\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TOP 100 QUESTIONS & ANSWERS OF ANGULARJS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*Angular js interview question for more than one year of experience\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1. What is the output of result1 and result2?**

var person = {'name' : 'sakshi', 'obj' :{'age': 24 }}, dest1 = {}, dest2 = {};  
angular.copy(person, dest1);  
angular.extend(person, dest2);  
var result1 = person.obj === dest1.obj ? false : true;  
var result2 = person.obj === dest2.obj ? false : true;

A-> true, false

### 2.  What is the output of host1 and host2?

// url → http://user:password@test.com:3000/#/path?param1=xyz¶m2=pqr

host1 = $location.host();

host2 = location.host;

A-> test.com, test.com:3000

### 3. What is the restriction type of below directive?

.directive('myDir', function() {

return {

restrict: 'AEC',

templateUrl: 'my-custom-template.html'

};

});

A-> None of the above

### 4. What are count values respectively when ‘Button One’ is clicked once and ‘Button Two’ clicked twice.

// index.html

<div ng-controller="MyController">  
Root scope <b> MyEvent</b> count: {{count}}  
<ul>  
<li ng-controller="MyController">  
<button ng-click="$emit('MyEvent')">Button Two  
<li ng-controller="MyController">  
Leaf scope <b> MyEvent</b> count: {{count}}  
</li>  
</ul>  
</li>  
</ul>  
</div>

// script.js

angular.module('eventExample', [])  
.controller('MyController', ['$scope', function($scope) {  
$scope.count = 0;  
$scope.$on('MyEvent', function() {  
$scope.count++;  
});  
}]);

A-> 1, 3, 2

### 5. What <p> tag will display?

File Name : index.js  
  
<div ng-controller="myController">  
<input type="input" ng-model="myTitle" />  
<br/> <p class="my-dir" title1="myTitle" title2="myTitle" title3="myTitle"></p>  
</div>  
  
File Name: script.js  
  
var app = angular.module('app', []);  
app.controller("myController", function ($scope) {  
$scope.myTitle = "one";  
});  
app.directive("myDir", function () {  
return {  
replace: true,  
restrict: 'C',  
transclude: true,  
scope: {  
title1: "=",  
title2: "@",  
title3: "&"  
},  
template: "<div><p>Result: {{title1}} / {{title2}} / {{title3}}</p></div>"  
};  
});

A-> Result: one/myTitle/

### 6. What is the output when transclude in directive takes values -1, 0 and 1?

File Name : index.htnl  
  
<div ng-app='myApp'>  
<div ng-controller='myCtrl'> <my-dir>  
<span>User</span> </my-dir>  
</div>  
</div>  
  
File Name : script.js  
  
var app = angular.module('myApp', [])  
app.controller('myCtrl', function ($scope) {  
})  
app.directive('myDir', function () {  
return {  
restrict: 'E',  
transclude: true,  
template: '<div ng-transclude>Hello </div>''  
}  
})

A-> Hello User, Hello, Hello User

### 7. What is result of below line? <div ng-non-bindable>{{1 + 2}}</div>

### A-> None of the above

### 8. What <p> tag will display?

File Name : index.html  
  
<div ng-controller="MyCtrl"> <p>{{one}}, {{two}}, {{three}}, {{four}}</p> </div>  
  
File Name : script.js  
  
var myApp = angular.module('myApp',[]);  
function MyCtrl($scope) {  
function myFunction(fn, input) {  
return (fn || angular.identity)(input);  
};  
$scope.one = myFunction(function(n) { return n + undefined; }, 10);  
$scope.two = myFunction(function(n) { return n + true; }, 10);  
$scope.three = myFunction(function(n) { return n + null; }, 10);  
$scope.four = myFunction(false, 10);  
}

A-> null, 11, 10, 10

### 9. Output of $location.url(), $location.path() for given url? given url http://test.com/#/some/path?param1=foo¶m2=bar

### A-> None of the above

### 10. What <p> tag will display?

File Name: index.html  
  
<div ng-controller="MyCtrl"> <p>{{one}}, {{two}}, {{three}}, {{four}}</p> </div>  
  
File Name: script.js  
  
var myApp = angular.module('myApp',[]);  
function MyCtrl($scope) {  
function myFunction(fn, input) {  
return (fn || angular.identity)(input);  
};  
$scope.one = myFunction(function(n) { return n + undefined; }, 10);  
$scope.two = myFunction(function(n) { return n + true; }, 10);  
$scope.three = myFunction(function(n) { return n + null; }, 10);  
$scope.four = myFunction(false, 10);  
}

A-> null, 11, 10, 10

### 11. What is the output?

File Name: index.html  
  
<div ng-app="Test"> <div ng-controller="Controller"> <ng-pluralize count="people.length" offset="3" when="{ '0': 'Nothing is there.', '1': '{{person1}} is present.', '2': '{{person1}} and {{person2}} are present.', 'one': '{{person1}}, {{person2}} and one other person are present.', 'other': '{{person1}}, {{person2}} and {} other people are present.'}"> </ng-pluralize> </div> </div>  
  
File Name: script.js  
  
angular.module('Test', [])  
.controller('Controller', ['$scope',function ($scope) {  
$scope.people = ['one', 'two', true, false];  
$scope.$watch('people', function () {  
$scope.person1 = ($scope.people[0] ? $scope.people[0] : null);  
$scope.person2 = ($scope.people[1] ? $scope.people[1] : null);  
})  
}  
]);

A-> one, two and one other person are present.

----reference side name->https://www.cronj.com/blog/angular-js-interview-questions-one-year-experience/---------------------------------------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Angular js interview question for fresher\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Which utility is used to track angular watches/digest cycles and what is the good maximum number of watches/digest cycles should be used on a page to improve performance?**

A-> ng-stats, 2000]

### 2. Where singleton pattern is used in angularjs?

A-> b&c

### 3. What are different possible ways to initialize angular app?

A-> a&b&c

### 4. Which one of the following apis to be invoked on rootScope service to get the child scopes?

A-> $new

### 5. What are true about provider?

A-> a&b&c

### 6. Which of the followings can’t be injected as a dependency in Angularjs?

A-> application module

### 7. Which of the following statement is true?

A-> Lowercase filter converts a text to lower case text.

### 8.  What is deep linking in Angular js?

A-> allows to encode the state of application in the URL

### 9. When a scope is terminated how many destroy events will be fired?

A-> 2

### 10. What are the ways to improve Angular js app?

A-> a&c

----reference side name->https://www.cronj.com/blog/angular-js-interview-questions-freshers/---------------------------------------------------------------------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*AngularJS Interview Questions\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**1.** **What are the basic steps to unit test an AngularJS filter?**

**A->**

1. Inject the module that contains the filter.  
2. Provide any mocks that the filter relies on.  
3. Get an instance of the filter using $filter('yourFilterName').  
4. Assert your expectations.

Dependency injection is a powerful software design pattern that Angular employs to compose responsibilities through an intrinsic interface. However, for those new to the process, it can be puzzling where you need to configure and mock these dependencies when creating your isolated unit tests. The open-source project “Angular Test Patterns” is a free resource that is focused on dispelling such confusion through high-quality examples.

This question is useful since it can give you a feel for how familiar the candidate is with automated testing (TDD, BDD, E2E), as well as open up a conversation about approaches to code quality.

**2.** **What should be the maximum number of concurrent “watches”? Bonus: How would you keep an eye on that number?**

**A->**

TL;DR Summary: To reduce memory consumption and improve performance it is a good idea to limit the number of watches on a page to 2,000. A utility called ng-stats can help track your watch count and digest cycles.

Jank happens when your application cannot keep up with the screen refresh rate. To achieve 60 frames-per-second, you only have about 16 milliseconds for your code to execute. It is crucial that the scope digest cycles are as short as possible for your application to be responsive and smooth. Memory use and digest cycle performance are directly affected by the number of active watches. Therefore, it is best to keep the number of watches below 2,000. The open-source utility ng-stats gives developers insight into the number of watches Angular is managing, as well as the frequency and duration of digest cycles over time.

Caution: Be wary of relying on a “single magic metric” as the golden rule to follow. You must take the context of your application into account. The number of watches is simply a basic health signal. If you have many thousands of watches, or worse, if you see that number continue to grow as you interact with your page. Those are strong indications that you should look under the hood and review your code.

This question is valuable as it gives insight into how the candidate debugs runtime issues while creating a discussion about performance and optimization.

**3.** **How do you share data between controllers?**

**A->** Create an AngularJS service that will hold the data and inject it inside of the controllers.

Using a service is the cleanest, fastest and easiest way to test.

However, there are couple of other ways to implement data sharing between controllers,like:  
– Using events  
– Using $parent, nextSibling, controllerAs, etc. to directly access the controllers  
– Using the $rootScope to add the data on (not a good practice)  
The methods above are all correct, but are not the most efficient and easy to test.

**4.** **What is the difference between** **ng-show/ng-hide and ng-if** **directives?**

**A->** ng-show/ng-hide will always insert the DOM element, but will display/hide it based on the condition. ng-if will not insert the DOM element until the condition is not fulfilled.

ng-if is better when we needed the DOM to be loaded conditionally, as it will help load page bit faster compared to ng-show/ng-hide.

We only need to keep in mind what the difference between these directives is, so deciding which one to use totally depends on the task requirements.

**5. What is a digest cycle in AngularJS?**

**A->** In each digest cycle Angular compares the old and the new version of the scope model values. The digest cycle is triggered automatically. We can also use $apply() if we want to trigger the digest cycle manually.

**6. Where should we implement the DOM manipulation in AngularJS?**

**A->** In the directives. DOM Manipulations should not exist in controllers, services or anywhere else but in directives.

**7.** **Is it a good or bad practice to use AngularJS together with jQuery?**

**A->** It is definitely a bad practice. We need to stay away from jQuery and try to realize the solution with an AngularJS approach. jQuery takes a traditional imperative approach to manipulating the DOM, and in an imperative approach, it is up to the programmer to express the individual steps leading up to the desired outcome.

AngularJS, however, takes a declarative approach to DOM manipulation. Here, instead of worrying about all of the step by step details regarding how to do the desired outcome, we are just declaring what we want and AngularJS worries about the rest, taking care of everything for us.

**8.** **If you were to migrate from Angular 1.4 to Angular 1.5, what is the main thing that would need refactoring?**

**A->** Changing .directive to .component to adapt to the new Angular 1.5 components

**9.** **How would you specify that a scope variable should have one-time binding only?**

**A->** By using “::” in front of it. This allows the check if the candidate is aware of the available variable bindings in AngularJS.

**10.** **What is the difference between one-way binding and two-way binding?**

**A->** – One way binding implies that the scope variable in the html will be set to the first value its model is bound to (i.e. assigned to)  
– Two way binding implies that the scope variable will change it’s value everytime its model is assigned to a different value

**11. Explain how $scope.$apply() works**

**A->** $scope.$apply re-evaluates all the declared ng-models and applies the change to any that have been altered (i.e. assigned to a new value)  
Explanation: $scope.$apply() is one of the core angular functions that should never be used explicitly, it forces the angular engine to run on all the watched variables and all external variables and apply the changes on their values

**12.** **What directive would you use to hide elements from the HTML DOM by removing them from that DOM not changing their styling?**

**A->** The ngIf Directive, when applied to an element, will remove that element from the DOM if it’s condition is false.

**13.** **What makes the angular.copy() method so powerful?**

**A->** It creates a deep copy of the variable.

A deep copy of a variable means it doesn’t point to the same memory reference as that variable. Usually assigning one variable to another creates a “shallow copy”, which makes the two variables point to the same memory reference. Therefore if we change one, the other changes as well

**14.** **How would you make an Angular service return a promise? Write a code snippet as an example**

**A->** To add promise functionality to a service, we inject the “$q” dependency in the service, and then use it like so:

angular.factory('testService', function($q){

return {

getName: function(){

var deferred = $q.defer();

//API call here that returns data

testAPI.getName().then(function(name){

deferred.resolve(name)

})

return deferred.promise;

}

}

})

The $q library is a helper provider that implements promises and deferred objects to enable asynchronous functionality

**15.** **What is the role of services in AngularJS and name any services made available by default?**

**A->** – AngularJS Services are objects that provide separation of concerns to an AngularJS app.  
– AngularJS Services can be created using a factory method or a service method.  
– Services are singleton components. All components of the application (into which the service is injected) will work with single instance of the service.  
– An AngularJS service allows developing of business logic without depending on the View logic which will work with it.

Few of the inbuilt services in AngularJS are:  
– the $http service: The $http service is a core Angular service that facilitates communication with the remote HTTP servers via the browser’s XMLHttpRequest object or via JSONP  
– the $log service: Simple service for logging. Default implementation safely writes the message into the browser’s console  
– the $anchorScroll: it scrolls to the element related to the specified hash or (if omitted) to the current value of $location.hash()  
Why should one know about AngularJS Services, you may ask. Well, understanding the purpose of AngularJS Services helps bring modularity to AngularJS code.  
Services are the best may to evolve reusable API within and AngularJS app

**Overview:**

* AngularJS Services help create reusable components.
* A Service can be created either using the service() method or the factory() method.
* A typical service can be injected into another service or into an AngularJS Controller.

**16.** **When creating a directive, it can be used in several different ways in the view. Which ways for using a directive do you know? How do you define the way your directive will be used?**

**A->** When you create a directive, it can be used as an attribute, element or class name. To define which way to use, you need to set the restrict option in your directive declaration.

The restrict option is typically set to:

‘A’ – only matches attribute name  
‘E’ – only matches element name  
‘C’ – only matches class name

These restrictions can all be combined as needed:

‘AEC’ – matches either attribute or element or class name

**17.** **When should you use an attribute versus an element?**

**A->** Use an element when you are creating a component that is in control of the template. Use an attribute when you are decorating an existing element with new functionality.

This topic is important so developers can understand the several ways a directive can be used inside a view and when to use each way.

**18.** **How do you reset a $timeout, $interval(), and disable a $watch()?**

**A->** To reset a timeout and/or $interval, assign the result of the function to a variable and then call the .cancel() function.

var customTimeout = $timeout(function () {

// arbitrary code

}, 55);

$timeout.cancel(customTimeout);

to disable $watch(), we call its deregistration function. $watch() then returns a deregistration function that we store to a variable and that will be called for cleanup

var deregisterWatchFn = $scope.$on(‘$destroy’, function () {

// we invoke that deregistration function, to disable the watch

deregisterWatchFn();

});

**19.** **Explain what is a $scope in AngularJS**

**A->** Scope is an object that refers to the application model. It is an execution context for expressions. Scopes are arranged in hierarchical structure which mimic the DOM structure of the application. Scopes can watch expressions and propagate events. Scopes are objects that refer to the model. They act as glue between controller and view.

This question is important as it will judge a persons knowledge about a $scope object, and it is one of the most important concepts in AngularJS. Scope acts like a bridge between view and model.

**20.** **What are Directives?**

**A->** Directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS’s HTML compiler ($compile) to attach a specified behavior to that DOM element (e.g. via event listeners), or even to transform the DOM element and its children. Angular comes with a set of these directives built-in, like ngBind, ngModel, and ngClass. Much like you create controllers and services, you can create your own directives for Angular to use. When Angular bootstraps your application, the HTML compiler traverses the DOM matching directives against the DOM elements.

This question is important because directives define the UI while defining a single page app. You need to be very clear about how to create a new custom directive or use the existing ones already pre-build in AngularJS.

**21.** **What is DDO Directive Definition Object?**

**A->** “DDO is an object used while creating a custome directive. A standard DDO object has following parameters.

var directiveDefinitionObject = {

priority: 0,

template: '<div></div>', // or // function(tElement, tAttrs) { ... },

// or

// templateUrl: 'directive.html', // or // function(tElement, tAttrs) { ... },

transclude: false,

restrict: 'A',

templateNamespace: 'html',

scope: false,

controller: function($scope, $element, $attrs, $transclude, otherInjectables) { ... },

controllerAs: 'stringIdentifier',

bindToController: false,

require: 'siblingDirectiveName', // or // ['^parentDirectiveName', '?optionalDirectiveName', '?^optionalParent'],

compile: function compile(tElement, tAttrs, transclude) {

return {

pre: function preLink(scope, iElement, iAttrs, controller) { ... },

post: function postLink(scope, iElement, iAttrs, controller) { ... }

}

// or

// return function postLink( ... ) { ... }

},

// or

// link: {

// pre: function preLink(scope, iElement, iAttrs, controller) { ... },

// post: function postLink(scope, iElement, iAttrs, controller) { ... }

// }

// or

// link: function postLink( ... ) { ... }

};"

This question mainly judges whether candidate knows about creating custom directives.

**22.** **What is a singleton pattern and where we can find it in Angularjs?**

**A->** Is a great pattern that restricts the use of a class more than once. We can find singleton pattern in angular in dependency injection and in the services.

In a sense, if you do 2 times ‘new Object()‘ without this pattern, you will be alocating 2 pieces of memory for the same object. With singleton pattern, if the object exists, you reuse it.

**23.** **What is an interceptor? What are common uses of it?**

A-> An interceptor is a middleware code where all the $http requests go through.

The interceptor is a factory that are registered in $httpProvider. You have 2 types of requests that go through the interceptor, request and response (with requestError and responseErrorrespectively). This piece of code is very useful for error handling, authentication or middleware in all the requests/responses.

**24.** **How would you programatically change or adapt the template of a directive before it is executed and transformed?**

**A->** You would use the compile function. The compile function gives you access to the directive’s template before transclusion occurs and templates are transformed, so changes can safely be made to DOM elements. This is very useful for cases where the DOM needs to be constructed based on runtime directive parameters.

**25.** **How would you validate a text input field for a twitter username, including the @ symbol?**

**A->** You would use the ngPattern directive to perform a regex match that matches Twitter usernames. The same principal can be applied to validating phone numbers, serial numbers, barcodes, zip codes and any other text input.

**26.** **How would you implement application-wide exception handling in your Angular app?**

**A->** Angular has a built-in error handler service called $exceptionHandler which can easily be overriden as seen below:

myApp.factory('$exceptionHandler', function($log, ErrorService) {

return function(exception, cause) {

if (console) {

$log.error(exception);

$log.error(cause);

}

ErrorService.send(exception, cause);

};

});

This is very useful for sending errors to third party error logging services or helpdesk applications. Errors trapped inside of event callbacks are not propagated to this handler, but can manually be relayed to this handler by calling $exceptionHandler(e) from within a try catch block.

**27.** **How do you hide an HTML element via a button click in AngularJS?**

**A->** You can do this by using the ng-hide directive in conjunction with a controller we can hide an HTML element on button click.

<div ng-controller="MyCtrl">

<button ng-click="hide()">Hide element</button>

<p ng-hide="isHide">Hello World!</p>

</div>

function MyCtrl($scope){

$scope.isHide = false;

$scope.hide = function(){

$scope.isHide = true;

}

}

**28.** **How would you react on model changes to trigger some further action? For instance, say you have an input text field called**email**and you want to trigger or execute some code as soon as a user starts to type in their email.**

**A->** We can achieve this using $watch function in our controller.

function MyCtrl($scope) {

$scope.email = "";

$scope.$watch("email", function(newValue, oldValue) {

if ($scope.email.length > 0) {

console.log("User has started writing into email");

}

});

}

**29.** **How do you disable a button depending on a checkbox’s state?**

**A->** We can use the ng-disabled directive and bind its condition to the checkbox’s state.

<body ng-app>

<label><input type="checkbox" ng-model="checked"/>Disable Button</label>

<button ng-disabled="checked">Select me</button>

</body>

----------reference side name->https://www.codementor.io/angularjs/tutorial/angularjs-interview-questions-sample-answers------------------------------------------------------------------------------------------------