initializes application.   
ng-model : It Binds the HTML controls to application data.   
ng-Controller : It Attaches a controller class to view.   
ng-repeat-: It Binds repeated HTML data elements. This is more or less like a for Loop   
ng-if: It Binds HTML elements with a condition.   
ng-show: It used to show the HTML elements.   
ng-hide: It used to hide the HTML elements.   
ng-class : It used to assign CSS class.   
ng-src : It used to pass the URL image.

**120) Mention some angularJS directives and their purpose?**

The beauty of AngularJS directives is that they are self explainatory. By just looking at directive name, you will get the idea about purpose and use of directive. Below are some mostly used directives.  
**ng-app** : Initializes application.  
**ng-model** : Binds HTML controls to application data.  
**ng-Controller** : Attaches a controller class to view.  
**ng-repeat** : Bind repeated data HTML elements. Its like a for loop.  
**ng-if** : Bind HTML elements with condition.  
**ng-show** : Used to show the HTML elements.  
**ng-hide** : Used to hide the HTML elements.  
**ng-class** : Used to assign CSS class.  
**ng-src** : Used to pass the URL image etc.  
Event Listeners  
**ng-click** : Click event to bind on HTML elements.  
**ng-dbl-click**  
Mouse event listeners  
**ng-mousedown  
ng-mouseup  
ng-mouseenter  
ng-mouseleave  
ng-mousemove  
ng-mouseover**  
Keyboard event listeners  
**ng-keydown  
ng-keyup  
ng-keypress  
ng-change**

It's a long list. You can find about all the directives **https://docs.angularjs.org/api/ng/directive**

**121) Can we have nested Controllers in Angular Js?**   
Yes we can have nested controllers. The thing is it works in a hierarchical way while using a View.   
Example:   
GrandFather --- Father --- Son   
<div ng-controller=”GrandFather”>   
<p>{{message}} {{name}}!</p>   
<div ng-controller="Father”>   
<p>Hello {{name}}!</p>   
<div ng-controller=”Son”>  
<p>{{message}} {{narne}}! Your username is {{username}}.</p>   
</div>   
</div>   
</div>

OR

YES. We can create nested controllers in AngularJS. Nested controller are defined in hierarchical manner while using in View. Take a look at below code. Here the hierarchy is "MainCtrl -> SubCtrl -> SubCtrl1".

<div ng-controller="MainCtrl">

<p>{{message}} {{name}}!</p>

<div ng-controller="SubCtrl">

<p>Hello {{name}}!</p>

<div ng-controller="SubCtrl2">

<p>{{message}} {{name}}! Your username is {{username}}.</p>

</div>

</div>

</div>

**122) How can we restrict Directives to be invoked only by specific methods?**We can restrict the Directive by using a Property “restrict” with value “A" so it will be invoked only by attributes.   
var app = angular.module(“myApp”, []);  
app.directive(“DNFTestDirective”, function() {   
return {   
restrict : “A”,  
template : “<h1>DotnetFunda is the best for online Dotnet Support</h1>”

};

});

**123)** **Is AngularJS a framework, library or a plugin?**

The suitable answer is that its a framework. As its lightweight so people also get confused between library and framework. AngularJS is open source client side MVC framework for creating dynamic web applications.

**124)** **Is it same as jQuery?**  
No. jQuery is great library for manipulating the DOM, providing better user experience with animations and effects. You can create website using jQuery but not a web application. jQuery is just a library to play around with HTML, where as AngularJS is a framework to build a dynamic web app as it supports two data binding, MVC, allow testability, templates and many more. Its like AngularJS like a toolbox and jQuery is just a tool.

**125) Does Angular use the jQuery library?**

Yes, Angular can use jQuery if it's present in your app when the application is being bootstrapped. If jQuery is not present in your script path, Angular falls back to its own implementation of the subset of jQuery that we call jQLite.

**126) Is AngularJS is compatible with all modern browsers?**

Yes, AngularJS team run extensive test suite against the following browsers: Safari, Chrome, Firefox, Opera 15, IE9 and mobile browsers (Android, Chrome Mobile, iOS Safari).

**127) What is the basic need to start with AngularJS?**

To start with AngularJS, one need to make reference of angular.js. The latest version of AngularJS can be downloaded from [**AngularJS.com**](https://angularjs.org/). You can either download the required js file and then host them locally or you can also use google CDN for referencing it. Here is the [**link**](https://developers.google.com/speed/libraries/devguide#angularjs) for google CDN url for referencing AngularJS.

**128) What is the bootstrapping in AngularJS?**

Bootstrapping in AngularJS is nothing but just initializing, or starting, your Angular app. AngularJS supports automatic bootstrapping as well as manual way as well.

**129) Can we create our own directives?**

YES, AngularJS allows us to create our own custom directive.

**130) What are different type or classification of directives?**

AngularJS directives can be classified in 4 different types .

• Element directives

<ng-directive></ng-directive>

• Attribute directives

<span ng-directive></span>

• CSS class directives

<span class="ng-directive: expression;"></span>

• Comment directives

<!-- directive: ng-directive expression -->

**131) What is the name of directive is used to bootstrap an angular app?**

ng-app directive is used to auto-bootstrap an AngularJS application. The ng-app directive defines the root element of the application and is typically present in the root element of the page - e.g. on the <body> or <html> tags.

**132)** **Can AngularJS have multiple ng-app directives in a single page?**

The answer is NO. Only one AngularJS application can be auto-bootstrapped per HTML document. The first ngApp found in the document will be used to define the root element to auto-bootstrap as an application. If you have placed another ng-app directive then it will not be processed by AngularJS. You need to manually bootstrap the second app, instead of using second ng-app directive.

**133)** **Can angular applications (ng-app) be nested within each other?**

NO. AngularJS applications cannot be nested within each other.

**134)** **Can you bootstrap multiple angular applications on same element?**

**NO.** If you try to do that then it will show an error "**App Already Bootstrapped with this Element**". This usually happens when you accidentally use both ng-app andangular.bootstrap to bootstrap an application. You can also get this error if you accidentally load AngularJS itself more than once.

**135)** **In how many different ways, you can define a directive and what is the best practice?**

Angular generally prefers camelCase for directives. But since HTML is not case-sensitive so it refers to directive in DOM in lower case form, delimited by dash (eg. ng-app). But when Angular complies then it normalize the directives.  
  
Below are example of valid directive declaration.

* ng-model
* ngModel
* ng:model
* ng\_model
* data-ng-model
* x-ng-model

The normalization process is as follows:  
1. Strip x- and data- from the front of the element/attributes.  
2. Convert the :, -, or \_-delimited name to camelCase.  
  
The best practice to use dash-delimited (ng-model) or directly camelCase form (ngModel). If you are using HTML validation tool, then it is advised to use data- prefixed version. And it also answers another question which is "**Difference between ng-\* and data-ng-\***".

**136) What is compilation process in Angular?**

Once you have the markup, the AngularJS needs to attach the functionality. This process is called "**compilation**" in Angular. Compiling includes rendering of markup, replacing directives, attaching events to directives and creating a scope. The AngularJS has compiler service which traverses the DOM looking for attributes. The compilation process happens in two phases.

* **Compilation :** traverse the DOM and collect all of the directives and creation of the linking function.
* **Linking:** combine the directives with a scope and produce a live view. The linking function allows for the attaching of events and handling of scope. Any changes in the scope model are reflected in the view, and any user interactions with the view are reflected in the scope model.

When you create a new directive, you can write compile and/or linking functions for it to attach your custom behavior.

**137)** **In case of nested controllers, does the $scope object shared across all controllers?**

**YES**. The $scope object is shared across all controllers and it happens due to scope inheritance. Since the ng-controller directive creates a new child scope, we get a hierarchy of scopes that inherit from each other. So if we define a property on a parent scope, the child scope can manipulate the property. And if the property is not present in child scope, then parent scope property's value is used. Lets consider, the previous question HTML where there are 3 controllers. And here is the AngularJS code to define these controllers.

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

myApp.controller('MainCtrl', ['$scope',

function($scope) {

$scope.message = 'Welcome';

$scope.name = 'Jon';

}

]);

myApp.controller('SubCtrl', ['$scope',

function($scope) {

$scope.name = 'Adams';

}

]);

myApp.controller('SubCtrl2', ['$scope',

function($scope) {

$scope.name = 'Ema';

$scope.username = 'ema123';

}

]);

You will see that the controller "SubCtrl2" doesn't have "message" property define but it is used in HTML. So in this case, the immediate parent scope property will be used. But immediate parent scope which is "SubCtrl" in this case, also doesn't have "message" property. So it again goes one level up and finds the property is present so it uses parent scope value for "message" property and displays it.

**138) Can you please explain what is testability like in Angular?**

Very testable and designed this way from ground up. It has an integrated dependency injection framework, provides mocks for many heavy dependencies (server-side communication).

**139) Why is this project called "AngularJS"? Why is the namespace called "ng"?**

Because HTML has Angular brackets and "ng" sounds like "Angular".

**140) Tell me can we use the open-source Closure Library with Angular?**

Yes, you can use widgets from the Closure Library in Angular.

**141) Do you know what is Angulars performance like?**

The startup time heavily depends on your network connection, state of the cache, browser used and available hardware, but typically we measure bootstrap time in tens or hundreds of milliseconds.

The runtime performance will vary depending on the number and complexity of bindings on the page as well as the speed of your backend (for apps that fetch data from the backend). Just for an illustration we typically build snappy apps with hundreds or thousands of active bindings.

**142) How will you display inprogress revolving image to indicate that RESTful data is bing loaded?**

<div ng-show="loading">

<img class="loading" src="portal/images/loading\_32.gif" />

</div>

$scope.loadReportData = function($http) {

$scope.loading = true; // start spinng the image

$http(

{

method : 'GET',

url : propertiesService.get('restPath')

+ '/myapp/portfolio/'

+ $scope.clientId

+ '/'

+ dateService

.userToRest($scope.reportDate),

cacheBreaker : true

}).success(

function(data, config) {

$scope.reportData = data;

portal.log('reportData: ',

$scope.reportData);

$scope.loading = false; // stop spinning the image

}).error(

function(data, status, headers, config) {

if(data.errorMsg != null) {

$scope.httpError = data.errorMsg;

}

else {

$scope.httpError = "Error retrieving data from " + errorService

.getApacheErrorTitleMessage(status,

data, config);

}

$scope.loading = false; // stop spinning the image

});

};

**143) Why to choose Angular JS Javascript Framework for front-end web development? (v.v important)**

AngularJS is quickly becoming the dominant JavaScript framework for professional web development. With the growth and strength of HTML5 and the increasing performance in modern browsers, many JavaScript frameworks have been created to help develop rich client applications. These frameworks/libraries have given developers a huge toolkit to build enterprise complexity into client-side applications. Server side frameworks are becoming a thing of the past and being replaced with applications written in Backbone, Ember, AngularJS, Knockout, etc.

So why am I talking about AngularJS over frameworks/libraries like Backbone, Ember, or Knockout?

For me, the major points of separation in AngularJS’s favor are the following:

1. Good documentation

2. Write less code to do more

3. Backed by Google

4. Good developer community

5. Simple Data-Binding

6. Small footprint

If you’re looking for a robust, well-maintained framework for any sized project, I strongly recommend that you take a look at AngularJS. It can be downloaded for free at AngularJS.org, which also contains a wealth of information, including the full API documentation, as well as numerous examples and tutorials that cover every facet of front-end web development. Following are some reasons why to choose Angular JS Javascript Framework for front-end web development?

1. Angular JS Framework is developed by Google

Angular is built and maintained by dedicated Google engineers. This one may seem obvious, but it’s important to remember that many (not all) frameworks are made by hobbyists in the open source community. While passion and drive have forged frameworks, like Cappucino and Knockout, Angular is built and maintained by dedicated (and highly talented) Google engineers. This means you not only have a large open community to learn from, but you also have skilled, highly-available engineers tasked to help you get your Angular questions answered.

This isn’t Google’s first attempt at a JavaScript framework; they first developed their comprehensive Web Toolkit, which compiles Java down to JavaScript, and was used by the Google Wave team extensively. With the rise of HTML5, CSS3, and JavaScript, as both a front-end and back-end language, Google realized that the web was not meant to be written purely in Java.

AngularJS came about to standardize web application structure and provide a future template for how client-side apps should be developed.

Angular JS is being used by a host of applications, ranging from hobby to commercial products. Adoption of AngularJS as a viable framework for client-side development is quickly becoming known to the entire web development community.

Because AngularJS is built by Google, you can be sure that you’re dealing with efficient and reliable code that will scale with your project. If you’re looking for a framework with a solid foundation, Angular is your choice!

2. Angular JS is equipped with a lot of features

If you’re familiar with projects, **like QUnit, Mocha or Jasmine, then you’ll have no trouble learning Angular’s unit-testing API.**

Angular, similar to Backbone or JavaScriptMVC, is a complete solution for rapid front-end development. No other plugins or frameworks are necessary to build a data-driven web application. Here’s an overview of Angular’s stand-out features:

A) REST Easy. RESTful actions are quickly becoming the standard for communicating from the server to the client. In one line of JavaScript, you can quickly talk to the server and get the data you need to interact with your web pages. AngularJS turns this into a simple JavaScript object, as Models, following the MVVM (Model View View-Model) pattern.

B) MVVM to the Rescue! Models talk to ViewModel objects (through something called the $scope object), which listen for changes to the Models. These can then be delivered and rendered by the Views, which is the HTML that expresses your code. Views can be routed using the $routeProvider object, so you can deep-link and organize your Views and Controllers, turning them into navigable URLs. AngularJS also provides stateless controllers, which initialize and control the $scope object.

C) Data Binding and Dependency Injection. Everything in the MVVM pattern is communicated automatically across the UI whenever anything changes. This eliminates the need for wrappers, getters/setters or class declarations. AngularJS handles all of this, so you can express your data as simply as with JavaScript primitives, like arrays, or as complex as you wish, through custom types. Since everything happens automatically, you can ask for your dependencies as parameters in AngularJS service functions, rather than one giant main() call to execute your code.

D) Extends HTML. Most websites built today are a giant series of <div> tags with little semantic clarity. You need to create extensive and exhaustive CSS classes to express the intention of each object in the DOM. With Angular, you can operate your HTML like XML, giving you endless possibilities for tags and attributes. Angular accomplishes this, via its HTML compiler and the use of directives to trigger behaviors based on the newly-created syntax you write.

E) Makes HTML your Template. If you’re used to Mustache or Hogan.js, then you can quckly grasp the bracket syntax of Angular’s templating engine, because it’s just HTML. Angular traverses the DOM for these templates, which house the directives mentioned above. The templates are then passed to the AngularJS compiler as DOM elements, which can be extended, executed or reused. This is key, because, now, you have raw DOM components, rather than strings, allowing for direct manipulation and extension of the DOM tree.

F) Enterprise-level Testing. As stated above, AngularJS requires no additional frameworks or plugins, including testing. If you’re familiar with projects, like QUnit, Mocha or Jasmine, then you’ll have no trouble learning Angular’s unit-testing API and Scenario Runner, which guides you through executing your tests in as close to the actual state of your production application as possible.

These are the fundamental principles that guide AngularJS to creating an efficient, performance-driven, and maintainable front-end codebase. As long as you have a source for storing data, AngularJS can do all of the heavy lifting on the client, while providing a rich, fast experience for the end user.

3. You can learn Angular JS easily

Getting started with AngularJS is incredibly easy. With a few attributes added to your HTML, you can have a simple Angular app up in under 5 minutes!

1. Add the ng-app directive to the <html> tag so Angular knows to run on the page:

<html lang="en" ng-app>

2. Add the Angular <script> tag to the end of your <head> tag:

<head>

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

...

</head>

3. Add regular HTML. AngularJS directives are accessed through HTML attributes, while expressions are evaluated with double-bracket notation:

<body ng-controller="ActivitiesListCtrl">

<h1>Today's activities</h1>

<ul>

<li ng-repeat="activity in activities">

{{activity.name}}

</li>

</ul>

</body>

</html>

**144) Is AngularJS a templating system?**

At the highest level, Angular does look like a just another templating system. But there is one important reason why the Angular templating system is different, that makes it very good fit for application development: bidirectional data binding. The template is compiled in the browser and the compilation step produces a live view. This means you, the developers, don't need to write code to constantly sync the view with the model and the model with the view as in other templating systems.

OR

We can use the **ng-include**directive for templating system. Angularjs support DOM based templating.

**145) Do I need to worry about security holes in AngularJS?**

Like any other technology, AngularJS is not impervious to attack. Angular does, however, provide built-in protection from basic security holes including cross-site scripting and HTML injection attacks. AngularJS does round-trip escaping on all strings for you and even offers XSRF protection for server-side communication.

AngularJS was designed to be compatible with other security measures like Content Security Policy (CSP), HTTPS (SSL/TLS) and server-side authentication and authorization that greatly reduce the possible attack vectors and we highly recommended their use.

**146) What are the key differences between AngularJS and jQuery?**

AngularJS and jQuery are the Javascript frameworks and are different with each other, so never mix up the AngularJS and jQuery code in your project. Use only one Javascript framework at a time. If you are starting a new project, must consider AngularJS over jQuery. If you are a experienced jQuery developer, then you have to invest some time to work in AngularJS way. There are a lot of difference between AngularJS and jQuery.

**1. Web designing approach in jQuery and AngularJS**

In jQuery, you design a page, and then you make it dynamic. This is because jQuery was designed for augmentation and has grown incredibly from that simple premise.

But in AngularJS, you must start from the ground up with your architecture in mind. Instead of starting by thinking "I have this piece of the DOM and I want to make it do X", you have to start with what you want to accomplish, then go about designing your application, and then finally go about designing your view.

**2. Don't augment jQuery with AngularJS**

Similarly, don't start with the idea that jQuery does X, Y, and Z, so I'll just add AngularJS on top of that for models and controllers. This is really tempting when you're just starting out, which is why I always recommend that new AngularJS developers don't use jQuery at all, at least until they get used to doing things the "Angular Way".

I've seen many developers here and on the mailing list create these elaborate solutions with jQuery plugins of 150 or 200 lines of code that they then glue into AngularJS with a collection of callbacks and $applys that are confusing and convoluted; but they eventually get it working! The problem is that in most cases that jQuery plugin could be rewritten in AngularJS in a fraction of the code, where suddenly everything becomes comprehensible and straightforward.

The bottom line is this: when solutioning, first "think in AngularJS"; if you can't think of a solution, ask the community; if after all of that there is no easy solution, then feel free to reach for the jQuery. But don't let jQuery become a crutch or you'll never master AngularJS.

**3. Always think in terms of architecture**

First know that single-page applications are applications. They're not webpages. So we need to think like a server-side developer in addition to thinking like a client-side developer. We have to think about how to divide our application into individual, extensible, testable components.

So then how do you do that? How do you "think in AngularJS"? Here are some general principles, contrasted with jQuery.

The view is the "official record"

In jQuery, we programmatically change the view. We could have a dropdown menu defined as a ul like so:

<ul class="main-menu">

    <li class="active">

        <a href="#/home">Home</a>

    </li>

    <li>

        <a href="#/menu1">Menu 1</a>

        <ul>

            <li><a href="#/sm1">Submenu 1</a></li>

            <li><a href="#/sm2">Submenu 2</a></li>

            <li><a href="#/sm3">Submenu 3</a></li>

        </ul>

    </li>

    <li>

        <a href="#/home">Menu 2</a>

    </li>

</ul>

In jQuery, in our application logic, we would activate it with something like:

$('.main-menu').dropdownMenu();

When we just look at the view, it's not immediately obvious that there is any functionality here. For small applications, that's fine. But for non-trivial applications, things quickly get confusing and hard to maintain.

In AngularJS, though, the view is the official record of view-based functionality. Our ul declaration would look like this instead:

<ul class="main-menu" dropdown-menu>

    ...

</ul>

These two do the same thing, but in the AngularJS version anyone looking at the template knows what's supposed to happen. Whenever a new member of the development team comes on board, he can look at this and then know that there is a directive called dropdownMenu operating on it; he doesn't need to intuit the right answer or sift through any code. The view told us what was supposed to happen. Much cleaner.

Developers new to AngularJS often ask a question like: how do I find all links of a specific kind and add a directive onto them. The developer is always flabbergasted when we reply: you don't. But the reason you don't do that is that this is like half-jQuery, half-AngularJS, and no good. The problem here is that the developer is trying to "do jQuery" in the context of AngularJS. That's never going to work well. The view is the official record. Outside of a directive (more on this below), you never, ever, never change the DOM. And directives are applied in the view, so intent is clear.

Remember: don't design, and then mark up. You must architect, and then design.

Data binding

This is by far one of the most awesome features of AngularJS and cuts out a lot of the need to do the kinds of DOM manipulations I mentioned in the previous section. AngularJS will automatically update your view so you don't have to! In jQuery, we respond to events and then update content. Something like:

$.ajax({

  url: '/myEndpoint.json',

  success: function ( data, status ) {

    $('ul#log').append('<li>Data Received!</li>');

  }

});

For a view that looks like this:

<ul class="messages" id="log">

</ul>

Apart from mixing concerns, we also have the same problems of signifying intent that I mentioned before. But more importantly, we had to manually reference and update a DOM node. And if we want to delete a log entry, we have to code against the DOM for that too. How do we test the logic apart from the DOM? And what if we want to change the presentation?

This a little messy and a trifle frail. But in AngularJS, we can do this:

$http( '/myEndpoint.json' ).then( function ( response ) {

    $scope.log.push( { msg: 'Data Received!' } );

});

And our view can look like this:

<ul class="messages">

    <li ng-repeat="entry in log">{{ entry.msg }}</li>

</ul>

But for that matter, our view could look like this:

<div class="messages">

    <div class="alert" ng-repeat="entry in log">

        {{ entry.msg }}

    </div>

</div>

And now instead of using an unordered list, we're using Bootstrap alert boxes. And we never had to change the controller code! But more importantly, no matter where or how the log gets updated, the view will change too. Automatically. Neat!

Though I didn't show it here, the data binding is two-way. So those log messages could also be editable in the view just by doing this:

<input ng-model="entry.msg" />.

And there was much rejoicing.

Distinct model layer

In jQuery, the DOM is kind of like the model. But in AngularJS, we have a separate model layer that we can manage in any way we want, completely independently from the view. This helps for the above data binding, maintains separation of concerns, and introduces far greater testability. Other answers mentioned this point, so I'll just leave it at that.

Separation of concerns

And all of the above tie into this over-arching theme: keep your concerns separate. Your view acts as the official record of what is supposed to happen (for the most part); your model represents your data; you have a service layer to perform reusable tasks; you do DOM manipulation and augment your view with directives; and you glue it all together with controllers. This was also mentioned in other answers, and the only thing I would add pertains to testability, which I discuss in another section below.

Dependency injection

To help us out with separation of concerns is dependency injection (DI). If you come from a server-side language (from Java to PHP) you're probably familiar with this concept already, but if you're a client-side guy coming from jQuery, this concept can seem anything from silly to superfluous to hipster. But it's not.

From a broad perspective, DI means that you can declare components very freely and then from any other component, just ask for an instance of it and it will be granted. You don't have to know about loading order, or file locations, or anything like that. The power may not immediately be visible, but I'll provide just one (common) example: testing.

Let's say in our application, we require a service that implements server-side storage through a REST API and, depending on application state, local storage as well. When running tests on our controllers, we don't want to have to communicate with the server - we're testing the controller, after all. We can just add a mock service of the same name as our original component, and the injector will ensure that our controller gets the fake one automatically - our controller doesn't and needn't know the difference.

**4. Test-driven development**

This is really part of section 3 on architecture, but it's so important that I'm putting it as its own top-level section.

Out of all of the many jQuery plugins you've seen, used, or written, how many of them had an accompanying test suite? Not very many because jQuery isn't very amenable to that. But AngularJS is.

In jQuery, the only way to test is often to create the component independently with a sample/demo page against which our tests can perform DOM manipulation. So then we have to develop a component separately and then integrate it into our application. How inconvenient! So much of the time, when developing with jQuery, we opt for iterative instead of test-driven development. And who could blame us?

But because we have separation of concerns, we can do test-driven development iteratively in AngularJS! For example, let's say we want a super-simple directive to indicate in our menu what our current route is. We can declare what we want in our view:

<a href="/hello" when-active>Hello</a>

Okay, now we can write a test:

it( 'should add "active" when the route changes', inject(function() {

    var elm = $compile( '<a href="/hello" when-active>Hello</a>' )( $scope );

    $location.path('/not-matching');

    expect( elm.hasClass('active') ).toBeFalsey();

    $location.path( '/hello' );

    expect( elm.hasClass('active') ).toBeTruthy();

}));

We run our test and confirm that it fails. So now we can write our directive:

.directive( 'whenActive', function ( $location ) {

    return {

        scope: true,

        link: function ( scope, element, attrs ) {

            scope.$on( '$routeChangeSuccess', function () {

                if ( $location.path() == element.attr( 'href' ) ) {

                    element.addClass( 'active' );

                }

                else {

                    element.removeClass( 'active' );

                }

            });

        }

    };

});

Our test now passes and our menu performs as requested. Our development is both iterative and test-driven.

**5. Conceptually, directives are not packaged jQuery**

You'll often hear "only do DOM manipulation in a directive". This is a necessity. Treat it with due deference!

But let's dive a little deeper...

Some directives just decorate what's already in the view (think ngClass) and therefore sometimes do DOM manipulation straight away and then are basically done. But if a directive is like a "widget" and has a template, it should also respect separation of concerns. That is, the template too should remain largely independent from its implementation in the link and controller functions.

AngularJS comes with an entire set of tools to make this very easy; with ngClass we can dynamically update the class; ngBind allows two-way data binding; ngShow and ngHide programmatically show or hide an element; and many more - including the ones we write ourselves. In other words, we can do all kinds of awesomeness without DOM manipulation. The less DOM manipulation, the easier directives are to test, the easier they are to style, the easier they are to change in the future, and the more re-usable and distributable they are.

I see lots of developers new to AngularJS using directives as the place to throw a bunch of jQuery. In other words, they think "since I can't do DOM manipulation in the controller, I'll take that code put it in a directive". While that certainly is much better, it's often still wrong.

Think of the logger we programmed in section 3. Even if we put that in a directive, we still want to do it the "Angular Way". It still doesn't take any DOM manipulation! There are lots of times when DOM manipulation is necessary, but it's a lot rarer than you think! Before doing DOM manipulation anywhere in your application, ask yourself if you really need to. There might be a better way.

Here's a quick example that shows the pattern I see most frequently. We want a toggleable button. (Note: this example is a little contrived and a skosh verbose to represent more complicated cases that are solved in exactly the same way.)

.directive( 'myDirective', function () {

    return {

        template: '<a class="btn">Toggle me!</a>',

        link: function ( scope, element, attrs ) {

            var on = false;

            $(element).click( function () {

                if ( on ) {

                    $(element).removeClass( 'active' );

                }

                else {

                    $(element).addClass( 'active' );

                }

                on = !on;

            });

        }

    };

});

There are a few things wrong with this. First, jQuery was never necessary. There's nothing we did here that needed jQuery at all! Second, even if we already have jQuery on our page, there's no reason to use it here; we can simply use angular.element and our component will still work when dropped into a project that doesn't have jQuery. Third, even assuming jQuery was required for this directive to work, jqLite (angular.element) will always use jQuery if it was loaded! So we needn't use the $ - we can just use angular.element. Fourth, closely related to the third, is that jqLite elements needn't be wrapped in $ - the element that is passed to the link function would already be a jQuery element! And fifth, which we've mentioned in previous sections, why are we mixing template stuff into our logic?

This directive can be rewritten (even for very complicated cases!) much more simply like so:

.directive( 'myDirective', function () {

    return {

        scope: true,

        template: '<a class="btn" ng-class="{active: on}" ng-click="toggle()">Toggle me!</a>',

        link: function ( scope, element, attrs ) {

            scope.on = false;

            scope.toggle = function () {

                scope.on = !scope.on;

            };

        }

    };

});

Again, the template stuff is in the template, so you (or your users) can easily swap it out for one that meets any style necessary, and the logic never had to be touched.

And there are still all those other benefits, like testing - it's easy! No matter what's in the template, the directive's internal API is never touched, so refactoring is easy. You can change the template as much as you want without touching the directive. And no matter what you change, your tests still pass.

So if directives aren't just collections of jQuery-like functions, what are they? Directives are actually extensions of HTML. If HTML doesn't do something you need it to do, you write a directive to do it for you, and then use it just as if it was part of HTML.

Put another way, if AngularJS doesn't do something out of the box, think how the team would accomplish it to fit right in with ngClick, ngClass, et al.

**147) How will you display different images based on the status being red, amber, or green?**

Use the ng-switch and ng-switch-when directives as shown below.

<div ng-switch on="account.status">

<div ng-switch-when="AMBER">

<img class="statusIcon"

src='apps/dashboard/amber-dot.jpg' />

</div>

<div ng-switch-when="GREEN">

<img class="statusIcon"

src='apps/dashboard/green-dot.jpg' />

</div>

<div ng-switch-when="RED">

<img class="statusIcon"

src='apps/dashboard/red-dot.jpg' />

</div>

</div>

**148) How will you show/hide buttons and enable/disable buttons conditionally?**

Using the **ng-show** and **ng-disabled** directives.

<div class="dataControlPanel"

ng-show="accounts.releasePortfolios">

<div class="dataControlButtons">

<button class="btn btn-primary btn-small"

ng-click="saveComments()" ng-disabled="disableSaveButton">Save</button>

<button class="btn btn-primary btn-small"

ng-click="releaseRun()" ng-disabled="disableReleaseButton">Release</button>

</div>

</div>

**149) How will you loop through a collection and list each item?**

Using the ng-repeat directive.

<table

class="table table-bordered table-striped table-hover table-fixed-head portal-data-table">

<thead>

<tr>

<th>account</th>

<th>Difference</th>

<th>Status</th>

</tr>

</thead>

<tbody>

<tr

ng-repeat="account in acounts">

<td width="40%">{{account.accountCode}}</td>

<td width="30%" style="text-align: right">{{account.difference

| currency: ""}}</td>

<td width="30%">

<div ng-switch on="account.status">

<div ng-switch-when="AMBER">

<img class="statusIcon"

src='apps/dashboard/amber-dot.jpg' />

</div>

<div ng-switch-when="GREEN">

<img class="statusIcon"

src='apps/dashboard/green-dot.jpg' />

</div>

<div ng-switch-when="RED">

<img class="statusIcon"

src='apps/dashboard/red-dot.jpg' />

</div>

</div>

</td>

</tr>

</tbody>

</table>

**150) Which of the following is true about ng controller directive.**

ng-controller directive tells AngularJS what controller to use with this view. AngularJS application mainly relies on controllers to control the flow of data in the application. A controller is a JavaScript object containing attributes/properties and functions. Each controller accepts $scope as a parameter which refers to the application/module that controller is to control.

**151) What happens when page containing angularjs based application loads.**

When the page is loaded in the browser; HTML document is loaded into the browser, and evaluated by the browser. AngularJS JavaScript file is loaded; the angular global object is created. Next, JavaScript which registers controller functions is executed.

**152) Which angularjs directives creates its own scope.**

Only ng-controller and ng-repeat directive of AngularJS creates its own child scope and attach it to the corresponding DOM element.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**153) What about Auto bootstrap process in AngularJs?**  
In angularjs**ng-app** is used to auto-bootstrapping to your applications.

Its also called Automatic Initialization.

The **ng-app**is the root of compilation and tell only this portion of DOM is angularjs part.

**154) What is ng-cloak directive and Why we use?**

The ng-cloak directive are use to prevent the un-compiled elements from being displayed and un-compiled elements can be an element that hold and wait for incoming data. i.e.

|  |
| --- |
| <div ng-cloack>{{ myCloackVar }}</div> |
|  |

If myCloackVar controller is not compiled or the myCloackVar data is not populated in ng-cloak prevent {{ myCloackVar }} from being displayed and display only the HTML div when the myCloackVar variable is compiled.

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="http://getbootstrap.com/2.3.2/assets/css/bootstrap.css">

<style>

[ng\: cloak], [ng-cloak], [data-ng-cloak], [x-ng-cloak], .ng-cloak, .x- ng-cloak {

display: none !important;

}

</style>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.4/angular.js"></script>

<script>

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

app.service('getDemoService', function($http) {

this.getDemoResult = function() {

return $http({

method: 'GET',

url: baseURL + 'Api/AccountPref/Get/' + uid

});

};

});

app.controller("MyDemoController", function($scope, getDemoService) {

$scope.resultData = [];

getDemoService.getDemoResult().then(function(data) {

var items = data.items;

angular.forEach(items, function(row) {

$scope.resultData.push({

"title": row.title

});

});

});

});

</script>

</head>

<body ng-controller="MyDemoController" ng-cloak>

<div>The ng-cloak example</div>

<div>

<div ng-repeat="rslt in resultData">

{{rslt.title}}

</div>

</div>

</body>

</html>

**155) Differences between $resource and $http.**

The [$resource](http://www.code-sample.com/2015/02/differences-between-resource-and-http.html) is a RESTfulwrapper around the [$http](http://www.code-sample.com/2015/02/differences-between-resource-and-http.html) that takes out all of the manual work needed when using $http.   
The $resourcereturns a resource object that has all the RESTfulmethods.  
The $httpwill use to making a [GET](http://www.code-sample.com/2015/02/differences-between-resource-and-http.html)request for a JSONcalls.

**156) What is $watch in angular?**

The angular create watch internally. The watch means that angular watches the changes in the variable on the $scope object. The watches are created using the [$scope.$watch()](http://www.code-sample.com/2014/11/watch-vs-on-in-angularjs.html) method and the$scope.watch() method creates a watch of some variables.

When you register a watch you need to pass two functions  
        1. One is value function.  
        2. And other is listener function.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ANGULARJS 2 interview question\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#### 157) What is Angular 2

#### Angular 2 is a JavaScript framework for developing web applications. It is a complete rewrite of Angular 1.0 ,so learning AngularJS 1.0 is not required for creating Angular 2 applications. It is well suited for developing mobile applications unlike AngularJS 1.0 which was suitable for building desktop applications.It is also much faster than Angular 1.0.

It supports the modern browsers as well as the older browsers.Applications are better structured than previous versions of Angular.It supports server side rendering for faster rendering of views even on slow devices such as mobile.The size of Angular 2 library is smaller compared to previous versions.Also Angular 2 applications use ahead of time compilation which makes them faster.

#### 158) Data bindings which are supported in Angular 2

a) Interpolation  
b) Property Binding  
c) Event Binding  
d) Two-way Binding

#### 159) Which languages are used to write Angular 2 applications

Angular 2 applications can be written in any of the following languages:

* Typescript   Prefered Language for developing Angular 2 applications.
* Javascript
* Dart

As Angular 2 is written in TypeScript so it is preferable to write Angular 2 applications in TypeScript or ECMA6. [*Typescript*](http://www.codecompiled.com/overview-of-typescript/)is the prefered language to use for developing Angular 2 applications.

#### Components

A component is a building block of Angular 2 application.Angular 2 application is created as a tree of components.A component is declared by using @Component() decorator function.

@Component({

selector: 'first-component',

template: `<p>Hello from first component</p>`

})

class FirstComponent {

}

#### Modules

Angular apps consists of different modules.Modules consists of a collection of components,directives and services.  
Angular modules are created using the NgModules() decorator function.  
Every Angular application consists of a root module apart from other feature modules.Every modules is created using NgModule decorator function.

#### Template

View of a component is declared by using the template. It is the template which is rendered.

#### 160) How Angular 2 application is launched

Angular 2 application is launched by bootstrapping the root module.

#### 161) What is router-outlet

The route which is matched by the router is used display the component.The template defined by the component is displayed in an area defined by the router-outlet

#### NgModule

Its a decorator function which has one argument ,a metadata object with properties describing the module.  
Some of its important properties are:

* declarations views which belong to this module.
* exports declarations which are visible in the components of other modules.
* imports other modules whose classes are needed in this module.
* providers services which are provided by this module
* bootstrap This property is set by the root module

@NgModule({

providers: list of providers,

exports:list of components,

imports:list of components

})

### AOT compilation

AOT compilation stands for  Ahead Of Time compilation, in it angular compiles  components to native JavaScript and HTML during the build time instead of runtime.

# 161) When will ngInit be called? How would you make use of ngOnInit()?

In Angular 1.x, ngInit is called when template is re-rendered. In other words “ng-init” is called, when I take turns back to a page.

In Angular2, there is no “ng-init” but we can create a ways like this using the directive and ngOnInit class. Angular 2 provides life cycle hook ngOnInit by default.

The ngOnInit is invoked when the component is initialized and invoked only once when the directive is instantiated. It is a best practice to implement these life-cycle interfaces.

### 162) List down the advantages and disadvantages associated with Angular2?

#### AngularJs 2 Advantages:

* **Get Rid of ‘$digest already in progress when calling $scope.$apply()’** **Exception**: In Angularjs 1.x, there are multiple times, developer get the exception ‘’ $digest already in progress when calling $scope.$apply()’. The  reason is, Angular 1.x  did not support digest cycle finished event,  because such event might trigger further events as well as changes which can trigger an infinite loop for  the current digest cycle. Developer used to make very distinctive reasoning in order to decide when to call $scope.apply or $scope.digest, which was not always very simple .Moreover, developer needs to use $timeout event to force Angular finish event to end the current digest cycle.  In Angular 2.0 Zone has made a revolutionary change in case of change detection and  zones mechanism  is capable of deciding whether to end digest cycle or not. Therefore, one of the major advantages of Angular 2.0 is change detection with Zone mechanism.
* **Improved Performance**: Another significant disadvantages of Angularjs 1.x was its poor performance, In case of ng-repeat the performance issue was really a blocking issue. However in Angular 2.0, the change detection mechanism does not travel the entire DOM tree to detect a change, rather its only travel a part of it, therefore, the performance has been improved significantly.
* **Improved modularity & Dependency Injection:**Angular modules in Angular 1.x are mostly architecture to group a number of related functions and then used as container for dependency injection containers. However, in 2.0 modular development and component isolation has made it more easier to instantiate a component and used it in its scope. Inheritance has been introduced in component isolation, therefore, a child service can be injected while it will inherits all the services of its parent. Child injectable component or service is also capable of overriding their parent’s functionality.
* **This is more friendly for mobile development and server side rendering.**

#### AngularJs 2 Downside/Disadvantages:

* Those who are not familiar with typescript would take longer time to learn.
* Most of the component need to be developed as the online community is still at the beginning stage.

### 163) What is ECMAScript ES6/ES7? Breifly explain both.

ECMAScript is a standard for modern scripting-language specification. Initially, it was It was JavaScript, now its being changing to ECMAScript.

**6th Edition – ECMAScript 2015:**

ECMAScript 2015 is the 6th Edition. This version has significant contribution for:

* Class Inheritance
* Iterators for loop
* Python-style generators and generator expressions
* Advanced set for collections
* number and math enhancements
* Promises
* Metaprogramming for virtual objects and wrappers and so on.

Although, ES6 still does not support browser. Therefore, compiler such as Traceur are required to compile EC6 code to pure javascript on the fly in browser.

**7th Edition – ECMAScript 2016:**

The 7th edition, officially known as ECMAScript 2016, has been finalized in June 2016.

**164)** **Which version is supported by Angular2?**

Angular has used 6th Edition – ECMAScript 2015 (Es6). However, there are configuration available to try some experimental feature from  7th Edition – ECMAScript 2016 (Es7).

* Inheritance of component has become so easy. Now Angular 2.0 can be developed using  Object oriented thinking. Inheritance has been possible for javascript. A number of libraries has been developed to support this concept. However, with ES2015 all those  nonstandard abstractions can be got rid of. As  ES2015 defines an easier inheritance.

### 165) What is Traceur compiler?

### Traceur compiler compiles ECMAScript Edition 6 (ES6) (including classes, generators and so on) code on the fly to regular Javascript (ECMAScript Edition 5 [ES5]) to make it compatible for the browser.

#### 166) What are the main routing COMPONENTS?

Angular2 provides 3 different components for routing configuration:

1. **Routes** is the configuration to describe application’s different routes
2. **RouterOutlet** is a “placeholder” component that holds the view for each route
3. **RouterLink** is a directive to link to routes

### 166) What are Modules in Angularjs 2.0?

### Both Angular 1.x and 2.0 apps are modular. However, in *****Angular2.0 NgModules***** has been introduced as its modularity system. Every Angular2.0 must have one module which acts as the root module and conventionally named as *****AppModule*****. If the project has been created using angular-cli, *****AppModule***** is created in file src/app/app.module.ts.

### 167) What are Components in Angular 2.0

This is main interesting feature of Angular 2.0. Now it is possible to develop angular app in more object oriented way. It is also possible to pass parameter between components.

In this section lest create two component and communicate between them.

The first component is PeopleComponent which will pass a list of person object to another component called ‘PersonComponent’ to show a list of person on card. The steps to create a component is very simple.

1. Create the template ‘people.component.html’

2. Create class for the component called ‘people.component.ts’

@Component decorators define the selector for the template of this component, the file for the html template has been defined by using  templateUrl , css file has been configured using styleUrls and any services that are required are configured by using providers.

3. Add the component in the main app module

### 4. Place the selector in parent template, in this case put it on app.component.html

### 5. Create the second component called **PersonComponent.**

Here PersonComponent take an input called ‘person’. Therefore, any native element declared with ‘person-detail’ selector will take an input called ‘person’.

6. Add the **PersonComponent** to **AppModule.**

7. Pass the input from the template of **PeopleComponent.**

### Communicating among components is very easy in Angular 2.0.

### 168) Briefly explain Angularjs 2 Directives?

Along with component, Angular 2.0 has also have two other kind of directives such  as  Structural directives and Attribute Directive. Directives are used to change the behavior as well as layout of the DOM elements.  It is possible that the same DOM element may have a number of directives. However, in case of component it is not possible, any DOM element can have only a single component.

### 170) Briefly explain Data Binding in Angularjs 2.0?

From angular 1.x to angular 2.0, the data binding has not been changed in template way. However, from performance point of view this has been changed dramatically.

There are two types of data binding available in angular 2.0

* One way
* Two way

However, any variable is available within the scope of the component and they can be passed from one component to another component. This way, one of the significant concentration that developed used to give in case of variable declaration in angular 1.x has become much simpler.

### 171) Briefly explain Event Binding in Angular 2?

Angular 2 has been developed to be more synchronized with modern HTML5. Therefore, events has been changed to bind to the DOM API Events with similar syntax.

For example, ng-click has been replaced with (click). Same as ng-submit has been changed to  (submit). () parenthesis has been used as the sytax for event emitter.

There is another way of binding event to DOM element is as following

### <input type="text" on-keypress="onKeyPress($event)">

### 172) Explain the life cycle hooks of Angular 2 application

Angular 2 component/directive has lifecycle events, managed by @angular/core. It creates the component, renders it, creates and renders its children, processes changes when its data-bound properties change, and then destroys it before removing its template from the DOM. Angular provides a set of lifecycle hooks(special events) which can be tapped into this lifecycle and perform operations when required. The constructor executes prior to all lifecycle events. Each interface has a single hook method prefixed with ng. For example, *ngOnint*interface has *Oninit*method that must be implemented in the component.

Some of the events are applicable for both component/directives while few are specific to components.

* **ngOnChanges**: Responds when angular sets its data-bound property which receives the current and previous object values.
* **ngOnInit**: Initializes the component/directive after first ngOnChange triggers. This is most frequently used method to retrieve the data for the template from a back-end service.
* **ngDoCheck**: Detect and act upon changes occuring outside Angular context. It is called when every change detection run.
* **ngOnDestroy**: Cleanup just before Angular destroys the directive/component. Unsubscribe observables and detach event handlers to avoid memory leaks.

**Component-specific hooks:**

* **ngAfterContentInit**: Component content has been initialized
* **ngAfterContentChecked**: After Angular checks the bindings of the external content that it projected     into its view.
* **ngAfterViewInit**: After Angular creates the component’s view.
* **ngAfterViewChecked**: After Angular checks the bindings of the component’s view.

### 173) What are the advantages of using Angular 2 over Angular 1?

1. Angular 2 is a platform not only a language:
2. Better Speed and Performance: *No $Scope in Angular 2, AOT*
3. Simpler Dependency Injection
4. Modular, cross platform
5. Benefits of ES6 and Typescript.
6. Flexible Routing with Lazy Loading Features
7. Easier to Learn

### 174) How routing works in Angular 2.

Routing is a mechanism which enables user to navigate between views/components. Angular 2 simplifies the routing and provide flexibility to configure and define at module level (Lazy loading).

The angular application has single instance of the Router service and whenever URL changes, corresponding Route is matched from the routing configuration array. On successful match, it applies redirects and the router builds a tree of ActivatedRoute objects and contains the current state of the router. Before redirection, the router will check whether new state is permitted by running guards ([CanActivate](https://blog.thoughtram.io/angular/2016/07/18/guards-in-angular-2.html)). Route Guards is simply an interface method that router runs to check the route authorization. After guard runs, it will resolve the route data and activate the router state by instantiation the required components into <router-outlet> </router-outlet>.

### 175) What are Event Emitters and how it works in Angular 2?

Angular 2 doesn’t have bi-directional digest cycle, unlike angular 1. In angular 2, any change occurred in the component always gets propagated from the current component to all its children in hierarchy. If the change from one component needs to be reflected to any of its parent component in hierarchy, we can emit the event by using Event Emitter api.

In short, EventEmitter is class defined in @angular/core module which can be used by components and directives to emit custom events.

@output() somethingChanged = new EventEmitter();

We use somethingChanged.emit(value) method to emit the event. This is usually done in setter when the value is being changed in the class.

This event emit can be subscribed by any component of the module by using subscribe method.

myObj.somethingChanged.subscribe(val) => this.myLocalMethod(val));

### 176) What is the use of codelyzer in angular 2 application.

All enterprise applications follows a set of coding conventions and guidelines to maintain code in better way. Codelyzer is an open source tool to run and check whether the pre-defined coding guidelines has been followed or not. Codelyzer does only static code analysis for angular and typescript project.

Codelyzer runs on top of tslint and its coding conventions are usually defined in tslint.json file. Codelyzer can be run via angular cli or npm directly. Editors like Visual Studio Code and Atom also supports codelyzer just by doing a basic settings.

To set up the codelyzer in Visual Studio code, we can go to File -> Preferences -> User Settings and add the path for tslint rules.

{

"tslint.rulesDirectory": "./node\_modules/codelyzer",

"typescript.tsdk": "node\_modules/typescript/lib"

}

To run from cli: ng lint.

To run from npm: npm run lint

### 177) What is lazy loading and How to enable lazy loading in angular 2?

Most of the enterprise application contains various modules for specific business cases. Bundling whole application code and loading will be huge performance impact at initial call. Lazy lading enables us to load only the module user is interacting and keep the rest to be loaded at runtime on demand.

Lazy loading speeds up the application initial load time by splitting the code into multiple bundles and loading them on demand.

Every Angular application must have one main module say AppModule. The code should be splitted into various child modules (NgModule) based on the application business case.

1. We don't require to import or declare lazily loading module in root module.
2. Add the route to top level routing (app.routing.ts) and set loadChildren. loadChildren takes absolute path from root folder followed by #{ModuleName}. RouterModule.forRoot() takes routes array and configures the router.
3. Import module specific routing in the child module.
4. In the child module routing, specify path as empty string ' ', the empty path. RouterModule.forChild again takes routes array for the child module components to load and configure router for child.
5. Then, export const routing: ModuleWithProviders **= RouterModule.forChild**(routes);

### 178) What are the security threats should we be aware of in angular 2 application?

Just like any other client side or web application, angular 2 application should also follow some of the basic guidelines to mitigate the security risks. Some of them are:

1. Avoid using/injecting dynamic Html content to your component.
2. If using external Html, that is coming from database or somewhere outside the application, sanitize it.
3. Try not to put external urls in the application unless it is trusted. Avoid url re-direction unless it is trusted.
4. Consider using AOT compilation or offline compilation.
5. Try to prevent XSRF attack by restricting the api and use of the app for known or secure environment/browsers.

### 179) How would you optimize the angular 2 application for better performance?

Well, optimization depends on the type and size of application and many other factors. But in general, I would consider the following points while optimizing the angular 2 app:

1. Consider AOT compilation.
2. Make sure the application is bundled, uglified, and tree shaking is done.
3. Make sure the application doesn’t have un-necessary import statements.
4. Make sure that any 3rd party library, which is not used, is removed from the application.
5. Have all dependencies and dev-dependencies are clearly separated.
6. I would consider lazy loading instead of fully bundled app if the app size is more.

### 180) How would you define custom Typings to avoid editor warnings?

Well, in most of the cases, the 3rd party library comes with its own .d.ts file for its type definition. In some cases, we need to extend the existing type by providing some more properties to it or if we need to define additional types to avoid Typescript warning.

If we need to extend the type definition for external library, as a good practice, we should not touch the node\_modules or existing typings folder. We can create a new folder, say “custom-typings” and keep all customized type definition in that.

To define typings for application (JavaScript/Typescript) objects, we should define interfaces and entity classes in models folder in the respective module of the application.

For those cases, we can define or extend the types by creating our own “.d.ts” file.

### 181) What is shadow DOM? How is it helping Angular 2 to perform better?

Shadow DOM is a part of the HTML spec which allows developers to encapsulate their HTML markup, CSS styles and JavaScript. Shadow DOM, along with a few other technologies, gives developers the ability to build their own 1st class tags, web components and APIs just like the <audio> tag. Collectively, these new tags and APIs are referred to as Web Components. Shadow DOM provides better separation of concern along with lesser conflict of styles and scripts with other HTML DOM elements.

Since shadow DOM are static in nature, it’s a good candidate to be cached as it is not accessible to developer. The cached DOM would be rendered faster in the browser providing better performance. Moreover, shadow DOM can be managed comparatively well while detecting the change in angular 2 application and re-paint of view can be managed efficiently.

### 182) What is AOT compilation?

AOT compilation stands for Ahead Of Time compilation, in which the angular compiler compiles the angular components and templates to native JavaScript and HTML during the build time. The compiled Html and JavaScript is deployed to the web server so that the compilation and render time can be saved by the browser.

**Advantages**

1. Faster download: Since the app is already compiled, many of the angular compiler related libraries are not required to be bundled, the app bundle size get reduced. So, the app can be downloaded faster.
2. Lesser No. of Http Requests: If the app is not bundled to support lazy loading (or whatever reasons), for each associated html and css, there is a separate request goes to the server. The pre-compiled application in-lines all templates and styles with components, so the number of Http requests to the server would be lesser.
3. Faster Rendering: If the app is not AOT compiled, the compilation process happens in the browser once the application is fully loaded. This has a wait time for all necessary component to be downloaded, and then the time taken by the compiler to compile the app. With AOT compilation, this is optimized.
4. Detect error at build time: Since compilation happens beforehand, many compile time error can be detected, providing a better degree of stability of application.

**Disadvantages**

1. Works only with HTML and CSS, other file types need a previous build step
2. No watch mode yet, must be done manually (bin/ngc-watch.js) and compiles all the files
3. Need to maintain AOT version of bootstrap file (might not be required while using tools like cli)
4. Needs cleanup step before compiling

### 183) What are the core differences between Observables and Promises?

A Promise handles a **single event** when an async operation completes or fails.

Note: There are Promise libraries out there that support cancellation, but ES6 Promise doesn't so far.

An Observable is like a **Stream** (in many languages) and allows to pass zero or more events where the callback is called for each event. Often Observable is preferred over Promise because it provides the features of Promise and more. With Observable it doesn't matter if you want to handle 0, 1, or multiple events. You can utilize the same API in each case. Observable also has the advantage over Promise to be **cancelable**. If the result of an HTTP request to a server or some other expensive async operation isn't needed anymore, the Subscription of an Observable allows to cancel the subscription, while a Promise will eventually call the success or failed callback even when you don't need the notification or the result it provides anymore. Observable provides **operators** like map, forEach, reduce, ... similar to an array. There are also powerful operators like retry(), or replay(), ... that are often quite handy.

**Promises vs Observables**

1. Promises:

2. returns a single value

3. not cancellable

4. Observables:

### 184) Explain local reference variables, ViewChild, and ContentChild.

Local template variables in angular2 is used to refer HTML elements and use their properties to access siblings or children.

Let’s consider you have an input field named username.

<input type="text" required ... />

This HTMLInputField can be made available to the template using # symbol with a variable name say username.

 <input type="text" #username required ... />

Now, this HTMLInputElement can be accessed from anywhere in the current template for example, checking validation and showing appropriate message based on the validation rule. But, username HTML reference is not accessible in the component/directive.

To access this in the component, angular 2 provides @ViewChild decorator which accepts the local reference variable.

@ViewChild('username') username: HTMLInputElement;

ViewChild element can be read after the view is initialized (ngAfterViewInit).

ContentChild is used to query the reference of the DOM within ng-content. Content Child are set before the ngAfterContentInit lifecycle hook.

|  |  |
| --- | --- |
|  |  |
|  |  |