1. **What are the basic steps to unit test an AngularJS filter?**
2. Inject the module that contains the filter.
3. Provide any mocks that the filter relies on.
4. Get an instance of the filter using $filter('yourFilterName').
5. 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.

Sources:

https://github.com/daniellmb/angular-test-patterns/blob/master/patterns/filter.md

https://docs.angularjs.org/guide/unit-testing

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

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.

Sources:

https://github.com/kentcdodds/ng-stats

http://jankfree.org

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

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.

Here is a good video explanation on egghead.io.

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

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.

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

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.

For more information, take a look in the ng-book explanation: The Digest Loop and $apply

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

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

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

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.

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

Changing .directive to .component to adapt to the new Angular 1.5 components.

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

By using “::” in front of it. This allows you to check if the candidate is aware of the available variable bindings in AngularJS.

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

– 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

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

$scope.$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.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

Source: https://docs.angularjs.org/api/ng/type/$rootScope.Scope

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

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

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

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 one is changed, the other changes as well.

Sources:

https://docs.angularjs.org/api/ng/function/angular.copy

https://en.wikipedia.org/wiki/Object\_copying

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

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.

Source: https://docs.angularjs.org/api/ng/service/$q

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

AngularJS Services are objects that provide separation of concerns to an AngularJS app. These can be created using a factory method or a service method. Services are singleton components and 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.

Sources:

https://docs.angularjs.org/guide/services

http://www.tutorialspoint.com/angularjs/angularjs\_services.htm

1. **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?**

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

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

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.

Source: https://docs.angularjs.org/api/ng/service/$compile#directive-definition-object

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

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();

});

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

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 developer's 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.

Source: https://docs.angularjs.org/guide/scope

1. **What are Directives?**

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. Candidates need to be very clear about how to create a new custom directive or use the existing ones already pre-build in AngularJS.

Source: https://docs.angularjs.org/guide/directive

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

“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.

Read more at https://docs.angularjs.org/guide/directive

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

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 the candidate does 2 times ‘new Object()‘ without this pattern, the candidate will be alocating 2 pieces of memory for the same object. With singleton pattern, if the object exists, it'll be reused.

Source: http://joelhooks.com/blog/2013/05/01/when-is-a-singleton-not-a-singleton/

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

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

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

Source: https://docs.angularjs.org/api/ng/service/$http

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

The candidate should use the compile function. The compile function gives 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.

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

The developer should use the ngPatterndirective 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.

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

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.

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

This can be done by using the ng-hide directive in conjunction with a controller to 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;

};}

1. **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.**

This can be achieved by using $watch function in the 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');

}

});

}

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

The developer 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>

**Question 1: What is AngularJS?**  
  
**Answer**  
  
AngularJS has been introduced by the giant, Google. It is a framework that helps you to create dynamic Web apps. Normally, AngularJS uses HTML as the backbone. AngularJS creates extended HTML tags that can be used as normal HTML tags. These tags will help you to write an efficient code. The interesting fact is that you can reduce the lines of code  you may need to write when you use normal JavaScript.  
  
AngularJS is an open-source JavaScript framework developed by Google. It is a structural framework for dynamic Web apps. It is easy to update and get information from your HTML document. It helps in writing a proper maintainable architecture, that can be tested at a client side code.

* This framework is developed on MVC (Model-View-Controller) design pattern.
* It provides full featured SPA (Single Page Application) framework.
* It supports Dependency Injection.
* It supports two-way data binding.
* It provides routing features.
* Testing was designed right from the beginning; so you can build robust tests.
* For DOM manipulation, jqLite is built-in; which is kind of like the Mini-Me of jQuery.
* Separation of the client side of an Application from the Server side.
* The AngularJS framework uses Plain Old JavaScript Objects(POJO), it doesn’t need the getter or setter functions.

**Using the code**  
Let's start using AngularJS. What would be the first step that you need to do? That would be to include the relevant JavaScript file as in the following:  
  
*<script src="~/Script/angular.min.js"></script>*  
  
**For more details, click the link**

* [Basics of AngularJS](http://www.c-sharpcorner.com/UploadFile/65794e/basics-of-angularjs/)

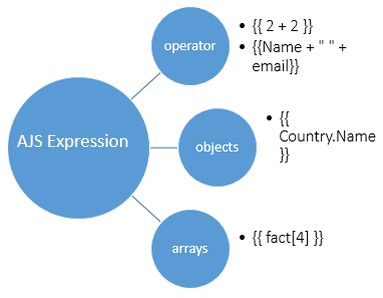
**Question 2: Explain Directives in AngularJS?  
  
Answer**AngularJS directives are only used to extend HTML and DOM elements' behavior. These are the special attributes, that start with ng- prefix, that tell AngularJS's HTML compiler ($compile) to attach a specified behavior to that DOM element.  
  
**AngularJS has a set of built-in directives like**

* ngBind,
* ngModel
* ngClass
* ngApp
* ngInit
* ngRepeat

We can create our own directives for Angular to use them in our AngularJS Application with the controllers and services too. In this article, we’ll learn about some most important built-in directives like: ng-app, ng-init, ng-model, ng-bind and ng-repeat.  
  
**ng-app**It is the most important directive for an Angular Application, which is used to indicate starting of an Angular Application to AngularJS HTML compiler ($compile), like a “Main()” function in any compile time language like C#, Java or C++ etc. If we do not use this directive first and directly try to write other directives, it gives an error.  
  
**ng-init**  
  
ng-init directive is used to initialize an AngularJS Application data variable's inline statement, so that we can use those in the specified block where we declare them. It is like a local member of that ng-app and it can be a value or a collection of the values and as an array, it directly supports JSON data.  
  
**ng-model**  
  
ng-model directive is used to define the model/variables value to be used in AngularJS Application’s HTML controls like <input type=’text’/> and it also provides two-way binding behavior with the model value. In some cases, it’s also used for databinding.  
  
**ng-bind**  
  
ng-bind directive is also used to bind the model/variable's value to AngularJS Applications HTML controls as well as with HTML tags attributes like: <p/>, <span/> and more, but it does not support two way binding. We can just see the output of the model values.  
  
**ng-repeat**  
  
ng-repeat directive is used to repeat HTML statements. Ng-repeat works the same as for each loop in C#, Java or PHP on a specific collection item like an array.  
  
**For more details, click the link**

* [What Are Directives In AngularJS: Part Three](http://www.c-sharpcorner.com/article/what-are-the-directives-in-angulasjs-part-three/)

**Question 3: What are expressions in AngularJS?**  
**Answer**Expressions in AngularJS are just like JavaScript code snippets. JavaScript code is usually written inside double braces: {{expression}}. In other words, Angular Expressions are JavaScript code snippets with limited sub-set. Expressions are included in the HTML elements.   
  
Like JavaScript expressions, AngularJS expressions can also have various valid expressions. We can use the operators between numbers and strings, literals, objects and arrarys inside the expression {{ }}. For example,

* *{{ 2 + 2 }} (numbers)*
* *{{Name + " " + email}} (string)*
* *{{ Country.Name }} (object)*
* *{{ fact[4] }} (array)*  
    
  

**Example**

1. <div ng-controller="appController">
2. <span>
3. 4+5 = {{4+5}}
4. </span>
5. <br />
6. <br />
7. <span ng-init="quantity=5;cost=25">
8. Total Cost = {{quantity \* cost}}
9. </span>
10. </div>

**For more details click on the link**

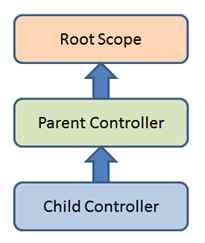
* [Expressions In AngularJS](http://www.c-sharpcorner.com/UploadFile/ff2f08/expression-in-angularjs/)

**Question 4: Explain currency filter in AngularJS  
  
Answer**One of the filters in AngularJS is the Currency Filter. This “currency” filter includes the “$” Dollar Symbol as the default. So we can use the following code as the html template format of Currency Filter.   
  
*{{ currency\_expression | currency : symbol : fractionSize}}*  
  
**How to use Currency Filter in AngularJS**  
There are two ways by which we can use Currency Filter.

* **Default**   
    
  If we did not provide any currency symbol then by default Dollar-Sign will be used; we can use it as follows:  
    
  *<!-- by default -->*  
  **Default Currency** *{{amount | currency}}*
* **User Defined**To use different type of currency symbols we have to define our own symbol by using the unicode or Hexa-Decimal code of that Currency.  
    
  E.g. - For Example If we want to define Indian Currency Symbol then we have to use (Unicode-value) or (Hexa-Decimal value)  
  **Indian Currency***{{amount | currency:"&# 8377"}}*

**For more details click on the link**

* [Currency Filter In AngularJS](http://www.c-sharpcorner.com/article/currency-filter-in-angular-js/)

**Question 5: What is $scope in AngularJS?  
  
Answer**   
  
$scope in AngularJS is an object which refers to an application model. It is an object that binds view (DOM element) with the controller. In controller, model data is accessed via $scope object. As we know, AngularJS supports MV\* pattern, $scope object becomes the model of MV\*.   
  
The $scope is a special JavaScript object. Both View and controller have access to the scope object. It can be used for communication between view and controller. Scope object contains both data and functions. Every AngularJS application has a $rootScope that is the top most scope created on the DOM element which contains the ng-app directive. It can watch expressions and propagate events.   
  
  
  
**Characteristics of scope object**

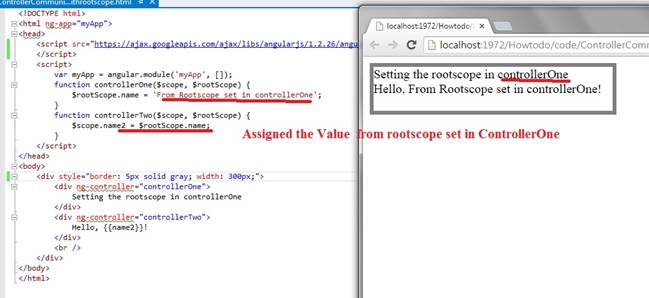
* It provides the APIs to observe model (example $watch).
* It can be nested, so that it limits access to the properties. Nested scopes are either child scope or isolated scope.
* It provides the APIs to propagate any model changes from the outside of "Angular realm" (example $apply).
* It provides context against the expression to be evaluated.

**Example**  
In the following example, I have created three controllers: parentController, firstChildControllerand secondChildController and defined one property in each controller; parentName, level1name, and level2name respectively. Here controllers are attached with DOM elements in a nested way.   
  
As described above, AngularJS evaluates expressions with current associated scope and then it searches in parent scope and so on until the root scope is reached.  
  
**TestAngularJs.html**

1. <!DOCTYPE html>
2. <html>
4. <head>
5. <title>AngularJS Test Application</title>
6. <script src="angular.js"></script>
7. </head>
9. <body ng-app="myapp">
10. <h2>AngularJS - Scope Inheritance</h2>
11. <div ng-controller="ParentController">
12. <div ng-controller="firstChildController">
13. <div ng-controller="secondChildController">
14. <p>Parent Name:{{parentName}}</p>
15. <p>First Child Name:{{level1name}}</p>
16. <p>Second Child Name:{{level2name}}</p>
17. </div>
18. </div>
19. </div>
21. <script>
22. var app = angular.module("myapp", []);
24. app.controller("ParentController", function($scope)
25. {
26. $scope.parentName = "Parent Controller";
27. });
29. app.controller("firstChildController", function($scope)
30. {
31. $scope.level1name = "First Child Controller";
32. });
33. app.controller("secondChildController", function($scope)
34. {
35. $scope.level2name = "Second Child Controller";
36. });
37. </script>
38. </body>
40. </html>

**For more details click on the link**

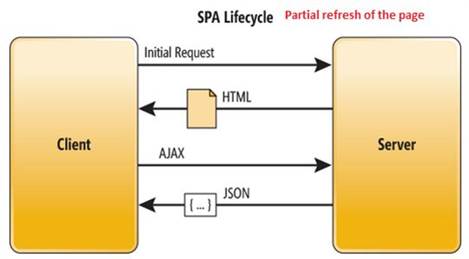
* [Scope In AngularJS](http://www.c-sharpcorner.com/UploadFile/ff2f08/scope-in-angularjs/)

**Question 6: What is “$rootScope” in AngularJS?**  
**Answer**A scope provides a separation between View and its Model. Every application has a $rootScope provided by AngularJS and every other scope is its child scope.  
  
**Using $Rootscope**Using rootscope we can set the value in one controller and read it from the other controller.  
  
**The following is the sample code snippet,**  


1. <!DOCTYPE html>
2. <html ng-app="myApp">
4. <head>
5. <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.26/angular.min.js">
6. </script>
7. <script>
8. var myApp = angular.module('myApp', []);
10. function controllerOne($scope, $rootScope)
11. {
12. $rootScope.name = 'From Rootscope set in controllerOne';
13. }
15. function controllerTwo($scope, $rootScope)
16. {
17. $scope.name2 = $rootScope.name;
18. }
19. </script>
20. </head>
22. <body>
23. <div style="border: 5px solid gray; width: 300px;">
24. <div ng-controller="controllerOne">
25. Setting the rootscope in controllerOne
26. </div>
27. <div ng-controller="controllerTwo">
28. Hello, {{name2}}!
29. </div>
30. <br />
31. </div>
32. </body>
34. </html>

As we know, Rootscope is the top-level data container in AngularJs, we can keep any data in rootscope and read it when needed.  
  
**For more details click on the link**

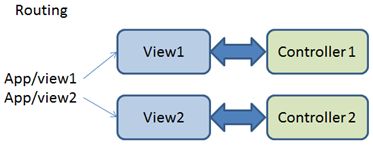
* [Communication Among Controllers in AngularJS](http://www.c-sharpcorner.com/UploadFile/dev4634/communication-between-controllers-in-angular-js/)

**Question 7: What is SPA (Single page application) in AngularJS?  
  
Answer**   
  
Single-Page Applications (SPAs) are web applications that load a single HTML page and dynamically update that page as the user interacts with the app. SPAs use AJAX and HTML to create fluid and responsive web apps, without constant page reloads. However, this means much of the work happens on the client side, in JavaScript.  
  
A single HTML page here means UI response page from the server. The source can be ASP, ASP.NET, ASP.NET MVC, JSP and so on.  
  
A single-page web application, however, is delivered as one page to the browser and typically does not require the page to be reloaded as the user navigates to various parts of the application. This results in faster navigation, more efficient network transfers, and better overall performance for the end user.  
  
  
  
**Key Points of Single-Page Applications**

* The application is responsive in the UI with no page flicker
* The Back/Forward buttons work as expected
* More JavaScript than actual HTML
* Dynamic data loading from the server-side API works with restful web service with JSON format
* Rich interaction among UI components
* Fewer data transfers from the server and most of the page processes in the UI occurs client-side.
* The application contains tabs and subtabs with multiple HTML containers on the click of the tabs or subtabs and the specific portions of the page that are loaded into the page (the page will be one using the application)
* Applications written in AngularJS are cross-browser compliant. Angular automatically handles the JavaScript code suitable for each browser.

**For more details click on the link**

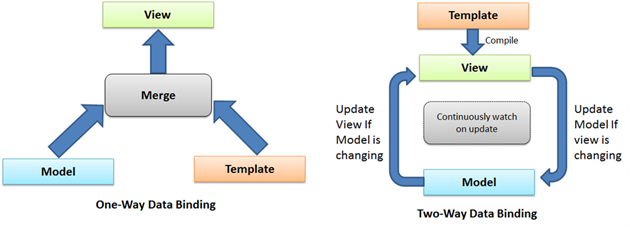
* [Single Page Application and AngularJs Background](http://www.c-sharpcorner.com/UploadFile/dev4634/single-page-application-and-angular-js-background/)

**Question 8: How to implement routing in AngularJS?  
  
Answer**Routing is a core feature in AngularJS. This feature is useful in building SPA (Single Page Application) with multiple views. In SPA application, all views are different Html files and we use Routing to load different parts of the application and it's helpful to divide the application logically and make it manageable. In other words, Routing helps us to divide our application in logical views and bind them with different controllers.  
  
  
  
**How to add routing**The$routeProvider definition contained by the module is called "ngRoute". In app.js file, I have defined an angular app using “angular. Module” method. After creating module, we need to configure the routes. The "config" method is used to configure $routeProvider. Using "when" and "otherwise" method of $routeProvider, we can define the route for our AngularJS application.

1. var app = angular.module("AngularApp", ['ngRoute']);
2. app.config(['$routeProvider',
3. function($routeProvider)
4. {
5. $routeProvider.
6. when('/page1',
7. {
8. templateUrl: 'Modules/Page1/page1.html',
9. controller: 'Page1Controller'
10. })
11. .
12. when('/page2',
13. {
14. templateUrl: 'Modules/Page2/page2.html',
15. controller: 'Page2Controller'
16. })
17. .
18. otherwise
19. ({
20. redirectTo: '/page1'
21. });
22. }
23. ]);

**For more details click on the link**

* [Routing In AngularJS](http://www.c-sharpcorner.com/UploadFile/ff2f08/routing-in-angularjs/)

**Question 9: How many types of data binding in AngularJS?  
  
Answer**Data binding is a very powerful feature of the software development technologies. Data binding is the connection bridge between view and business logic (view model) of the application. Data binding in AngularJs is the automatic synchronization between the model and view. When the model changes, the view is automatically updated and vice versa. AngularJs support one-way binding as well as two-way binding.  
  
  
Figure 1: One-Way and Two-Way Data Binding  
  
**Binding Directives in AngularJs**

* ng-bind
* ng-bind-html
* ng-bind-template
* ng-non-bindable
* ng-model

**ng-bind**This directive updates the text content of the specified HTML element with the value of the given expression and the text content is changing on expression changes. It is very similar to double curly markup ( {{expression }}) but less verbose.  
  
**Syntax**

1. <ANY ELEMENT ng-bind="expression"> </ANY ELEMENT>

**Ng-bind-html**It (whatever it is) evaluates the expression and inserts the HTML content into the element in a secure way. It uses the $sanitize service, so to use this functionality, be sure that $sanitize is available.  
  
**Syntax**

1. <ANY ELEMENT ng-bind-html=" expression "> </ANY ELEMENT>

**ng-bind-template**It (whatever it is) replaces the element text content with the interpolation of the template. It can contain multiple double curly markups.  
  
**Syntax**

1. <ANY ELEMENT ng-bind-template=" {{expression1}} {{expression2}} … {{expressionn}} "> </ANY ELEMENT>

**ng-non-bindable**   
  
This (whatever "this" is) directive informs AngularJs to not compile or bind the contents of the current DOM element This element is useful when we want to display the expression but it should not be executed by AngularJs.  
  
**Syntax***<ANY ELEMENT ng-non-bindable > </ANY ELEMENT>*  
  
**ng-model**  
This (whatever "this" is) directive can be bound with input, select, text area or any custom form control. It provides two-way binding. It also provides validation behavior. It also keeps the state of the control (valid/invalid, dirty/pristine, touched/untouched and so on).  
  
**Syntax**  
*<input ng-bind="expression"/>*  
  
**For more details click on the link**

* [Data Binding in AngularJS](http://www.c-sharpcorner.com/UploadFile/ff2f08/data-binding-in-angularjs/)

**Question 10: What is a Factory method in AngularJS?  
  
Answer**  
  
AngularJS Factory: the purpose of Factory is also the same as Service, however in this case we create a new object and add functions as properties of this object and at the end we return this object.  
  
**Factories***module.factory( 'factoryName', function );*  
  
**Example**

1. <div ng-app="Myapp">
2. <div ng-controller="exampleCtrl">
3. <input type="text" ng-model="num.firstnumber" />
4. <input type="text" ng-model="num.secondnumber" />
5. <input type="button" ng-click="Factoryclick()" value="Factoryclick" />
6. <input type="button" ng-click="servclick()" value="Serviceclick" /> factory result {{facresult}} service result {{secresult}}
7. </div>
8. </div>
9. var myapp = angular.module('Myapp', []);
10. myapp.controller('exampleCtrl', ['$scope', '$http', 'factories', 'services', function (scope, http, fac, ser)
11. {
12. scope.Factoryclick = function ()
13. {
14. var firstnumber = parseInt(scope.num.firstnumber);
15. var secondnumber = parseInt(scope.num.secondnumber);
16. scope.facresult = fac.sumofnums(firstnumber, secondnumber);
17. }
18. scope.servclick = function ()
19. {
20. var firstnumber = parseInt(scope.num.firstnumber);
21. var secondnumber = parseInt(scope.num.secondnumber);
22. debugger;
23. scope.secresult = ser.sersumofnums(firstnumber, secondnumber);
24. }
25. }]);
26. myapp.factory('factories', function ($http)
27. {
28. return {
29. sumofnums: function (a, b)
30. {
31. return a + b;
32. }
33. }
34. });
35. myapp.service('services', function ($http)
36. {
37. debugger;
38. this.sersumofnums = function (a, b)
39. {
40. return a + b;
41. };
42. });

When to use Factory: It is just a collection of functions like a class. Hence, it can be instantiated in different controllers when you are using it with a constructor function.  
  
**For more details click on the link**

* [Service and Factory in AngularJS](http://www.c-sharpcorner.com/code/1869/service-and-factory-in-angularjs.aspx)
* [Services in AngularJS Simplified With Examples](http://www.c-sharpcorner.com/UploadFile/dhananjaycoder/services-in-angularjs-simplified-with-examples/)

**Question 11: How are validations implemented in AngularJS?**  
**Answer**One of the coolest features of AngularJS is client-side validation. There are so many form directives available in AngularJS. We will talk about some of them here, we will also explain custom validation. Using it you can create your own validations.  
  
Initial requirement is reference,

1. <script src="~/Scripts/angular.js"></script>
2. **Data type validation**a.In Html control use type field to specify the type of file.  
   b..$error.{your data type} will help you to disply the message.
   1. <p>
   2. <input type="number" name="StudentRollNumber" ng-model="StudentRollNumber" required>
   3. <span style="color:red" ng- show="myForm.StudentRollNumber.$dirty && myForm.StudentRollNumber.$invalid">
   4. <span ng-show="myForm.StudentRollNumber.$error.required">Student Roll Number is required.</span>
   5. <span ng-show="myForm.StudentRollNumber.$error.number">Not valid number!</span>
   6. </span>
   7. </p>
3. **Required filed validation**a. Put attribute as required in HTML control.  
   b..$error.required helps you to display the required field message.
   1. <p>
   3. <input type="text" name="Student" ng-model="Student" required>
   4. <span style="color:red" ng-show="myForm.Student.$dirty && myForm.Student.$invalid">
   5. <span ng-show="myForm.Student.$error.required">Student Name is required.</span>
   6. </span>
   7. </p>
4. **Date Validation**a. Specify the type as date and  
   b. Format it will take as systems built-in format  
   c. .$error.date helps you to display the required field message.
   1. <p>
   2. Student Birth Date:<br>
   3. <input type="date" name="BirthDate" ng-model="BirthDate" required placeholder="yyyy-MM-dd">
   4. <span style="color:red" ng-show="myForm.BirthDate.$dirty && myForm.BirthDate.$invalid">
   5. <span ng-show="myForm.BirthDate.$error.required">Student Birth Date is required.</span>
   6. <span ng-show="myForm.BirthDate.$error.date">Not a Valid Date.</span>
   7. </span>
   8. </p>
5. **Email Validation**a. Specify the type as Email and  
   b..$error.email helps you to display the required field message.
   1. <input type="email" name="email" ng-model="email" required>
   2. <span style="color:red" ng-show="myForm.email.$dirty && myForm.email.$invalid">
   3. <span ng-show="myForm.email.$error.required">Email is required.</span>
   4. <span ng-show="myForm.email.$error.email">Invalid email address.</span>
   6. </span>
6. **Range Validation Max and Min**a. Specify Max or Min attribute  
   b..$error.max or .$error.min helps you to display the error message.
   1. <input type="number" name="marks" ng-model="marks" max="100" required>
   2. <span style="color:red" ng-show="myForm.marks.$dirty && myForm.marks.$invalid">
   3. <span ng-show="myForm.marks.$error.required">Email is required.</span>
   4. <span ng-show="myForm.marks.$error.number">Invalid number.</span>
   5. <span ng-show="myForm.marks.$error.max">Max Percentage is 100.</span>
   6. </span>

**For more details click on the link**

* [AngularJS Validation In MVC - Part Two](http://www.c-sharpcorner.com/article/angularjs-validation-in-mvc-part-two2/)

**Question 12: What is $rootscope and how do we use it?  
  
Answer**$rootscope provides access to the top of the scope hierarchy, a scope provides a separation between View and its Model. Every application has a $rootScope provided by AngularJS and every other scope is its child scope.  
  
Now see how to use it step by step,  
  
**Step 1**First of all you need to add an external Angular.js file to your application, "angular.min.js."  
  
For this you can go to the AngularJS official site or download my source code and then fetch it or you can click on this link to download it: [ANGULARJS](http://www.c-sharpcorner.com/downloads/762/angularjs.aspx).  
  
After downloading the external file you need to add this file to the Head section of your application.

1. <head runat="server">
2. <title></title>
3. <script src="angular.min.js"></script>
4. </head>

**Step 2**Now after adding the External JS file the first thing you need to do is to add ng-app in the <HTML> Tag otherwise your application will not run.

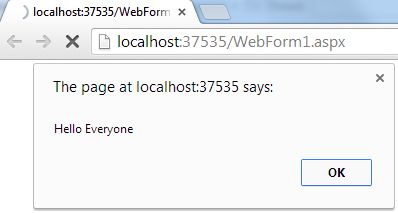
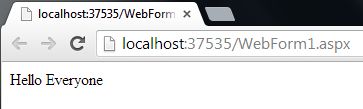
1. <html ng-app xmlns="http://www.w3.org/1999/xhtml">

Now, I will create a JavaScript function in which the $rootScope service will be initiated.

1. <script>
2. angular.module('app', []).controller('x', **function** ($scope, $rootScope) {
3. $rootScope.showAlert = "Hello Everyone";
4. });
6. angular.module('app').controller('x2', **function** ($scope, $rootScope) {
7. $scope.val = $rootScope.showAlert;
8. alert($scope.val);
9. });
10. </script>

Here, I created two angular.modules, in the first module I created a controller named "x", in this controller the "showAlert" variable is used with the $rootScope, in this a showAlert message is provided.  
  
In the second controller a variable "val" is used but it is taken under $scope, the value of rootScope is provided in this variable and then is provided in the alert.  
  
Now our work on the View is completed so we can work on the ViewModel.  
  
**Step 3**

1. <body>
2. <form id="form1" runat="server">
3. <div ng-app="app">
5. <div ng-controller="x"></div>
6. <div ng-controller="x2">{{val}}</div>
7. </div>
8. </form>
9. </body>

Here, I took a Div that is bound using the ng-app, after this two Divs are used, one of these is bound to the first controller, "x", and the second is bound to "x2"; both use the ng-controller.  
  
In the second div the "val" variable is bound so this div will display the text that is passed in the val.  
  
Now our application is ready for execution.  
  
**Output**On running the application an Alert Message will be displayed while the page is loading,  
  
  
  
When you click on the "OK" button the same message will be displayed on the webform as well.  
  
  
  
The complete code of this application is as follows,

1. <html ng-app="app" xmlns="http://www.w3.org/1999/xhtml">
2. <head runat="server">
3. <title></title>
4. <script src="angular.min.js"></script>
5. <script>
6. angular.module('app', []).controller('x', function ($scope, $rootScope) {
7. $rootScope.showAlert = "Hello Everyone";
8. });
10. angular.module('app').controller('x2', function ($scope, $rootScope) {
11. $scope.val = $rootScope.showAlert;
12. alert($scope.val);
13. });
14. </script>
15. </head>
16. <body>
17. <form id="form1" runat="server">
18. <div ng-app="app">
20. <div ng-controller="x"></div>
21. <div ng-controller="x2">{{val}}</div>
22. </div>
23. </form>
24. </body>
25. </html>

For more visit the following link,

* [$rootScope Service in AngularJS](http://www.c-sharpcorner.com/UploadFile/cd7c2e/rootscope-service-in-angularjs/)

**Question 13: Explain what is Dependency Injection in AngularJS?**  
**Answer**   
  
Dependency Injection is one of the best features of AngularJS. It is a software design pattern in which objects are passed as dependencies. It helps us to remove hard coded dependencies and makes dependencies configurable. Using Dependency Injection, we can make components maintainable, reusable and testable.  
  
**Dependency Injection is required for the following**

* Separating the process of creation and consumption of dependencies.
* It allows us to create independent development of the dependencies.
* We can change the dependencies when required.
* It allows injecting mock objects as dependencies for testing.

**AngularJS uses dependency with several types**

* Value
* Factory
* Service
* Provider
* Constant

**A simple case of dependency injection in Angular js**

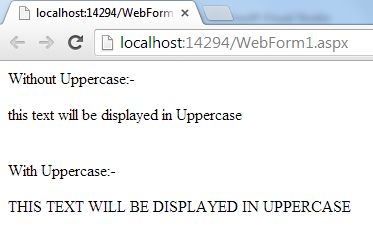
1. AppModule.controller("AppController", function($scope, $window, $log,$http)
2. {
4. });

**For more details click on the link**

* [Basic Concepts in AngularJs](http://www.c-sharpcorner.com/UploadFile/dev4634/basic-concepts-in-angular-js/)

**Question 14: Explain Convert Text To Uppercase Using AngularJS.**  
**Answer**  
  
AngularJS provides a feature for converting all the letters of text into uppercase letters. I will explain this feature by creating a sample application.  
  
First of all you need to add an external Angular.js file to your application, in other words "angular.min.js".For this you can go to the AngularJS official site. After downloading the external file you need to add this file to the Head section of your application.  
  
**The complete code of this application is as follows**

1. <html ng-app xmlns="http://www.w3.org/1999/xhtml">
3. <head runat="server">
4. <title></title>
5. <script src="angular.min.js"></script>
6. <script>
7. function x($scope) {
8. $scope.test = "this text will be displayed in Uppercase";
9. }
10. </script>
11. </head>
13. <body>
14. <form id="form1" runat="server">
15. <div ng-controller="x">
16. Without Uppercase:-
17. <p>{{test}}</p>
18. <br /> With Uppercase:-
19. <p>{{test|uppercase}}</p>
20. </div>
21. </form>
22. </body>
24. </html>

  
  
  
**For more details click on the link**

* [Convert Text To Uppercase Using AngularJS](http://www.c-sharpcorner.com/UploadFile/cd7c2e/convert-a-data-into-uppercase-letters-using-angularjs/)

**Question 15: Explain ng-repeat directive.  
  
Answer**The ng-repeat directive is the most used and very useful AngularJS Directive feature. It iterates over a collection of items and creates DOM elements. It constantly monitors the source of data to re-render a template in response to change.  
  
**Syntax of ng-repeat**

1. <table class="table table-bordered">
2. <tr ng-repeat="empin empDetails">
3. <td>{{emp.name}}</td>
4. <td>{{emp.salary}}</td>
5. </tr>
6. </table>

Here, ng-repeat directive iterates over the empDetails collection and creates a <tr> DOM element for each entry in the collection.  
  
The ng-repeat directive creates a new scope for each element of a collection.  
  
**Variables created by ng-repeat**  
AngularJS ng-repeat directive creates so many special variables in a scope created for each and every individual entry. These variables are very important to find the position of an element within a collection.  
  
**Below are the some important variables created by ng-repeat**

1. $index
2. $first
3. $middle
4. $last

**For more details click on the link**

* [AngularJS ngRepeat Directive](http://www.c-sharpcorner.com/blogs/angularjs-ngrepeat-directive)

**Question 16: What is controller in AngularJS?  
  
Answer**A controller is a set of JavaScript functions which is bound to a specified scope, the ng-controllerdirective. Angular will instantiate the new controller object, and injects the new scope as a dependency. It contains business logic for the view and avoids using controller to manipulate the DOM.   
  
controller  
  
**Controller Rules**

* We can use controller to set up the initial state of the scope object and add behavior to that object.
* We do not use controller to manipulate DOM. It should contain only business logic and can use data binding and directives for the DOM manipulation.
* We do not use controllers to format input but can use angular from controls instead of that.
* We do not use filter output but can use angular filters instead of that.
* We do not use controllers to share code or state across controllers but can use angular services instead of that.
* We do not manage the life-cycle of other components.

**Creating a Controller**

* Requires ng-controller directive.
* Add controller code to a module.
* Name your controller based on functionality.
* Controllers are named using camel case (i.e. SimpleController).
* Setup the initial state of the scope object.

**ng-Controller directive**  
ng-Controller directive is an associated controller class to the view.  
  
**How to use ng-Controller**

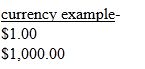
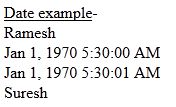
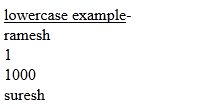
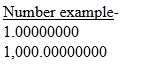
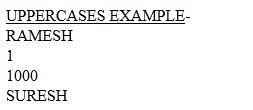
1. <Any ng-Controller=”expression”>
2. </Any>
3. <div ng-app="mainApp" ng-controller="SimpleController">
4. </div>

**For more details click on the link**

* [Understanding The Controllers - Part 4](http://www.c-sharpcorner.com/UploadFile/f2823e/understanding-the-controllers-part-4/)

**Question 17: What are the filters in AngularJS?**  
**Answer**Filters are used to modify the data and can be clubbed in expression or directives using a pipe character. A filter formats the value of an expression for display to the user. They can be used in view templates, controllers, or services, and we can easily create our own filter. Filter is a module provided by AngularJS. There are nine components of filter which are provided by AngularJS. We can write custom as well.

* currency
* date
* filter
* json
* limitTo
* lowercase
* number
* orderBy
* uppercase

**Currency** It will change all the digits to currency and "$" is the default currency.  
  
*{{ x | currency}}*  
  
**Output**  
  
  
**Date**It will change all the digits into the date according to some rules, like the default date will be "44 years 2 months 10 days" earliar and 1000 will add 1 second into it.  
  
*{{ x | date:'medium' }}*  
  
**Output** Change the 1 and 1000 into dates.  
  
  
  
**Filter**  
*{{ filter\_expression | filter : expression : comparator}}*  
  
**limitTo**It will show the values depending on the limit of an array variable that has been set.  
  
*{{ names | limitTo:2 }}*  
  
**Output  
  
Output**Here the limit is 2, so you can only see 2 values.  
  
**lowercase**It will change all the letters into lowercase as in the following:  
  
*{{ x | lowercase }}*  
  
**Output**  
  
  
**Number**It will show all the digits with 3 decimal places by default as in the following:  
  
*{{ x | number:8}}*  
  
**Output** I am using 8 here.  
  
  
  
**OrderBy***{{ orderBy\_expression | orderBy : expression : reverse}}*  
  
**uppercase**   
  
It will change all the letters to uppercase.  
 *{{ x | uppercase }}*  
  
**Output**  
  
  
**For more details click on the link**

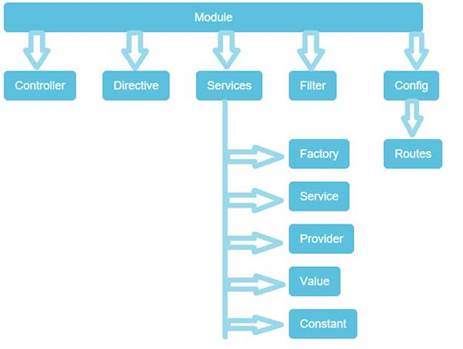
* [Built-in Filters in Angular.js](http://www.c-sharpcorner.com/UploadFile/a20beb/in-build-filters-in-angular-js/)

**Question 18: Explain Module And Controller In AngularJS.**  
**Answer**  
  
AngularJS module is nothing but a container of all angular components like controller, services, directive, filter, config etc  
  
**What is Module**  
Let me explain why module is required in AngularJS. In .NET console application there is a main method and what main method does is, it’s an entry point of application. It is the same as angular module and is an entry point. Using module we can decide how the AngularJS application should be bootstrapped.  
  
We can create a simple module using the following code.

1. var myApp = angular.module(‘myModuleApp’,[]);

In the above code myModuleApp is the module name and if this module is dependent on other modules we can inject in “[]”.  
  
**What is Controller?**  
Controller is a JavaScript constructor function which controls the data. I am not going to cover what are the types of functions in this article but let me give some brief information about constructor function. In constructor function when we call that function that function creates a new object each time.  
  
**Let’s make a controller.**

1. myApp.controller(‘myController’, function($scope)
2. {
4. });

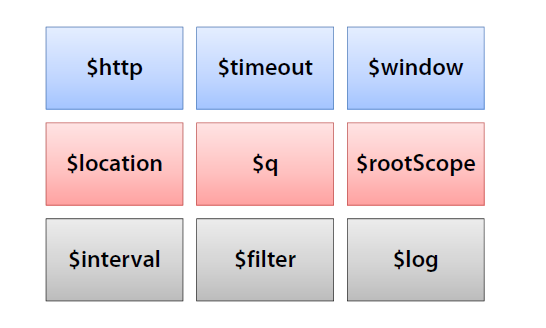
  
  
**For more details click on the link**

* [Module And Controller In AngularJS](http://www.c-sharpcorner.com/article/module-and-controller-in-angularjs/)

**Question 19: What are the services in AngularJS?**  
**Answer**Services are one of the most important concepts in AngularJS. In general services are functions that are responsible for specific tasks in an application. AngularJS services are designed based on two principles.

1. **Lazily instantiated**   
     
   Angular only instantiates a service when an application component depends on it using dependency injection for making the Angular codes robust and less error prone.
2. **Singletons**  
   Each component is dependent on a service that gets a reference to the single instance generated by the service factory.  
     
   AngularJS provides many built in services, for example, $http, $route, $window, $location and so on. Each service is responsible for a specific task, for example, $http is used to make an Ajax call to get the server data. $route defines the routing information and so on. Builtin services are always prefixed with the $ symbol.

**AngularJS internal services**  
AngularJS internally provides many services that we can use in our application. $http is one example. There are other useful services, such as $route, $window, $location and so on. Some of the commonly used services in any AngularJS applications are listed below.

* *$window* Provide a reference to a DOM object.
* *$Location* Provides reference to the browser location.
* *$timeout*Provides a reference to window.settimeout function.
* *$Log* Used for logging.
* *$sanitize* Used to avoid script injections and display raw HTML in page.
* *$Rootscope* Used for scope hierarchy manipulation.
* *$Route* Used to display browser based path in browser URL.
* *$Filter* Used for providing filter access.
* *$resource* Used to work with Restful API.
* *$document* Used to access the window. Document object.
* *$exceptionHandler* Used for handling exceptions.
* $q: Provides a promise object.****

**For more details click on the link**

* [Services in AngularJS For Beginners](http://www.c-sharpcorner.com/UploadFile/dev4634/services-in-angular-js-for-the-beginners/)

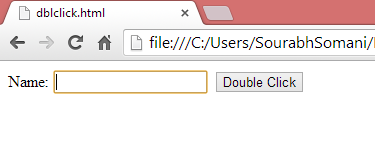
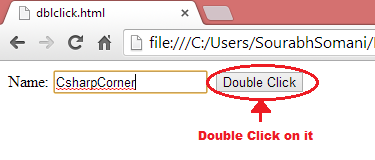
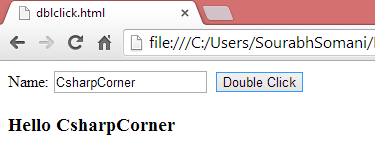
**Question 20: Explain double click event in AngularJS?**  
**Answer**ng-dblclick allows you to specify custom behavior on a double-click event of the mouse on the web page. We can use it (ng-dblclick) as an attribute of the HTML element like,

1. <ANY\_HTML\_ELEMENT ng-dblclick="{expression}">
2. ...
3. </ANY\_HTML\_ELEMENT>

Use the following procedure to create a sample of a double-click event using AngularJS.  
  
First of all you need to add an external Angular.js file to your application, for this you can go to the AngularJS official site or download my source code and then fetch it or click on this link and download it: ANGULARJS. After downloading the external file you need to add this file to the Head section of your application as in the following,  
  
**Complete Code**

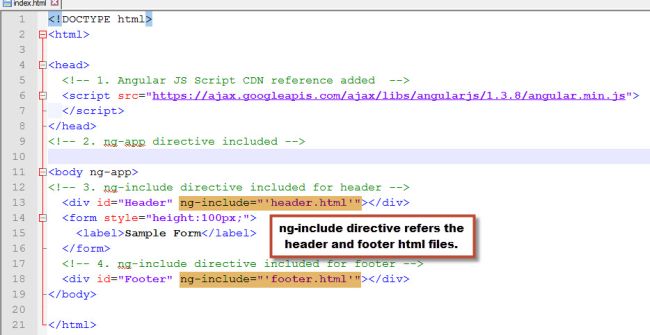
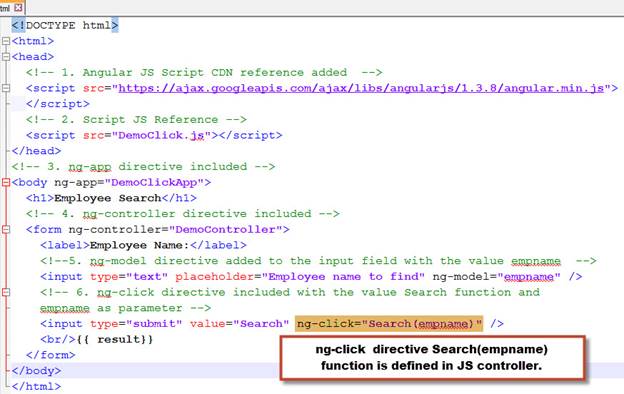
1. <!doctype html>
2. <html ng-app>
4. <head>
5. <script src="angular.min.js"></script>
6. </head>
8. <body>
9. Name:
10. <input ng-model="name" type="text" />
11. <button ng-dblclick="Msg='Hello '+name">
12. Double Click
13. </button>
14. </br>
15. <h3>
16. {{Msg}}</h3>
17. </body>
19. </html>

**Output**

* Initially when Page loads  
    
  
* Before double-click  
    
  
* After double-click  
    
  

**For more details click on the link**

* [Double Click Event in AngularJS](http://www.c-sharpcorner.com/UploadFile/75a48f/double-click-using-angularjs/)

**Question 21: Explain ng-include, Click, and Repeat directive in AngularJS.**  
**Answer**ng-include is an AngularJS directive, it is very helpful to include the various files in a main page using the ng-include attribute.  
  
For example, you have one page layout that contains a header, body, footer and so on. In that scenario, you can create various HTML files for a header and footer then you can include those in the main HTML file. Because of this implementation the page looks cleaner and the code/design is also separated.  
  
  
  
**ng-click**This is also one of the directives; you can use this in one of the scenarios like when you click on a button. If you do any operation then this will be useful.  
  
The form contains an input text box and Search button, whenever the user enters a value into a text box and clicks on the search button you need to display the user-entered value, if the user clicks on the search button without entering anything then we need to display a message.  
  
The index.html file looks as in the following.  
  
  
  
**ng-repeat**  
This directive is like a foreach loop in C#. By using this directive you can display a collection of items in a view (HTML page).  
  
You can display a list of employees using the ng-repeat directive in AngularJS.

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <!-- 1. Angular JS Script CDN reference added -->
5. <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.8/angular.min.js"></script>
6. <!-- 2. Script JS Reference -->
7. <script src="Employee.js"></script>
8. </head>
9. <!-- 3. ng-app directive included -->
10. <body ng-app="DemoRepeatApp">
11. <h1>List of Emplooyees</h1>
12. <!-- 4. ng-controller directive included -->
13. <form ng-controller="DemoController">
14. <table>
15. <thead>
16. <tr>
17. <th>Name</th>
18. <th>Designation</th>
19. <th>Location</th>
20. </tr>
21. </thead>
22. <tbody>
23. <!-- 5. ng-repeat directive included -->
24. <tr ng-repeat="emp in employees">
25. <td>{{emp.Name}}</td>
26. <td>{{emp.Designation}}</td>
27. <td>{{emp.Location}}</td>
28. </tr>
29. </tbody>
30. </table>
31. </form>
32. </body>
33. </html>

**For more details click on the link**

* [AngularJS Core Directives: Include, Click and Repeat](http://www.c-sharpcorner.com/UploadFile/2ed7ae/angularjs-core-directives-include-click-and-repeat/)

**Question 22: Explain ng-disabled Directive in AngularJS.**  
**Answer**  
  
ng- disabled directive is used to enable or disable HTML elements. Let us see this with the help of an example.  
  
**Write the following HTML mark up in the webpage.**

1. <!doctype html>
2. <html ng-app>
4. <head>
5. <title>My Angular App</title>
6. <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
7. </head>
9. <body>
10. <div ng-app="" ng-init="Switch=true">
12. <p>
13. <input type="checkbox" ng-model="Switch" />
14. </p>
15. <p>
16. <button ng-disabled="Switch">Submit</button>
17. </p>
19. </div>
20. </body>
22. </html>

In the above example we have a checkbox and a button. If the checkbox is selected the button is disabled, but if we uncheck the checkbox then the button is enabled.  
  
So let us check the output of the program.  
  
  
When we select the checkbox let us see what happens!!  
  
**For more details click on the link**

* [ng-Disabled Directive in AngularJS](http://www.c-sharpcorner.com/blogs/ngdisabled-directive-in-angularjs)
* [ng-disabled Directive of AngularJS](http://www.c-sharpcorner.com/UploadFile/cd7c2e/ng-disabled-directive-of-angularjs/)

**Question 23: Explain ng-app directive.  
  
Answer**  
  
ng-app directive is used to define AngularJS applications. We can use this to auto-bootstrap an AngularJS application. It designates the root element of AngularJS application and is generally kept near the  <body> or <html> tag. We can define any number of ng-app directives inside the HTML document but only one AngularJS application can be bootstrapped automatically (auto-bootstrapped); the other applications needs to be bootstrapped manually.  
  
**Example**

1. <div ng-app="myApp" ng-controller="myCtrl">
3. First Name :
4. <input type="text" ng-model="firstName">
5. <br />
6. Middle Name:
7. <input type="text" ng-model="middleName">
8. <br />
9. Last Name :
10. <input type="text" ng-model="lastName">
11. <br>
13. Full Name: {{firstName + " " + middleName + " " + lastName }}
15. </div>

**For more details click on the link**

* [Creating Angular App Without Using ng-app Directive](http://www.c-sharpcorner.com/UploadFile/dbd951/creating-angular-app-without-using-ng-app-directive/)

**Question 24: Why are we using AngularJS and what are the advantages of AngularJS?**  
**Answer**As we know AngularJS follows the  MVW\* pattern and it allows us  to build well-structured, testable, and maintainable front end applications.  
  
**Note** W\* means "whatever," in place of which we use C (controller) or VM (view model)  
  
**Why we are using AngularJS**

1. As we know AngularJS is based on MVC pattern; it helps us to organize our web apps or web application properly.
2. It helps to make responsive and well organized web applications that are more expansive and readable.
3. It follows two way data binding. Two way data binding helps us so that any changes in model will be updated view and vice-versa without any manipulation on DOM or events.
4. AngularJS supports create your own directive that makes reusable components to be used according to your requirement. It is also abstract DOM manipulation logic.
5. It supports services and dependency injection which we can easily inject in our controller and provides some utility code as per our requirement.

**Advantages of AngularJS**

1. AngularJS has code reusability that allows us to write code & reuse it as required as Custom directive.
2. AngularJS supports powerful data binding; it is two way data binding with the help of HTML & scope.
3. AngularJS is easily customizable as per our requirement. Here we can create own custom components like directive and services.
4. AngularJS has good support over the internet and over time it has new changes available for developers. It also supports IE, Opera, Safari, and Chrome.
5. AngularJS has inbuilt form validation & template with all old plain html.
6. AngularJS has easily testable Unit testing, it  doesn't need to load all the app, just loading that specific module is enough to start unit testing.

**For more details click on the link**

* [Overview Of AngularJS: Part 1](http://www.c-sharpcorner.com/UploadFile/d63fc5/overview-of-angularjs/)

**Question 25: What is Representational State Transfer(REST) in AngularJS.  
  
Answer**REST is a style of API that operates over HTTP requests. The requested URL identifies the data to be operated on, and the HTTP method identifies the operation that is to be performed. REST is a style of API rather than a formal specification, and there is a lot of debate and disagreement about what is and isn’t RESTful, a term used to indicate an API that follows the REST style. AngularJS is pretty flexible about how RESTful web services are consumed. You should use the services that I describe in this article when you are performing data operations on a RESTful API. You may initially prefer to use the $http service to make Ajax requests, especially if you are coming from a jQuery background. To that end, I describe the use of $http at the start of the article, before explaining its limitations when used with REST and the advantages of using the $resource service as an alternative. For this, we first need to create a RESTful web API.  
  
A REST web service is said to be RESTful when it adheres to the following constrants:

* It’s URL-based (e.g., <http://www.micbutton.com/rs/blogPost>).
* It uses an Internet media type such as JSON for data interchange.
* It uses standard HTTP methods (GET, PUT, POST, DELETE).

HTTP methods have a particular purpose when used with REST services. The following is the standard way that HTTP methods should be used with REST services,  
  
**POST should be used to,**

1. Create a new resource.
2. Retrieve a list of resources when a large amount of request data is required to be passed to the service.
3. PUT should be used to update a resource.
4. GET should be used to retrieve a resource or a list of resources.
5. DELETE should be used to delete a resource.

**For doing this, we first create a model class with the below mention members**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Required** |
| name | String | Yes |
| category | String | Yes |
| price | number | Yes |

**For more details click on the link**

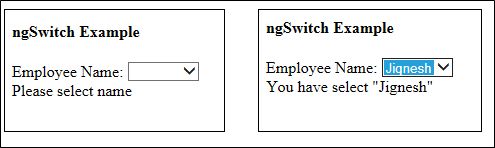
* [AngularJS From Beginning: REST API - Part Eight](http://www.c-sharpcorner.com/article/angularjs-from-beginning-rest-api-with-http-part-eight/)

**Question 26: Explain ng-Switch Directive in AngularJS.**  
**Answer**This directive is used to swap DOM structure conditionally in our template based on a scope expression. ngSwitchWhen or ngSwitchDefault directives are used to show and hide the element within ngSwitch directive. We can show or hide the element inside this directive and are required to place a "when" attribute per element. The "when" attribute is used to inform the ngSwitch directive which element is to display based on expression, if the matching expression is found, the element is displayed, else it is hidden.  
  
**Example**  
**HTML**

1. <h4>ngSwitch Example</h4>
2. <div ng-controller="HelloController">
3. Employee Name:
5. <select ng-model="selection" ng-options="name for name in names"></select>
6. <div ng-switch on="selection">
7. <div ng-switch-when="Tejas">You have select "Tejas"</div>
8. <div ng-switch-when="Rakesh">You have select "Rakesh"</div>
9. <div ng-switch-when="Jignesh">You have select "Jignesh"</div>
10. <div ng-switch-default>Please select name</div>
11. </div>
12. </div>

**Controller**

1. var app = angular.module("app", []);
2. app.controller("HelloController", function($scope)
3. {
4. $scope.names = ['Tejas', 'Jignesh', 'Rakesh'];
5. });

**Output**  
  
  
**For more details click on the link**

* [ngIf, ngSwitch and ngShow Directives In AngularJS](http://www.c-sharpcorner.com/UploadFile/ff2f08/directives-in-angularjs121/)

**Question 27: Why we use $http service or ajax request in AngualrJS?**  
**Answer**Ajax is the foundation of the modern web application, and you will use the services that I describe in this article every time that you need to communicate with a server without causing the browser to load new content and, in doing so, dump your AngularJS application. That said, if you are consuming data from a RESTful API, then you should use the $resource service. I will describe REST and $resource in the next article, but the short version is that $resource provides a higher-level API that is built on the services I describe in this article and makes it easier to perform common data operations.  
  
**Making Ajax Requests**The $http service is used to make and process Ajax requests, which are standard HTTP requests that are performed asynchronously.  
  
The first—and most common—is to use one of the convenience methods that the service defines, which I have described in the below table and which allows you to make requests using the most commonly needed HTTP methods.

|  |  |
| --- | --- |
| **Name** | **Descriptions** |
| get(url, config) | Performs a GET request for the specified URL. |
| post(url, data, config) | Performs a POST request to the specified URL to submit the specified data. |
| delete(url, config) | Performs a DELETE request to the specified URL. |
| put(url, data, config) | Performs a PUT request with the specified data and URL. |
| patch(url, data, config) | Performs a PATCH request with the specified data and URL. |
| head(url, config) | Performs a HEAD request to the specified URL. |
| jsonp(url, config) | Performs a GET request to obtain a fragment of JavaScript code that is then executed. JSONP, which stands for JSON with Padding, is a way of working around the limitations that browsers apply to where JavaScript code can be loaded from. |

The other way to make an Ajax request is to treat the $http service object as a function and pass in a configuration object.  
  
**Configuring Ajax Requests**The methods defined by the $http service all accept an optional argument of an object containing configuration settings. For most applications, the default configuration used for Ajax requests will be fine, but you can adjust the way the requests are made by defining properties on the configuration object corresponding to the below table.

|  |  |
| --- | --- |
| **Name** | **Descriptions** |
| data | Sets the data sent to the server. If you set this to an object, AngularJS will serialize it to the JSON format. |
| headers | Used to set request headers. Set headers to an object with properties whose names and values correspond to the headers and values you want to add to the request. |
| method | Sets the HTTP method used for the request. |
| params | Used to set the URL parameters. Set params to an object whose property names and values correspond to the parameters you want to include. |
| timeout | Specifies the number of milliseconds before the request expires. transformRequest Used to manipulate the request before it is sent to the server. |
| transformResponse | Used to manipulate the response when it arrives from the server |
| url | Sets the URL for the request. |
| withCredentials | When set to true, the withCredentials option on the underlying browser request object is enabled, which includes authentication cookies in the request. |

The most interesting configuration feature is the ability to transform the request and response through the aptly named transformRequest and transformResponse properties. AngularJS defines two built-in transformations; outgoing data is serialized into JSON, and incoming JSON data is parsed into JavaScript objects.  
  
**Html file code**

1. <!DOCTYPE html>
2. <html ng-app="TestApp">
3. <head>
4. <title>AngularJS AJax </title>
5. <script src="angular.js"></script>
6. <link href="../../RefStyle/bootstrap.min.css" rel="stylesheet" />
7. <script src="app.js"></script>
8. <script src="ajax\_config.js"></script>
9. </head>
10. <body ng-controller="ajaxController">
11. <div class="panel panel-default">
12. <div class="panel-body">
13. <table class="table table-striped table-bordered">
14. <thead>
15. <tr>
16. <th>Name</th>
17. <th>Category</th>
18. <th>Price</th>
19. </tr>
20. </thead>
21. <tbody>
22. <tr ng-hide="products.length">
23. <td colspan="4" class="text-center">No Data</td>
24. </tr>
25. <tr ng-repeat="item in products">
26. <td>{{item.Category}}</td>
27. <td>{{item.Book}}</td>
28. <td>{{item.Publishers}}</td>
29. <td>{{item.price | currency}}</td>
30. </tr>
31. </tbody>
32. </table>
33. <p>
34. <button class="btn btn-primary" ng-click="loadData()">
35. Load Data
36. </button>
37. <button class="btn btn-primary" ng-click="loadXMLData()">
38. Load Data (XML)
39. </button>
40. </p>
41. </div>
42. </div>
43. </body>
44. </html>

**AngularJS file code**

1. testApp.controller("ajaxController", function($scope, $http)
2. {
4. $scope.loadData = function()
5. {
6. $http.get("data.json").success(function(data)
7. {
8. $scope.products = data;
9. });
10. }
12. $scope.loadXMLData = function()
13. {
14. var config =
15. {
16. transformResponse: function(data, headers)
17. {
18. if ((headers("content-type") == "application/xml" || headers("content-type") == "text/xml") && angular.isString(data)) {
19. products = [];
20. var productElems = angular.element(data.trim()).find("product");
21. for (var i = 0; i < productElems.length; i++)
22. {
23. var product = productElems.eq(i);
24. products.push
25. ({
26. Category: product.attr("Category"),
27. Book: product.attr("Book"),
28. Publishers: product.attr("Publishers"),
29. price: product.attr("price")
30. });
31. }
32. return products;
33. } else
34. {
35. return data;
36. }
37. }
38. }
39. $http.get("data.xml", config).success(function(data)
40. {
41. $scope.products = data;
42. });
43. }
44. });

**For more details click on the link**

* [AngularJS From Beginning: Http Request or Ajax - Part Seven](http://www.c-sharpcorner.com/article/angularjs-from-beginning-http-request-or-ajax-part-seven/)

**Question 28: Why to use AngularJS Global Object services?**  
**Answer**The main reason that AngularJS includes these services is to make testing easier, but an important facet of unit testing is the need to isolate a small piece of code and test its behavior without testing the components it depends on—in essence, creating a focused test. The DOM API exposes functionality through global objects such as document and window.   
  
These objects make it hard to isolate code for unit testing without also testing the way that the browser implements its global objects. Using services such as $document allows AngularJS code to be written without directly using the DOM API global objects and allows the use of AngularJS testing services to configure specific test scenarios.  
  
The followings are the services that expose DOM API features.

|  |  |
| --- | --- |
| **Name** | **Descriptions** |
| $anchorScroll | Scrolls the browser window to a specified anchor |
| $document | Provides a jqLite object that contains the DOM window.document object |
| $interval | Provides an enhanced wrapper around the window.setInterval function |
| $location | Provides access to the URL |
| $log | Provides a wrapper around the console object |
| $timeout | Provides an enhanced wrapper around the window.setITimeout function |
| $window | Provides a reference to the DOM window object |

**For more details click on the link**

* [AngularJS From Beginning: Global Object Service - Part Six](http://www.c-sharpcorner.com/article/angularjs-from-beginning-global-object-service-part-six/)

**Question 29: When and Why to use and create Services?**  
**Answer**   
  
Services are used to encapsulate functionality that we need to reuse in an application but don’t fit clearly into Model-View-Controller pattern as we discussed in the article. Services are mainly used for the purpose of cross cutting concerns. The AngularJS Module defines three methods for defining services : factory, service and provider. The result of using these methods is the same – a service object that provides functionality that can be used throughout the AngularJS application – but the way that the service object is created and managed by each method is slightly different. Below I mentioned the built in services of AngularJS.

|  |  |
| --- | --- |
| **Name** | **Descriptions** |
| $anchorScroll | Scrolls the browser window to a specified anchor |
| $animate | Animates the content transitions. |
| $compile | Processes an HTML fragment to create a function that can be used to generate content. |
| $controller | A wrapper around the $injector service that instantiates controllers |
| $document | Provides a jqLite objects that contains the DOM window.documentobject. |
| $exceptionHandler | Handles exceptions that arise in the application. |
| $filter | Provides access to filters |
| $http | Creates and manages Ajax requests |
| $injector | Creates instances of AngularJS components |
| $interpolate | Processes a string that contains binding expressions to create a function that can be used to generate content. |
| $interval | Provides an enhanced wrapper around the window.setInterval function. |
| $location | Provides a wrapper around the browser location object. |
| $log | Provides a wrapper around the global console object. |
| $parse | Processes an expression to create a function that can be used togenerate content. |
| $provide | Implements many of the methods that are exposed by Module. |
| $q | Provides deferred objects/promises. |
| $resource | Provides support for working with RESTful APIs. |
| $rootElement | Provides access to the root element in the DOM. |
| $rootScope | Provides access to the top of the scope hierarchy |
| $route | Provides support for changing view content based on the browser’sURL path. |
| $routeParams | Provides information about URL routes. |
| $sanitize | Replaces dangerous HTML characters with their display-safecounterparts. |
| $swipe | Recognizes swipe gestures. |
| $timeout | Provides an enhanced wrapper around the window.setITimeout function. |
| $window | Provides a reference to the DOM window object. |

I will discuss about this built in service of AngularJS in a later article. The main focus of this article is what are the different ways to create custom services as per our requirement in AngularJS.   
  
**Using Factory method**The simplest method of defining a service is to use the Module.factory method, passing an argument, the name of the service and a factory function that returns the service objects. For doing this, we create three files as follows,  
  
**ServiceApp.Js**

1. var serviceApp = angular.module('ServiceApp', []);
3. serviceApp.factory("logService", function()
4. {
5. var messageCount = 0;
6. return
7. {
8. log: function(msg)
9. {
10. console.log("(LOG + " + messageCount++ + ") " + msg);
11. }
12. };
13. });

In the above file, I first create an angular module named serviceApp for defining the factory service which creates log message on execution.  
  
**App.js**

1. var testApp = angular.module('TestApp', ['ServiceApp']);

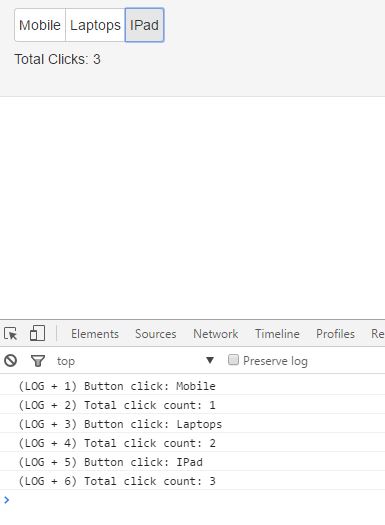
Now, I define another angualr module named testApp in which I inject the ServiceApp module. This testApp module will be used from html page for controller.  
  
**Factory.html**

2. <!DOCTYPE html>
3. <html ng-app="TestApp">
4. <head>
5. <title>AngularJS Factory</title>
6. <script src="angular.js"></script>
7. <link href="../../RefStyle/bootstrap.min.css" rel="stylesheet" />
8. <script src="serviceApp.js"></script>
9. <script src="app.js"></script>
10. <script src="Factory.js"></script>
11. </head>
12. <body ng-controller="FactoryController">
13. <div class="well">
14. <div class="btn-group" tri-button counter="data.totalClicks" source="data.device">
15. <button class="btn btn-default" ng-repeat="item in data.device">
16. {{item}}
17. </button>
18. </div>
19. <h5>Total Clicks: {{data.totalClicks}}</h5>
20. </div>
21. </body>
22. </html>

**Factory.js**

1. testApp.controller('FactoryController', function($scope, logService)
2. {
3. $scope.data =
4. {
5. device: ["Mobile", "Laptops", "IPad"],
6. totalClicks: 0
7. };
9. $scope.$watch('data.totalClicks', function(newVal)
10. {
11. logService.log("Total click count: " + newVal);
12. });
14. });


18. testApp.directive("triButton", function(logService)
20. {
21. return
22. {
23. scope:
24. {
25. counter: "=counter"
26. },
27. link: function(scope, element, attrs)
28. {
29. element.on("click", function(event)
30. {
31. logService.log("Button click: " + event.target.innerText);
32. scope.$apply(function()\
33. {
34. scope.counter++;
35. });
36. });
37. }
38. }
39. });

The output is as follows,  
  
  
  
**For more details click on the link**

* [Learn AngularJS From Beginning: Service - Part Three](http://www.c-sharpcorner.com/article/learn-angularjs-from-beginning-service-part-three/)

**Question 30: Explain Provider Method in AngularJS.**  
**Answer**The Module.provider method allows you to take more control over the way that a service object is created or configured. The arguments to the provider method are the name of the service that is being defined and a factory function. The factory function is required to return a provider object that defines a method called $get, which in turn is required to return the service object. When the service is required, AngularJS calls the factory method to get the provider object and then calls the $get method to get the service object. Using the provider method doesn’t change the way that services are consumed, which means that I don’t need to make any changes to the controller or directive in the example.  
  
The advantage of using the provider method is that you can add functionality to the provider method that can be used to configure the service object.  
  
To demonstrate this process, I again change the serviceapp.js file as below,

1. var serviceApp = angular.module('ServiceApp', []);
3. serviceApp.provider("logService", function()
4. {
5. var counter = true;
6. var debug = true;
7. return
8. {
9. messageCounterEnabled: function(setting)
10. {
11. if (angular.isDefined(setting))
12. {
13. counter = setting;
14. return this;
15. } else
16. {
17. return counter;
18. }
19. },
20. debugEnabled: function(setting)
21. {
22. if (angular.isDefined(setting))
23. {
24. debug = setting;
25. return this;
26. } else
27. {
28. return debug;
29. }
30. },
31. $get: function()
32. {
33. return
34. {
35. messageCount: 0,
36. log: function(msg)
37. {
38. if (debug)
39. {
40. console.log("(LOG" + (counter ? " + " + this.messageCount++ + ") " : ") ") + msg);
41. }
42. }
43. };
44. }
45. }
46. });

**For more details click on the link**

* [Learn AngularJS From Beginning: Service - Part Three](http://www.c-sharpcorner.com/article/learn-angularjs-from-beginning-service-part-three/)

**Question 31: What is event handling in AngularJS?**  
**Answer**When we want to create advanced AngularJS applications such as User Interaction Forms, then we need to handle DOM events like mouse clicks, moves, keyboard presses, change events and so on. AngularJS has a simple model for how to add event listeners. We can attach an event listener to an HTML element using one of the following AngularJS event listener directives.

* ng-click
* ng-dbl-click
* ng-mousedown
* ng-mouseup
* ng-mouseenter
* ng-mouseleave
* ng-mousemove
* ng-mouseover
* ng-keydown
* ng-keyup
* ng-keypress
* ng-change

Here is a simple AngularJS event listener directive example,

1. @{
2. Layout = null;
3. }

6. <!DOCTYPE html>
7. <html>
8. <head>
9. <meta name="viewport" content="width=device-width" />
10. <title>Acgular Event</title>
11. <script src="~/Scripts/angular.js"></script>
12. <script>
13. angular.module("myapp", [])
14. .controller("Controller1", function ($scope) {
15. $scope.myData = {};
16. $scope.myData.dvClick = function () {
17. alert("Div clicked");
18. }
19. });
21. </script>
22. </head>
23. <body ng-app="myapp">
24. <div ng-controller="Controller1">
25. <div ng-click="myData.dvClick()">Click here</div>
26. </div>
27. </body>
28. </html>

When we click the text within the div, the myData.dvClick() function will be called. As you can see in the controller function, the myData object has a dvClick() function added to it. The event listener functions called are functions added to the $scope object by the controller function.  
  
**For more details click on the link**

* [AngularJS Event Handling](http://www.c-sharpcorner.com/UploadFile/1d3119/angular-js-event-handling/)

**Question 32: What are the top reasons why developers choose AngularJS?  
  
Answer**  
  
AngularJS, an Open Source web application framework by Google, is widely used in building highly robust and scalable Single Page Applications (SPA). Single Page Applications are websites or web applications that encompass a single web page, rendering a seamless and immersive user experience. The framework is written in JavaScript, and allows using HTML as template language. It helps build rich and intuitive web applications, and also provides web developers the option to build client-side applications.  
  
**High Performance**  
  
AngularJS is a popular choice among web developers because of  ease of use and maintenance, intuitive features, robustness, and the efficiency to build new features. It is obvious that when a problem arises, developers are not ready to spend hours debugging it. At the same time, they should be able to make minor changes with much ease. AngularJS gives you the ease of maintenance.   
  
**Effective Handling of Dependencies**  
  
AngularJS does dependency injection extremely well. For Single Page Applications, Angular makes it extremely easy to organize things like dynamic loading and dependencies, and use them as required without worrying about “Did I spin up an instance?” or “What namespace does it live in?” Simply mention what you need, and Angular will get it for you and also manage the entire life-cycle of the objects.  
  
For testing, the framework allows you to segregate the app into logical modules that may have dependencies on each other, but are separately initialized. This helps to take a tactical approach towards testing as it provides only the modules that you need. Now, since the dependencies are injected, you can have an existing service like Angular $HTTP and swap it easily with $httpBackend mock for effective testing.  
  
**DOM has Markup in AngularJS**  
In most client-side JavaScript frameworks, the temples operate in something like this way,

* **Template with markup -> framework template engine -> HTML -> DOM**However, in AngularJS, the markup is directly put into the HTML document and flow looks something like this,
* **HTML with Angular markup -> DOM -> Angular Template Engine**The framework evaluates the markup only when HTML has been loaded into DOM.

This has three major benefits – simplicity, integration with existing apps, and extensibility. You can work with AngularJS in basic HTML documents from a local file system. Additionally, it allows you to build custom attributes and elements that can extend the basic HTML vocabulary.  
  
**For more visit the following link**

* [Top Reasons Why Web Developers Choose AngularJS](http://www.c-sharpcorner.com/article/top-reasons-why-web-developers-choose-angularjs/)

**Question 33: Explain $routeProvider in AngularJS?  
  
Answer**The $routeProvider is used to set the configuration of urls and map them with the corresponding html page or ng-template and also attach a controller. Routing in AngularJS is taken care of by a service provide that is called $routeProvider. Routes for templates and urls in Angular are declared via the$routeProvider, that is the provider of the $route service. This service makes it easy to wire together controllers, view templates, and the current URL location in the browser.  
  
We can use config() method of “myApp” module to configure $routeProvider. The when method of$routeProvideris used to bind the url with a template. This method takes a url(i.e. “/viewDelhi”) that will map with a template (i.e. delhi.htm) using the templateUrl parameter. The when method also binds a controller for templates using the controller parameter (i.e. controller: 'AddDelhi'), otherwise the method is used to set the default view.  
  
**Example**

1. mainApp.config(['$routeProvider', function($routeProvider)
2. {
3. $routeProvider.
4. when('/viewDelhi',
5. {
6. templateUrl: 'delhi',
7. controller: 'AddDelhi'
8. }).
9. when('/viewMumbai',
10. {
11. templateUrl: 'mumbai',
12. controller: 'AddMumbai'
13. }).
14. when('/viewJaipur',
15. {
16. templateUrl: 'jaipur',
17. controller: 'AddJaipur'
18. }).
19. otherwise
20. ({
21. redirectTo: '/viewDelhi'
22. });
23. }]);

**For more details click on the link**

* AngularJS View And Routing

**Question 34: What are the attributes can be used during creation of a new AngularJS Directives?**  
**Answer**The following attributes can be used during creation of a new AngularJS Directives,

1. **Restrict**  
   The restrict attribute is how AngularJS triggers the directive inside a template. The default value of the restrict option is “A”. The value of “A” causes the directives to be triggered on the attribute name. Other than “A”, restrict option has “E” (only match element name), “C” (only match class name) and “M” (only match the comment name) or any combination among four options.
2. **TemplateUrl**  
   The templateUrl attribute tells the AngularJS HTML compiler to replace custom directive inside a template with HTML content located inside a separate file. The link-Menu (say, our custom directive name) attribute will be replaced with the content of our original menu template file.
3. **Template**Specify an inline template as a string. Not used if you’re specifying your template as a URL.
4. **Replace**If true, replace the current element. If false or unspecified, append this directive to the current element.
5. **Transclude**Lets you move the original children of a directive to a location inside the new template.
6. **Scope**Create a new scope for this directive rather than inheriting the parent scope.
7. **Controller**Create a controller which publishes an API for communicating across directives.
8. **Require**Require that another directive be present for this directive to function correctly.
9. **Link**Programmatically modify resulting DOM element instances, add event listeners, and set up data binding.
10. **Compile**  
    Programmatically modify the DOM template for features across copies of a directive, as when used in other directives. Your compile function can also return link functions to modify the resulting element instances.

**For more details click on the link**

* [AngularJS From Beginning: Directive - Part Four](http://www.c-sharpcorner.com/article/angularjs-from-beginning-directive-part-four/)

**Question 35: What are the different types of Directives in AngularJS?  
  
Answer**Directives are one of the most important components of AngularJS application. They are extended HTML attributes. In other words, directives are something that introduces new syntax. They are markers on the DOM element which provides some special behavior to DOM elements and tell AngularJS's HTML compiler to attach.   
  
Their are many built in directives such as ng-model, ng-repeat, ng-show, ng-bind etc. All these directives provide special behavior to DOM elements. For example, ng-show directive conditionally shows an element, ng-click directive adds click events to the element; ng-app directive initializes an AngularJS application, etc.  
  
**Types of Directives**  
Type of directive determines how they are used. We can implement directives in the following ways,

* *Attribute directives* Directive is active when matching attribute is found.  
    
  **Example**
  1. <input type="text" numeric />
* *Element directives* Directive is active when matching element is found.  
    
  **Example**
  1. <numeric-Textbox id = "txtAge" />
* *Component directives* Directive is active when matching component is found.  
    
  **Example**
  1. <!-- directive: numeric-Textbox exp -->
* *CSS class directives* Directive is active when matching CSS style is found.  
    
  **Example**
  1. <input type="text" class=" numeric "/>

**For more details click on the link**

* [Directives In AngularJS](http://www.c-sharpcorner.com/UploadFile/ff2f08/directives-in-angularjs/)

**Question 36: What are compile & link options in Custom Directives?**  
**Answer**Understanding the compile vs. link option is one of the more advanced topics across AngularJS, and it gives us a feel for how Angular actually works. Out of both the functions the link function is used very often. Basically both are mutually exclusive; i.e,. if both the options are set then compile will overwrite the link functions defined. The concept of compile and link comes from C language, where you first compile the code and then link it to actually execute it. The process is very much similar in AngularJS as well.  
  
**Compile**It traverses the DOM and collects all of the directives and deals with transforming the template DOM. The result is a linking function.  
  
**Link**The link function deals with linking scope to the DOM.  
  
**Using Code for Compile**  
  
While defining a custom directive we have the option to define a link against which either we can define a function or we have the option to assign an object which will have pre and post function.  
  
If compile is defined as defined below then it will override the link function as shown in below example.  
  
  
  
**Using Code for Pre & Post Link**  
  
While defining a custom directive we have the option called “link” against which either we can define a single function or we have the option to assign an object in which we can define further two functions i.e. Pre-link and Post- link functions.  
  
If only a single function is defined against link option that will be same as Post link function.  
  
Both Pre and Post link function have the same syntax as defined below but the only difference is the order in which they get executed.  
  
Link example,  
  
  
  
In the above example we are defining a function against link option which will get executed before linking scope and the template.  
  
**For more details click on the link**

* [Link Function In AngularJS](http://www.c-sharpcorner.com/article/link-function-in-angularjs/)

**Question 37: Explain what is injector in AngularJS?**  
**Answer**The $injector service is responsible for determining the dependencies that a function declares and resolving those dependencies. The below table lists the methods supported by the $injector service.

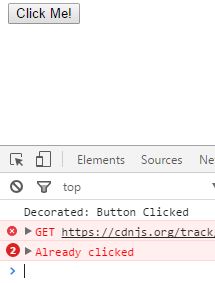
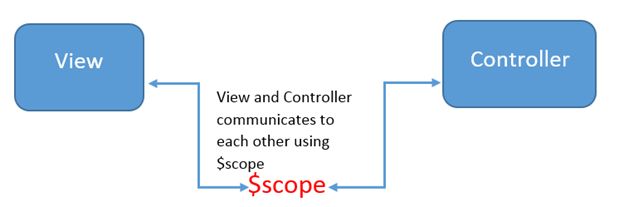
|  |  |
| --- | --- |
| **Name** | **Descriptions** |
| annotate(fn) | Gets the arguments for the specified function, including those that do not correspond to services |
| get(name) | Gets the service object for the specified service name |
| has(name) | Returns true if a service exists for the specified name |
| invoke(fn, self, locals) | Invoked the specified function, using the specified value for this and the specified non-service argument values. |

The $injector service is right at the core of the AngularJS library, and there is rarely a need to work directly with it, but it can be useful for understanding and customizing how AngularJS works. However, these are the kind of customizations that should be considered carefully and tested thoroughly.  
  
**Getting the $injector Service from the Root Element**The $rootElement service provides access to the HTML element to which the ng-app directive is applied and which is the root of the AngularJS application. The $rootElement service is presented as a jqLite object, which means you can use jqLite to locate elements or modify the DOM using the jqLite methods I described in Chapter 15. Of interest in this chapter, the $rootElement service object has an additional method called injector, which returns the $injector service object. You can see how I replaced the dependency on the $injector service with the $rootElement service in the below example.  
  
**Index.html**

1. <!DOCTYPE html>
2. <html xmlns="http://www.w3.org/1999/xhtml" ng-app="TestApp">
4. <head>
5. <title>Angular Injection</title>
6. <script src="angular.js"></script>
7. <script src="app.js"></script>
8. <script src="Index.js"></script>
10. </head>
12. <body ng-controller="indexController">
13. <div class="well">
14. <button class="btn btn-primary" ng-click="handleClick()">Click Me!</button>
15. </div>
16. </body>
18. </html>

**Index.js**

1. testApp.controller("indexController", function($scope, $log, $rootElement)
3. var counter = 0;
4. var logClick = function($log, $exceptionHandler, message)
5. {
6. if (counter == 0)
7. {
8. $log.log(message);
9. counter++;
10. } else {
11. $exceptionHandler("Already clicked");
12. }
13. }
14. $scope.handleClick = function()
15. {
16. var localVars =
17. {
18. message: "Button Clicked"
19. };
20. $rootElement.injector().invoke(logClick, null, localVars);
21. };
22. });

The output of the code is as below,  
  
  
  
**Question 38: Mention what are the characteristics of “Scope”?**  
**Answer**$scope is a glue between the View and the Controller. It connects a Controller with the View,  
  
 

1. $scope serves as the glue between the Controller and the View.
2. The $scope is the connection between the HTML and the View.
3. The View and the model both have access to the $scope.
4. In the context of MVC, $scope can be seen as the ViewModel.
5. $scope provides the execution context for the DOM and the expression.
6. $scope provides an execution context in which the DOM element is bound.
7. $scope is the source of the truth.
8. $scope is modified when the View changes and the View is modified when $the scope changes its value.
9. The $scope object is a plain JavaScript object. We can add and remove a property as required.
10. $scope holds data and functions from the Controller that should be displayed and executed in the View.
11. The $rootScope is the eventual parent of all the $scope.
12. $rootScope is the top-most scope in a DOM element with the ng-app directive.
13. In angular all the $scope are created with prototypal inheritance.
14. $scope has access to their parent scope.
15. $scope contains data and the functionality to be used to render the View.
16. For each Controller created a new $scope is created.
17. It is ideal to contain the application logic in the Controller and the data in the $scope of the Controller.
18. When $the scope object is not needed in the View, the scope will be cleaned up and destroyed.
19. Directives do not have their own scope but with some exceptions ng-controller and ng-repeat do.
20. When angular starts running all the $scope are attached to the View.
21. $scope passes data and behavior to the View.

**Question 39: Give the differences between AngularJS and Backbone and Knockout?  
  
Answer**Comparison with Backbone.js and Knockout.js,

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison** | **AngularJs** | **Backbone.js** | **Knockout.js** |
| File Size | ~142 KB total (compressed and minified) | ~ 7.3 KB total (gzip / minified) | ~21 KB total (gzip / minified) |
| Version & Licence | V1.4.2 & MIT (Open-source) | V1.2.1 & MIT (Open-source) | V3.3.0 & MIT (Open-source) |
| Dependencies | No Dependencies | Dependends on underscore.js and jQuery | No Dependencies |
| Data Binding | It supports full data binding and provides options for creating custom data bindings | Does not support data binding by default but does using plugins for data bindings | It fully supports data binding and can bind many attributes. It provides options for creating custom data bindings |
| Routing | It supports routing feature and it's very simple | It supports routing features and it's very simple | Does not support routing by defualt but it is available with some thrid-party libraries |
| Views | Uses HTML as the templating language | Does not have templates by default but we can add them easily by a thrid-party template like underscore.js and handlebars | It uses HTML as the templating language |
| Testing | Can support Test Driven Development (TDD) | Does not support testing by defualt but we can use some thrid-party tester like Jasmine and Sinon.JS | Does not support testing by defualt but we can use some thrid-party tester like Jasmine and Sinon.JS |
| Data | Does not support jQuery but we can use Angular's $http | Can support jQuery's $.ajax and is very easy to understand | It can support jQuery's $.ajax and knockout mapping |
| Design Pattern | Can support the MVC and MVVM design patterns | It can support MVP design pattern | It can support the MVVM design pattern |
| Browser | Can support IE 9, IE 10 and IE 11 | It dependends on jQuery supporting browsers like IE 6+, Chrome, Firefox, Safari 5.1+ and Opera | It can support all major browsers like IE 6+, Firefox 3.5+, Chrome, Opera and Safari |
| Third-party Integration | Does not support third-party integration | Does not support third-party integration | It supports third-party integration |
| Documentation | It has available documentation and community | To my knowledge there is no documentation | It has available documentation and community |

**For more details click on the link**

* [AngularJS Comparison with Backbone, Knockout - Part 3](http://www.c-sharpcorner.com/UploadFile/f2823e/angularjs-comparison-with-backbone-knockout-part-3/)

**Question 40: What is the difference between AngularJS and jQuery?**  
**Answer**   
  
jQuery and AngularJS have some common features like Unit test runner, animation support, AJAX/JSONP but they also have some differences.

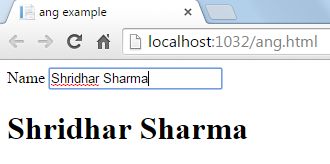
* AngularJS came with RESTful API whereas we don't have that in jQuery.
* AngularJS supports the MVC pattern whereas jQuery doesn't.
* AngularJS has the feature called Two Way Data Binding whereas we don't have that in jQuery.
* Deep Linking Routing is supported by AngularJS whereas jQuery doesn't.
* The AngularJS file size is quite heavier than that of the jQuery file size.  
  We can prefer AngularJS only if we are developing a heavy web application.

**jQuery Example**

1. <!DOCTYPE html>
2. <html xmlns="http://www.w3.org/1999/xhtml">
4. <head>
5. <title>jquery example</title>
6. <script src="js/jquery-1.11.2.js"></script>
7. </head>
9. <body>
10. <script type="text/javascript">
11. $(function() {
12. $(document).keyup(function() {
13. var name = $("#txt").val();
14. $("#lbl").html(name);
15. });
16. });
17. </script>
18. <div>
19. Name <input type="text" id="txt" placeholder="please enter name" />
20. <h1><b id="lbl"></b></h1>
21. </div>
22. </body>
24. </html>

  
  
**AngularJS Example**

1. <!DOCTYPE html>
2. <html data-ng-app>
4. <head>
5. <title>ang example</title>
6. <script src="js/angular.min.js"></script>
7. </head>
9. <body>
10. <div>
11. Name <input type="text" placeholder="please enter name" data-ng-model="name" />
12. <h1>{{name}}</h1>
13. </div>
14. </body>
16. </html>



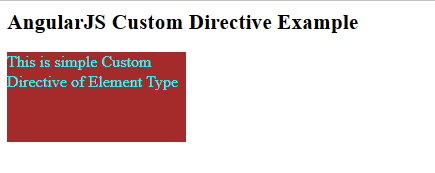
**For more details click on the link**

* [AngularJS Vs jQuery](http://www.c-sharpcorner.com/UploadFile/092589/angularjs-vs-jquery/)

**Question 41: What are the custom directives in AngularJS?**  
**Answer**In AngularJS we can create the custom directive for the following types of elements.  
  
**Element directives**Directive activates when a matching element is encountered. Restrict mode is defined by “E”.   
  
**Example** <ng-directives></ng-directives>  
  
**Attribute**Directive activates when a matching attribute is encountered. Restrict mode is defined by “A”.  
  
**Example***<span ng-directive></span>*  
  
**CSS**Directive activates when a matching css style is encountered. Restrict mode is defined by “C”.   
  
**Example***<span class="ng-directive"></span>*  
  
**Comment** Directive activates when a matching comment is encountered. Restrict mode is defined by “M”.   
  
**Example** *<!-- directive: ng-directive -->*  
Let us create some custom directives:  
  
Now we read how to create custom directives. We start with some simple examples and move towards some complex custom directives.  
  
**Example 1**

1. <html>
3. <head>
4. <title>Angular JS Example</title>
5. <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js">
6. </script>
7. <style>
8. .sparkline
9. {
10. background-color: brown;
11. font-size: large;
12. height: 100px;
13. width: 200px;
14. color: aqua
15. }
16. </style>
18. </head>
20. <body>
21. <h2>AngularJS Custom Directive Example</h2>
22. <div ng-app="mainApp">
23. <divng-demo>
24. </div>
25. </div>
27. </body>
28. <script>
29. var mainApp = angular.module("mainApp", []);
30. mainApp.directive('ngDemo', function() {
31. return
32. {
33. restrict: 'A',
34. template: '<div class="sparkline">This is simple Custom Directive of Element Type</div>'
35. }
36. });
37. </script>

40. </html>

**Output  
  
**

**For more details click on the link**

* [AngularJS Custom Directives](http://www.c-sharpcorner.com/UploadFile/f0b2ed/angularjs-custom-directives/)

**Question 42: What is AngularJS BootStrap Process?**  
**Answer**   
  
Bootstrapping an angular application is as simple as making coffee for yourself.  
  
There are two ways to bootStrap Angular Application.

* Automatic BootStrap (Coffee by Machine)
* Manual BootStrap (Handmade coffee, you can face some trouble)

**Automatic BootStrap**  
When DOM content is loaded, Angular looks for the ngApp directive which designates application root.  
If it finds ngApp directive   
  
a. It loads the module associated with this directive.   
  
**e.g.**

1. <html ng-app='myApp'>
3. <head>
4. <script src='angular.js'>
5. </script>
6. <script>
7. var app = angular.module('myApp', []);
8. app.controller('myController', function($scope) {
9. $scope.message = 'Dear';
10. });
11. </script>
12. </head>
14. <body>
15. <div ng-controller="myController">
16. <p> Hi {{ message}} </p>
17. </div>

From the above script, It will load "myApp".  
  
Now let's move to the Manual process of bootstraping an Angular application.  
  
**Manual Bootstrap:**  
There is a big difference in Automatic and manual Bootstrap.

1. You do not need to attach ng-app directive with the html element.
2. You call function angular.bootstrap(document,['myApp']).
3. <!doctype html>
4. <html>
6. <body>
7. <div ng-controller="myController"> Best movie: {{MovieName}}! </div>
8. <script src='angular.js'>
9. </script>
10. <script>
11. angular.module('myApp', []).controller('MyController', ['$scope', function($scope) {
12. $scope.MovieName = 'The IRON MAN';
13. }]);
14. angular.element(document).ready(function() {
15. angular.bootstrap(document, ['myApp']);
16. });
17. </script>
18. </body>
20. </html>

Angular.bootstrap can not create a module for you until you make a custom module to give it as a parameter inside.  
  
Before bootstrapping the process you need to add controllers, directives and services etc.  
  
Manual bootstrap comes into picture when you need more control over the initialization process, like if you want to perform an operation before Angular compiles a page.  
  
**NOTE** You should not use ng-app directive in case of manual bootstrapping of an angular application.

**For more details click on the link**

* [AngularJS BootStrap Process](http://www.c-sharpcorner.com/blogs/angularjs-bootstrap-process)

**Question 43: What is Constants in AngularJS?**  
**Answer**Constant are like services in AngularJS in which we can define our global data. It is declared using "constant" keyword.  
  
As we define our app-keys in Web.Config file for ASP.NET application, which further we can use anywhere in the application, likewise we can declare constant data in AngularJS globally that can be used throughout the application.  
  
We can inject Constant everywhere in controller or service like any other dependency (e.g.$http).AngularJS uses Singleton structure for creating Constant dependency in our Angular application.  
  
So, using the Constant you can create your Config.js file and it can be injected anywhere in your application.  
  
Now, let's start to define constant and will use it in controller.  
  
**First of all create angular module**

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

Then, create Config.js file and define Constant in it,

1. app.constant('config',
2. {
3. appName: 'Constants',
4. appVersion: 2.0
5. });

Now, use the above to declare Constant in our controller,

1. app.controller('mainController', function ($scope,config) {
3. $scope.ApplicationName = config.appName;
4. }

At last now consume this scope in our HTML view,

1. <title>{{ApplicationName}}</title>

**Conclusion**   
  
You can use constants for a lot of things. This blog is just a basic demo explanation about constant.  
  
**For more details click on the link**

* [Constant in AngularJS](http://www.c-sharpcorner.com/blogs/constant-in-angularjs1)

**Question 44: Explain ngClick And ngDblclick Directives In AngularJS?**  
**Answer**  
  
AngularJS provides many built-in directives. In this article we will discuss about ngClick and ngDbclick directives.  
  
**ngClick**  
This directive allows us to define custom behavior on element click event. This directive has highest priority. In expression, event object is accessible as $event.  
  
**Syntax**

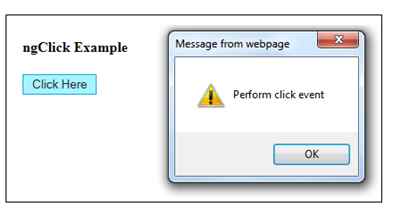
1. <ANY ELEMENT ng-click="expression">
2. ...
3. </ANY ELEMENT>

**Example**  
**HTML**

1. <h4>ngClick Example</h4>
2. <div ng-controller="HomeController">
3. <button ng-click="click()"> Click Here </button>
4. </div>

**Controller**

1. var app = angular.module("app", []);
2. app.controller("HomeController", function($scope)
3. {
4. $scope.click = function()
5. {
6. alert('Perform click event');
7. }
8. });

**Output**  
  
  
**ngDblclick**  
This directive allows us to define custom behavior on element double click event. This directive has highest priority. In expression, event object is accessible as $event.  
  
**Syntax**

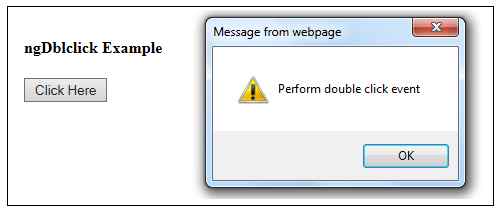
1. <ANY ELEMENT ng- dblclick ="expression">
2. ...
3. </ANY ELEMENT>

**Example**  
**HTML**

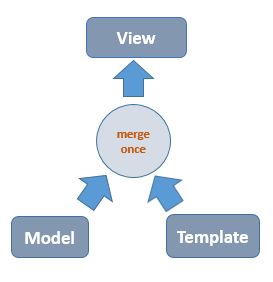
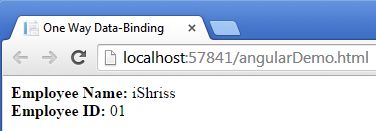
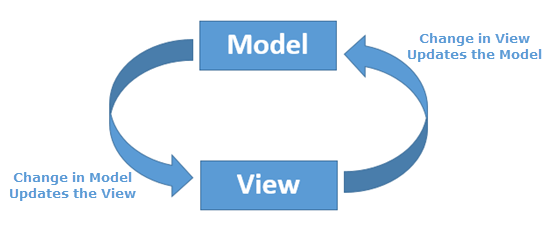
1. <h4>ngDblclick Example</h4>
2. <div ng-controller="HomeController">
3. <button ng-dblclick="dblclick()"> Click Here </button>
4. </div>

**Controller**

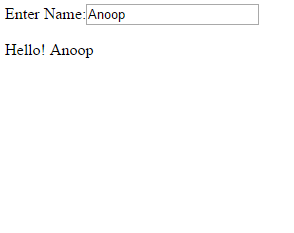
1. var app = angular.module("app", []);
2. app.controller("HomeController", function($scope)
3. {
4. $scope.dblclick = function()
5. {
6. alert('Perform double click event');
7. }
8. });

**Output**  
  
  
**For more details click on the link**

* [ngClick And ngDblclick Directives In AngularJS](http://www.c-sharpcorner.com/UploadFile/ff2f08/ngclick-and-ngdblclick-directives-in-angularjs/)

**Question 45: What is One-Way Data Binding in AngularJS?  
  
Answer**   
  
One-Way Data Binding simply means that HTML elements reflect the change. When the model values change the HTML elements don't change the values in the model.  
  
In other words, when the model changes, this change is reflected in the view but the view doesn't change the model. In other words the developer must write extra code for automatic synchronization of data between the model and the view components. One-Way binding may also be called one-direction binding.  
  
  
         Figure 1 One-way Binding  
  
Let us understand with an example.  
  
  
  
Press F5  
  
  
  
**Question 46: What is One-Way Data Binding in AngularJS?  
  
Answer**One of the core features of AngularJS which makes it popular is two way data binding. In two way data binding, any changes to the model are immediately reflected in the View and any changes in the View updates the model.  
  
  
  
**Example**

1. <!DOCTYPE html>
2. <html xmlns="http://www.w3.org/1999/xhtml">
4. <head>
5. <title></title>
6. <script src="Script/angular.js"></script>
7. <script type="text/javascript">
8. var myApp = angular.module('myApp', [])
9. .controller('myController', function($scope)
10. {
11. $scope.name = "Anoop";
12. });
13. </script>
14. </head>
16. <body ng-app="myApp">
17. <div ng-controller="myController">
18. Enter Name:<input type="text" ng-model="name" />
19. <p>Hello! {{name}}
20. </div>
21. </body>
23. </html>

In the above code, we have created a controller (i.e. myController) and registered it with myApp module. We used ng-model property for displaying the value of HTML control with the help of {{name}} Template. If we change the value in Textbox then it will update the model or if we change the value of model then it will immediately update the View.  
  
**Preview  
  
**  
**For more details click on the link**

* [Two Way Data Binding In AngularJS](http://www.c-sharpcorner.com/UploadFile/1e050f/two-way-data-binding-in-angularjs/)

**Question 47: Explain ng-hide Directive in AngularJS?  
  
Answer**  
  
ng- hide directive is used to make HTML elements invisible. Let us see this with the help of an example.  
  
Write the following HTML mark up in the webpage.

1. <!doctype html>
2. <htmlng-app>
4. <head>
5. <title>My Angular App</title>
6. <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
7. </head>
9. <body>
10. <div ng-app="">
12. <p ng-hide="true">This text is not visible.</p>
14. <p ng-hide="false">This text is visible.</p>
16. </div>
17. </body>
19. </html>

We have taken twp<p> tags in HTML. In one of the tags we will assign a value, false, to the ng-hide directive and in the other we will keep the value to true. Let us see how the output turns out.  
  
  
**For more details click on the link**

* [ng-hide Directive in AngularJS](http://www.c-sharpcorner.com/blogs/nghide-directive-in-angularjs)

**Question 48: What are animating elements in AngularJS?  
  
Answer**The $animate service allows you to provide transition effects when elements are added, removed, or moved in the DOM. The $animate service doesn’t define any animations itself but relies on the CSS3 animation and transition features. Animations can be a useful means of drawing the user’s attention to an important change in the layout of an application, making the transition from one state to another less jarring. Many developers treat animations as an outlet for their frustrated artistic ambition and ladle on as many as possible. The results can be annoying, especially for the user who has to endure endless special effects every time they perform a task. For a line-of-business application, where the user could be repeating the same set of actions all day, the effect is demoralizing beyond description. Animations should be subtle, brief, and quick. The goal is to draw the user’s attention to the fact that something has changed. Use animations consistently, cautiously, and—above all—sparingly.  
  
**Defining and Applying an Animation**You don’t work directly with the $animate service to apply animations. Instead, you define animations or transitions with CSS, following a special naming convention, and then apply those names as classes to elements, which also have AngularJS directives.  
  
The built-in directives that support animation and the names associated with them: The name enter is used when content is shown to the user. The name leave is used when content is hidden from the user. The name move is used when content is moved within the DOM. The names add and remove are used when content is added and removed from the DOM.

|  |  |
| --- | --- |
| **Directive** | **Names** |
| ng-repeat | enter, leave, move |
| ng-view | enter, leave |
| ng-include | enter, leave |
| ng-switch | enter, leave |
| ng-if | enter, leave |
| ng-class | add, remove |
| ng-show | add, remove |
| ng-hide | add, remove |

**For more details click on the link**

* [AngularJS From Beginning: Animation - Part Ten](http://www.c-sharpcorner.com/article/angularjs-from-beginning-animation-part-ten/)

**Question 49: What is ngClass directive in AngularJS?  
  
Answer**   
  
*ngClass directive*This directive lets us do things like,

* Add/Remove classes based on Angular variables.
* Add/Remove classes based on evaluated expressions.
* Bind single or multiple classes based on dynamic data.

**Some Points about ng-class**

1. The ng-class directive dynamically binds one or more CSS classes to an HTML element.
2. The value of the ng-class directive can be a string, an object, or an array.
3. If it is a string, it should contain one or more, space-separated class names.
4. As an object, it should contain key-value pairs, where the key is a Boolean value, and the value is the class name of the class you want to add. The class will only be added if the key is set to true.  
     
   **Examples  
     
   Ex 1**
   1. <div ng-class="{class1 : expression1, class2 : expression2}">
   3. Hello World!
   5. </div>

Here class1 will apply if the expression1 is true and class2 will apply if the expression2 is true.  
  
**Ex 2**

* 1. < div ng-class="{'class1 class2' : expression1}">
  2. Hello World!
  4. < /div>

Here class1 and class2 will apply if the expression1 is true.  
  
We can reduce the above code into,

* 1. < div ng-class="{'class1 class2' : expression1}">
  2. Hello World!
  4. < /div>

**For more details click on the link**

* [Ng-class in AngularJS](http://www.c-sharpcorner.com/blogs/ngclass-in-angularjs)

**Question 50: Why is scopeless controller used in AngularJS?  
  
Answer**Sometimes controller become complex by using $scope for providing data and behavior to view, in that situation we can use scopeless controller.   
  
But if you have designed your AngularJS application perfectly, there is no need to go for scopeless controllers.  
  
**Creating scope-less controller**

1. angular module(app.js):
2. angular.module('myApp', []);

**Controller (homeController.js)**

1. var app = angular.module("myApp");
3. app.controller("myController", function()
4. {
6. **this**.title = 'scopeless Controller Test';
7. **this**.name = 'Anupam';
8. **this**.sayHello = function()
9. {
10. alert('Hello ' + **this**.name);
11. }
13. });

As you can see I have used JavaScript for this keyword to add data and behavior in my controller.  
  
I would love to explain this here but I am still exploring what thi is in JavaScript.  
   
Here is how we can get data from controller to view.  
  
Binding Data to View using scope-less controller  
  
**View (index.html)**

1. <!DOCTYPE html>
2. <html ng-app="myApp" ng-controller="myController as ctrl">
4. <head>
5. <script src="Scripts/angular.min.js"></script>
6. <script src="app/app.js"></script>
7. <script src="app/homeController.js"></script>
8. <link href="Css/bootstrap.min.css" rel="stylesheet" />
9. <title>{{ctrl.title}}</title>
10. </head>
12. <body>
13. <nav role="navigation" class=" navbar navbar-default">
14. <div class="navbar-header">
15. <a href="#" class="navbar-brand">
16. {{ctrl.title}}
17. </a>
18. </div>
20. </nav>
21. <div class="container body-content">
22. <div class="col md-6">
23. <div class="row">
24. <div class="well-lg">
25. Hi {{ctrl.name}}
26. </div>
27. </div>
28. <div class="row">
29. <div class="well-lg">
30. <input type="button" ng-click="ctrl.sayHello()" value="Say Hello" class="btn" />
31. </div>
32. </div>
33. </div>
34. </div>
35. </body>
37. </html>

Here I have used a variable ctrl (myController as ctrl) which is an instance of myController.  
  
**For more details click on the link**

* [Scope-less Controller In AngularJS](http://www.c-sharpcorner.com/UploadFile/56fb14/scopeless-controller-in-angularjs/)