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

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Top 32 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

*“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();

}

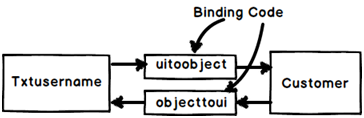
function ObjecttoUi()

{

$("#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.

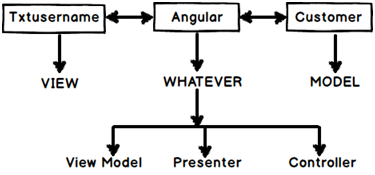
So now whatever you type in the textbox updates the “Customer” object and when the “Customer” object gets updated it also updates the UI.

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

The key features of AngularJS are

* Scope
* Controller
* Model
* View
* Services
* Data Binding
* Directives
* Filters
* Testable
* Validation
* Modules
* Templates
* Expressions
* Routing
* Dependency Injection
* Testing

**3) 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.**

## 4) 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>

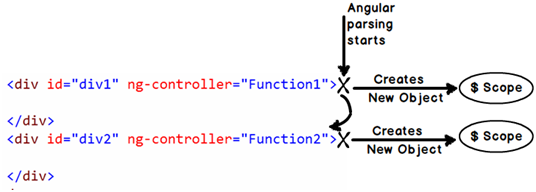
<div id="div2" ng-controller="Function2">

Instance of {{ControllerName}} created

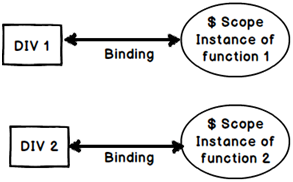
</div>

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.



4) What is $rootScope?

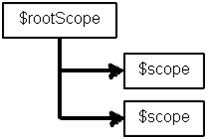
Scope is a special JavaScript object which plays the role of joining controller with the views.   
Scope contains the model data.

In controllers, model data is accessed via $scope object.

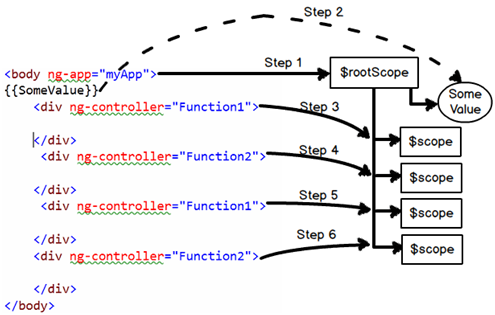
$rootScope is the parent of all of the scope variables.

## 5) 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}}

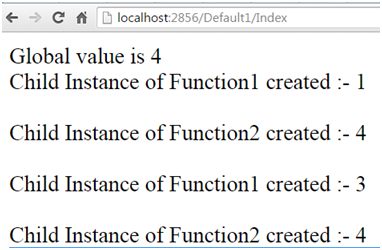
</div><br />

<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.

**6) 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.

## 7) 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*

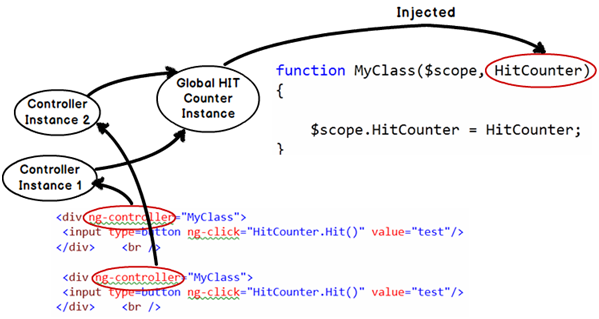
app.controller("MyClass", MyClass); *// Registering the VM*

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.

**8) 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

**9) 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 ( ) ”>

**10) 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.

**11) 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.

**12) 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

**13)  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**)**

## 14)  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.

**15)  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

**16)  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.

**17)  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

## 18) 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.

**19)  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.

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

directive.restrict = 'EA';

**21)  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

**22)  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.

**23)  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.

## 24) Explain compile and link phase?

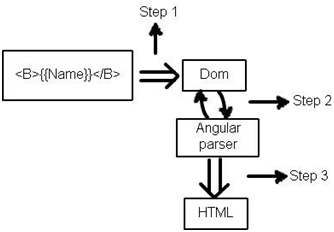
At the heart Angular framework is a parser. A parser which parses the Angular directives and render’s HTML output.

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.

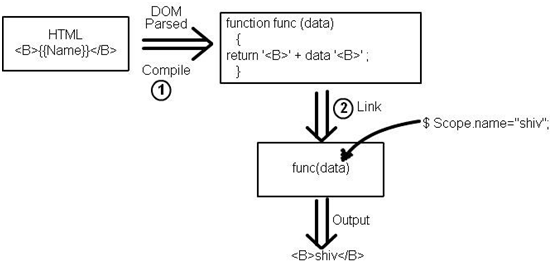


Now the above angular parsing is not so simple as it looks to be. It occurs in two phases “Compile” and “Link”. Firs the compile phase occurs then the link phase.

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.



**25)  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

}

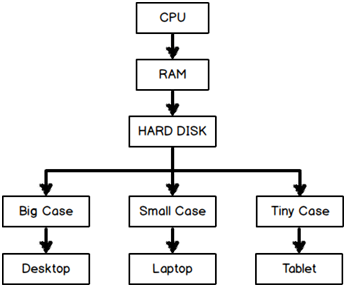
return factory;

});

## 26) 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()

{

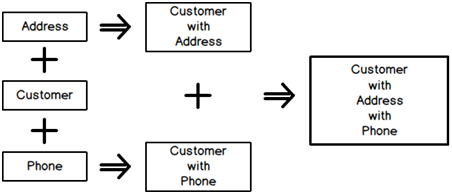
        this.Address1 = "";

        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.



## 27) 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.

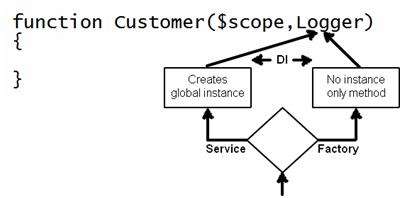
So when we define DI using “service” as shown in the code below. This creates a new GLOBAL instance of the “Logger” object and injects it in to the function.

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. |

**28)  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

**29)  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

**30)  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

## 32) 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*

}

## 33)  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()

{

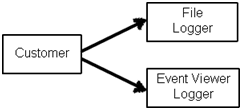
        this.Log = function () {

            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.

**34)  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

**35)  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.

**36)  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.

**37)  Who created Angular JS ?**

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

38) 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.

## 39) 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.

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

1. Eclipse
2. Visual Studio
3. WebStorm
4. TextMate etc.

### 41) 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.

### 42) 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.

### 43) 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.

44) 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.

### 44.1) 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.

## 45) 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>

## 46) 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>

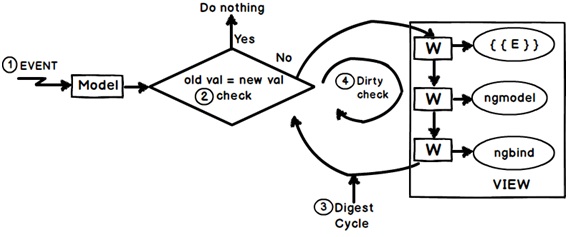
## 47) Explain the concept of digest cycle, watchers and dirty checking?

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:-

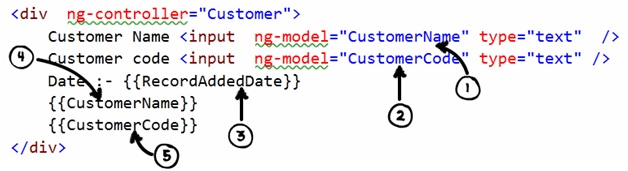
* Digest cycle: - It is a simple loop which updates the model and view.
* Watchers :- They are listeners which are attached to expression and angular directives and fire when the model data changes.
* Dirty check :- This is a extra digest loop which runs to check any cascading left over updates due to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

## 48) 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.

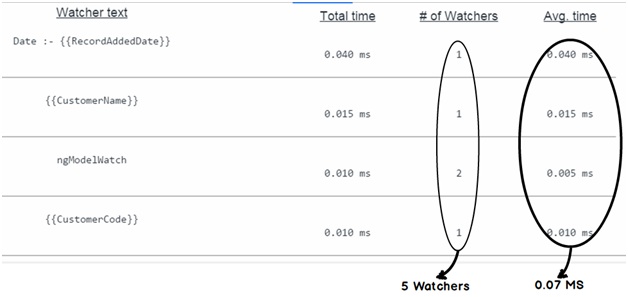
## 49) How can we measureno: of watchers & time spent on digest cycle?

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.



## 50) 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

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

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

## 52) 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*

}

## 53) 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)

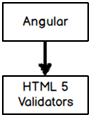
{

*// Here goes code after success*

}

## 54) 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" />

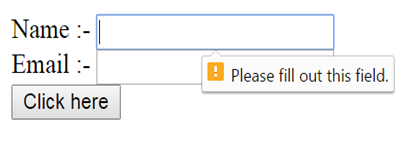
<input type=submit value="Click here"/>

</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.

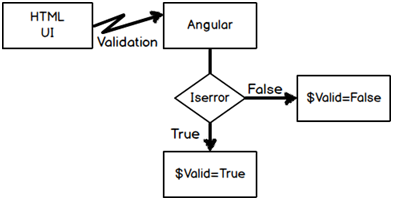
<form name="frm1" novalidate>

-----

</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.*

## 55) 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

## 56) 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.

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

By using Angular routes.

## 58) 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.

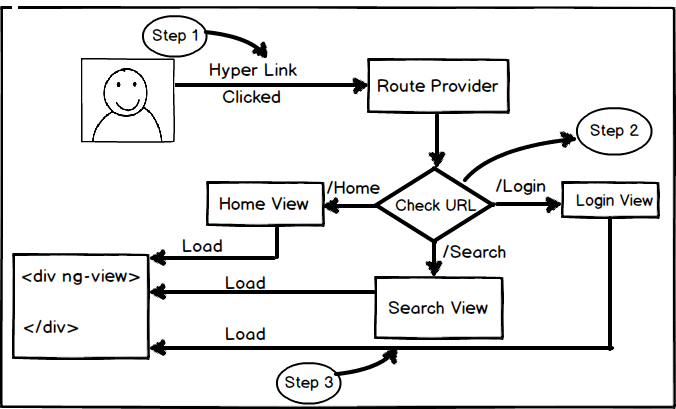
Once the action comes to the router provider it needs a place holder to load views. That’s defined by using the “ng-view” tag on a HTML element. You can see in the below code we have created a “DIV” tag with a place holder. So the view will load in this section.

<div ng-view>

</div>

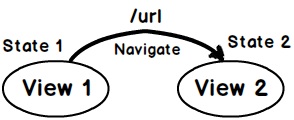
So if we summarize angular routing is a three step process (Below is a visual diagram for the same): -

* 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.



## 59) 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>

## 60) 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>'

})};

## 61) 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);

}

});

## 62) 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>

## 63) 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>

## 64) 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.

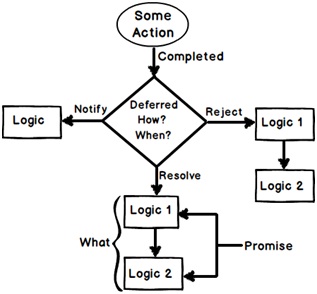
directive.templateUrl = "/templates/footer.html";

## 65) 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.



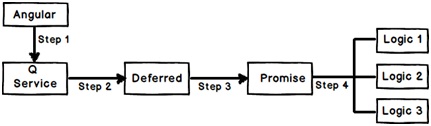
We are thankful to [www.stepbystepschools.net](http://www.stepbystepschools.net/) for the above image.

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.

Hide   Copy Code

*// 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();

66) What is routing in AngularJs?

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

67) What is deep linkingin 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.

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

6. Next, AngularJS executes the controller functions.

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

8. The page gets ready.

69) 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.

70) 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.

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

72) 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.

73) 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.

74) 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.

75) 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.

76) 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.

77) 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.

78) 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.

79) 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.

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

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

82) Explain currency filter.   
Currency filter formats text in a currency format.

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

84)Explain ng-disabled directive.   
ng-disabled directive disables a given control.

85)Explain ng-show directive.   
ng-show directive shows a given control.

86) Explain ng-hide directive.   
ng-hide directive hides a given control.

87) Explain ng-click directive.

ng-cick directive represents a AngularJS click event.

88) 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);

}

});

89) 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

90) How angular.module works?

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

91) 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.

92) 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.

93) 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.

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

95) 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;

};

});

});

96) 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”);

97)

-