**#Angular-Js Interview Questions**

**1) 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.

**2) 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.

**3)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.

**4) What are the characteristics of ‘Scope’?**

Scope is an object that refers to the application model. It is an execution context for expressions. Scopes are arranged in hierarchical structure which mimics the DOM structure of the application. Scopes can watch expressions and propagate events. The characteristics of Scope are:

Scopes provide APIs ($watch) to observe model mutations.

Scopes provide APIs ($apply) to propagate any model changes through the system into the view from outside of the “Angular realm” (controllers, services, Angular event handlers).

Scopes can be nested to limit access to the properties of application components while providing access to shared model properties. Nested scopes are either “child scopes” or “isolate scopes”. A “child scope” (prototypically) inherits properties from its parent scope. An “isolate scope” does not. See isolated scopes for more information.

Scopes provide context against which expressions are evaluated. For example {{username}} expression is meaningless, unless it is evaluated against a specific scope which defines the username property.

**5) What is factory method in AngularJS?**

A)Factory method is used for creating a directive. It is invoked when the compiler matches the directive for the first time. We can invoke the factory method using **$injector.invoke**.

**Syntax: module.factory( 'factoryName', function );**

Result: When declaring factoryName as an injectable argument you will be provided with the value that is returned by invoking the function reference passed to module.factory.

**6)Explain $q service, deferred and promises.**

‘Promises’ are post processing logics which are executed after some operation/action is completed whereas ‘deferred’ is used 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.

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

These are some of the frequently asked AngularJS interview questions with answers. You can brush up on your knowledge of AngularJS with these blogs. You can also access a tutorial here.

Feeling overwhelmed with all the questions the interviewer might ask in your AngularJS interview? It’s never too late to strengthen your basics. Learn from industry experts on how to use AngularJS in real life use cases via a structured course.

**7) How Is AngularJS Compiled?**

Agular’s HTML compiler allows you to teach the browser, new HTML syntax. It allows the developer to attach new behaviors or attributes to any HTML element called as directives. AngularJS compilation process takes place in the web browser itself. It does not involve any server-side or pre-compilation step.

AngularJS uses <$compiler> service to compile the angular HTML page. Its compilation begins after the HTML page (static DOM) is fully loaded.

It occurs in two phases.

**Compile** – It looks into the entire DOM and collects all of the directives. The result is a linking function.

**Link** – It combines the directives with a scope and produces a live view. Any changes in the <scope model> get reflected in the view, and any operations done by the user in the view gets reflected in the <scope model>.

The concept of compile and link has come from C language. Here the code is compiled first and then linked

**8)What Is The Difference Between $Scope And Scope.**

It is mandatory to use <$scope> while defining a controller. However, the “scope” will be used to create a link function for the custom directive. Both of them refer to “scope” object in AngularJS. The difference between them is that <$scope> uses dependency injection whereas “scope” does not.

Factory methods like controller, filter, service etc receive its arguments via dependency injection (DI). In DI, the order of passing the arguments does not matter. For example, a controller may be defined as (let’s define $scope as the first parameter in this case):

**myApp.controller(‘MyController’, [‘$scope’, function($scope, $http) {**

**//rest of the code goes here }**

Thus, AngularJS does not care for the position of “$scope” in both the cases. It uses the argument name to retrieve an object out of the dependency injection container.

But, in the case of directive linker function, the position of scope matters, as it does not use DI. The reason being that the supplied arguments are received by its caller. In this case, the very first parameter has to be the scope as per AngularJS syntax.

**app.directive(“myDirective”, function() {**

**return {**

**scope: {};**

**link: function(scope, element, attrs) {**

**// code goes here.**

**}**

**};**

**});**

In the case of non-dependency injected arguments, we can also give a name to injected objects. The above code can be re-written as:

**app.directive(“myDirective”, function() {**

**return {**

**scope: {};**

**link: function(foo, bar, biz) {**

**// code goes here.**

**}**

**};**

**});**

**9) What Is Ng-Template In AngularJs**

The ng-template directive is used to create an HTML view using script tag. It contains “id” attribute which is used by $routeProvider to map a view with a controller.

While defining ng-template, it is mandatory to specify the type of the <script> element as text/ng-template. Also, assign a cache name to the template using the element’s id. Later on, this name gets used as directive’s templateUrl.

Following is the syntax of using a ng-template directive in angularJS application.

<div ng-app = "mainApp">

  <scrip t type = "text/ng-template" id = "addEmployee.htm">

      <h2> Add Employee </h2>

      {{message}}

   </scrip t>

</div>

$routeProvider part.

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

mainApp.config(['$routeProvider', function($routeProvider) {

   $routeProvider.

   when('/addEmployee', {

      templateUrl: 'addEmployee.htm', controller: 'AddEmployeeController'

   }).

otherwise({

      redirectTo: '/addEmployee'

   });

}]);

**10) What Is A Factory Method In AngularJS?**

A factory is a simple function which allows you to add some logic before creating the object. In the end, it returns the created object.

**app.factory('serviceName',function(){ return serviceObj;})**

#### Creating Service Using The Factory Method

<script>

//creating module

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

//define a factory using factory() function

app.factory('MyFactory', function () {

  var serviceObj = {};

serviceObj.function1 = function () {

//TO DO:

};

serviceObj.function2 = function () {

//TO DO:

};

return serviceObj;

});

</script>

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

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

#### A) $Watch() –

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

$watch(watchExpression, listener, [objectEquality]).

<html>

<head>

    <title>AngularJS Watch</title>

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

    <script>

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

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

              ($scope) {

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

               $scope.counter = 0;

              //watching change in name value

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

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

                       });

});

</script>

</head>

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

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

<br /><br />

Counter: {{counter}}

</body>

</html>

#### B) $Digest() –

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

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

<html>

<head>

<title>AngularJS Digest Example</title>

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

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

</head>

<body ng-app="app">

     <div ng-controller="Ctrl">

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

                 <br />

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

    </div>

<script>

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

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

            $scope.obj = { value: 1 };

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

                 console.log("digest clicked!");

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

                 //update value

                $scope.$digest();

});

});

</script>

</body>

</html>

#### C) $Apply() –

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

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

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

<html>

<head>

    <title>AngularJS Apply Example</title>

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

   <script>

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

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

        ($scope) {

               $scope.datetime = new Date();

              $scope.updateTime = function () {

                 $scope.datetime = new Date();

                             }

       //outside angular context

document.getElementById("updateTimeButton").addEventListener('click', function () {

//update the value

$scope.$apply(function () {

          console.log("update time clicked");

          $scope.datetime = new Date();

          console.log($scope.datetime);

  });

});

});

</script>

</head>

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

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

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

<br />

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

</body>

</html>