InterviewBit

AngularJS Interview Questions

To view the live version of the page, click here.

Contents

AngularJS Interview Questions for Experienced

(.....Continued)

19. What can you tell about the given piece of code?
20. What is the importance of track by in the ng-repeat directive?
21. What does the following code do? What are the permitted values of the restrict attribute?
22. Differentiate between compile and link in AngularJS?
23. How does routing work in AngularJS?
24. Write a syntax to send sample HTTP POST request in AngualrJS?
25. What is the importance of the $location service?
26. What is the importance of orderBy?
27. Why do we use ng-include?
28. Is it possible to create nested controllers?
29. What are AngularJS filters?
30. What can you say about the digest phase in AngularJS?
31. What are the different phases of the lifecycle of AngularJS Scope?
32. How will you improve performance of an AngularJS application?
33. What is the difference between the scopes of a directive and the scopes of a controller?
34. How will you hide an HTML tag element on click of a button click in AngularJS? Write a program for the same.
35. How can you maintain logs in AngularJS?
36. How do you achieve internationalization?
37. What is the auto bootstrap process?
38. What are the lifecycle hooks available?

AngularJS Interview Questions for Experienced

(.....Continued)

39. What is the difference between the $ and the $$ prefixes?

Let's get Started

AngularJS is the most popular, open-source, structural JavaScript-based framework, developed by Google, that was mainly built for developing large-scale, enterprise-level, dynamic, single-page web applications by extending the HTML syntax and following the MVC (Model-View-Controller) architecture. AngularJS is used for creating easily maintainable, responsive, and cross-browser-compatible enterprise-level web applications. It was developed in 2009 by 2 developers Misko Hevery and Adam Abrons and is currently being maintained by Google. The latest stable version of AngularJS is **v1.8.2 / 21** as of October 2020. Since AngularJS is completely free and it provides the option to focus on developing client-side logic in a cleaner way, it has been adopted by thousands of developers across the world for developing web applications.

AngularJS Interview Questions for Freshers

1. Define AngularJS and what are its key features?

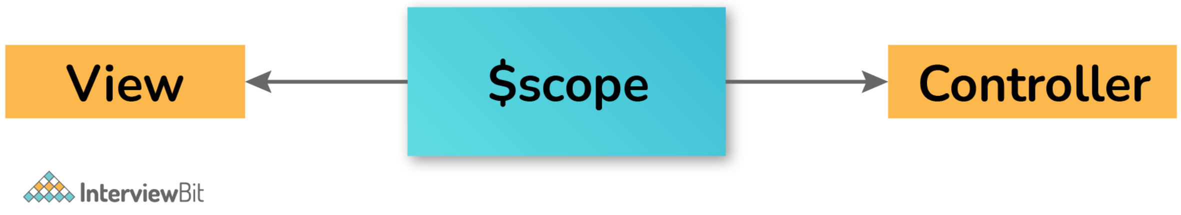
AngularJS is one of the most popular, open-source, JavaScript-based frameworks, developed by Google, that was mainly built for developing large-scale, enterprise-level, dynamic, single-page web applications. AngularJS uses HTML as its main template language and uses its syntax for representing the application's components such as directives. AngularJS is used mainly for writing client-side logic using the combined power of JavaScript and MVC architecture. This combined power results in the creation of easily maintainable, cross-browser-compatible enterprise-level web applications.

The main features of AngularJS are listed below:

- **Applications** developed in AngularJS are testable.
- **Data-binding** – AngularJS provides the most important feature of data binding which facilitates the synchronization of data between the model and the view components in the framework.
- **Controller** – AngularJS is built on JavaScript components and the JavaScript functions bound to scope are called controllers.
- **Services** – AngularJS has many in-built services such as $http which helps in making XMLHttpRequests and AJAX calls.
- **Scope** – AngularJS provides special objects called Scope which refer to the models and is a glue between the view and the controller.
- **Filters** – AngularJS supports several in-built filters as well as provides the ability to define custom filters that aid in subsetting the array items and filtering based on required conditions.
- **Directives** – Directives represent the markers of the DOM elements like attributes, elements, CSS, etc. They are used for creating custom HTML tags that act as widgets. AngularJS supports in-built directives like ngBind, ngModel, ngHide, etc, and also supports the creation of user-defined directives to achieve code reusability.
- **Routing** – Routing is the most important concept supported by AngularJS that involves switching of the views based on any condition.
- **MVC pattern** – MVC pattern also stands for Model-View-Controller pattern is followed by AngularJS that helps it allocate responsibilities appropriately. Model does the task of managing the application data. Views do the task of displaying the application data and the controllers act as an interface between the Model and View to implement application logic.
- **Dependency Injection** – AngularJS was mainly created to demonstrate the feature of dependency injection. This feature helps developers to develop, maintain and test applications easily by defining the interactions and resolving the dependencies between various components.

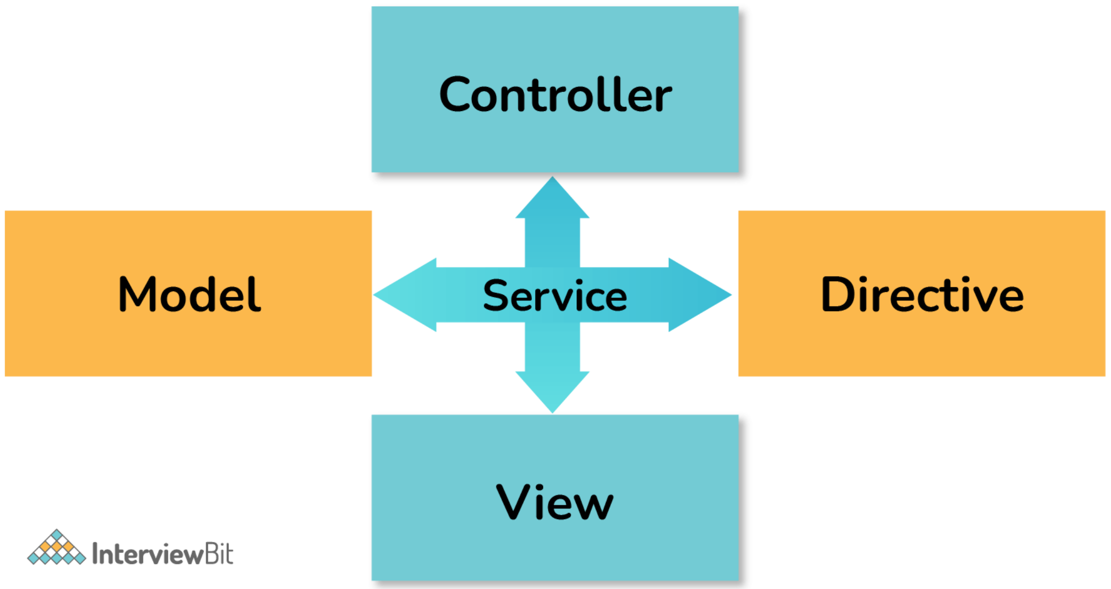
2. Define Scope in AngularJS.

Scopes are special objects in AngularJS that act as a glue between the view and the controller. They refer to the model component of the MVC architecture. They are arranged in a hierarchical way to mimic the DOM structure hierarchy. AngularJS has an in-built `$scope` object that has all the application data and the corresponding methods bound to that scope.



3. What do the services represent in AngularJS?

Services are single objects which carry out tasks they are created for. They interact with each other and are wired by using the concept of Dependency Injection that helps the framework in organizing and sharing the code across the application. There are various in-built services provided by AngularJS. AngularJS also supports the creation of custom services that are more commonly used by developers.



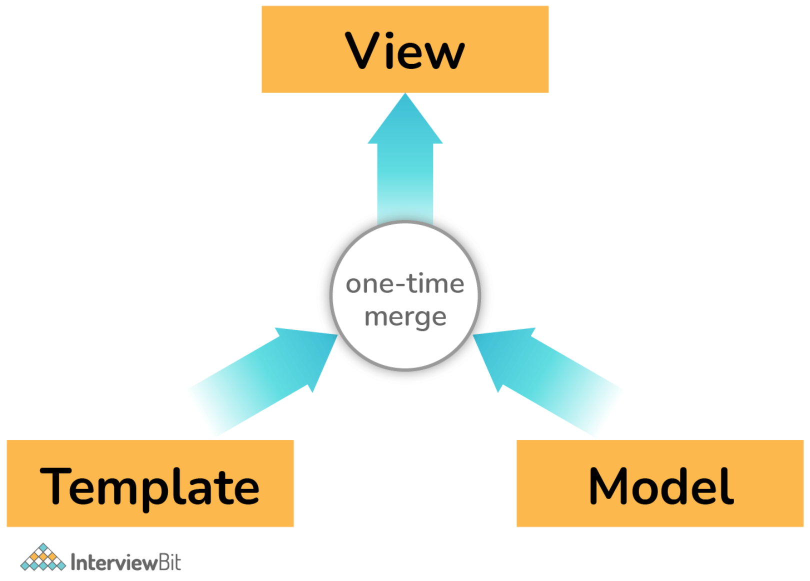
4. What are directives?

Directives are the most important components of AngularJS elements that represent the DOM element markers providing new behavior to the DOM elements like elements name, attributes, CSS classes, or comments. They are used for creating custom HTML tags that operate similarly to custom widgets. AngularJS provides various in-built directives such as ng-model for data binding, ng-repeat for iterating elements, ng-app for bootstrapping AngularJS applications, ng-show, ng-hide for manipulating the display of DOM elements, etc.

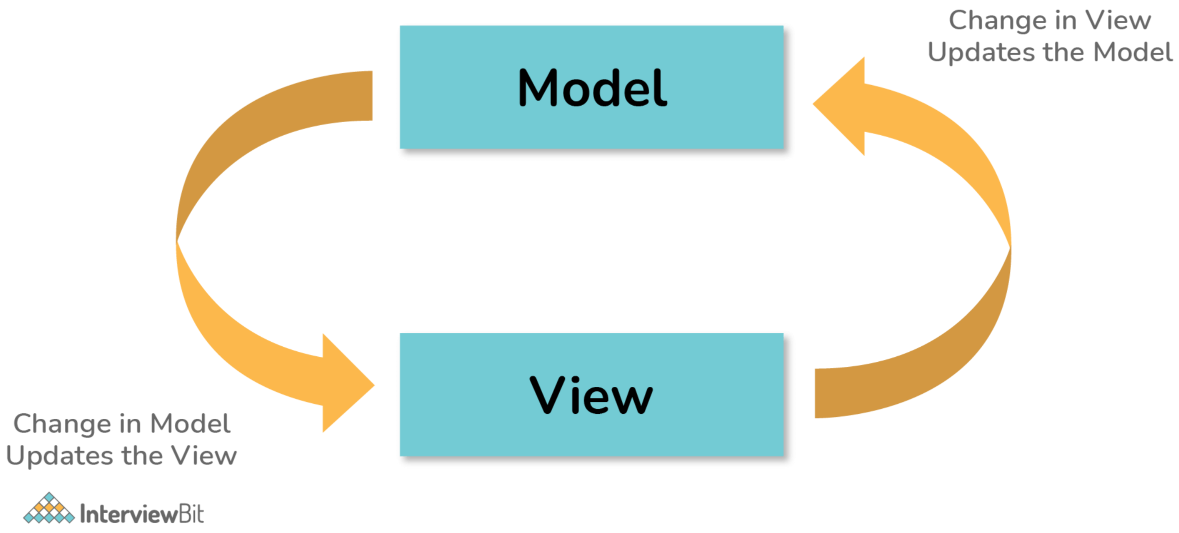
5. Explain the data binding process in AngularJS.

Data binding is the process of automatic syncing of data between the view and the model components. AngularJS achieves this by making use of the ng-model and ng-bind built-in directives. This directive ensures that the model is the single point of truth for the view and ensures that the view synchronizes with the model at any instant of time. There are two ways of data-binding:

- **One Way Data Binding**: Changes in the model are reflected on the view but changes in the view to that data are not reflected on the model. The binding is one way from the model to view. This is achieved by making use of the ng-bind directive.

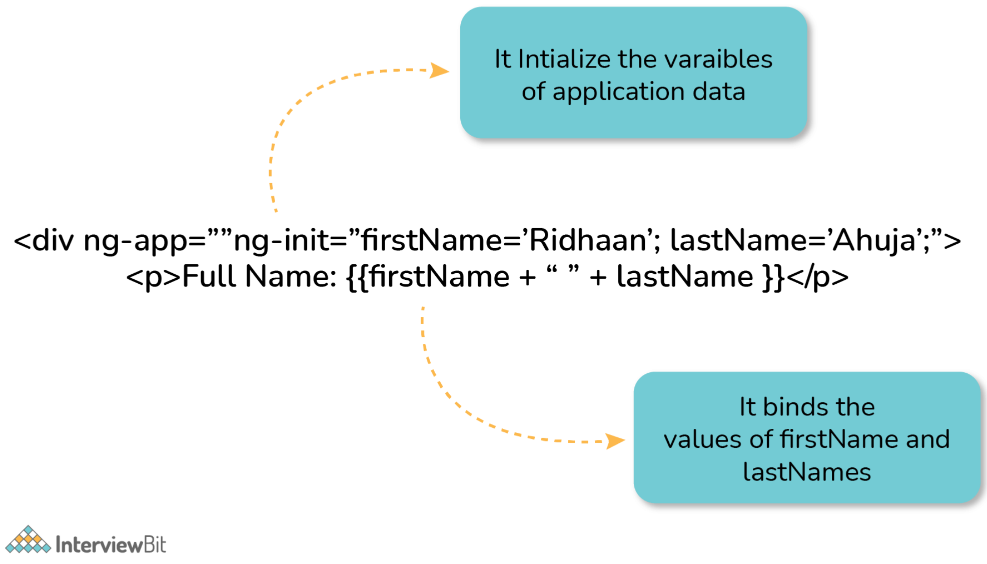


- **Two Way Data Binding**: As the name itself suggests, the changes in the model are reflected on the view as well as the view changes are reflected in the model. This is achieved by making use of the ng-model directive.


Model
View
Change in View Updates the Model
Change in Model Updates the View
InterviewBit

6. Explain the purpose of interpolation in AngularJS?

Interpolation refers to the phenomenon of binding data by embedding expressions to the attribute and text nodes. The compiler does the task of matching the text and the attributes during the compilation. Internally, AngularJS uses `$interpolate` built-in service to check if there is any markup having embedded expressions and if found then they are updated and registered in the form of watches.


It Intialize the varaibles of application data
<div ng-app=""ng-init="firstName='Ridhaan'; lastName='Ahuja';">
<p>Full Name: {{firstName + " " + lastName }}</p>
It binds the values of firstName and lastNames
InterviewBit

7. How can you integrate AngularJS with HTML?

We can integrate AngularJS in the HTML page by first binding the AngularJS library to the HTML page using the `<script>` tag in the HTML head section and then bootstrapping the AngularJS application using the ng-app directive as shown below.

```
<html>
   <head>
      <script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.mir
      <!--Other libraries-->
   </head>
   <!--ng-app attribute to bootstrap AngularJS application-->
   <body ng-app = "myApp">
        <!--Web page elements-->
   </body>
</html>
```

8. Define $rootScope in AngularJS application.

`$rootScope` refers to the scope object created on the DOM element containing the ng-app directive meant for bootstrapping the AngularJS application. This object is available across the whole AngularJS application. There can be only one `$rootScope` object in the entire application. All other scope objects are known as the child scopes of that `$rootScope` object.

9. Differentiate between ng-if and ng-show directives.

The `ng-if` directive **does not render** the DOM element portion if the condition specified is not satisfied. Here, the scope of the element would be destroyed if that is not rendered.

`ng-show` directive on the other hand renders the DOM element but **hides** the display (applying the `ng-hide` class on that DOM element) if the condition specified within it is not satisfied.

10. Why is $watch used?

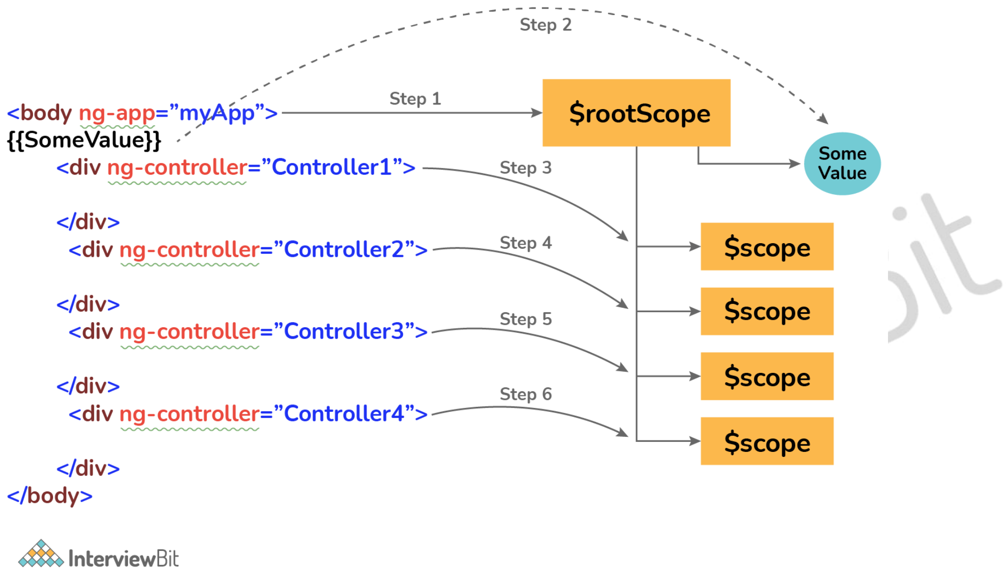
`$watch` is used for keeping track of the old and new values of the expression or the model variable that is being watched. One such usage is as shown below:

```
$scope.$watch("trackedVariable",
   function (newValue, oldValue){
       console.log("Value Changed : ", newValue);
   });
```

In the above example, we are watching the model variable named `trackedVariable`. If its value changes, we are immediately printing on the console.

11. What is scope hierarchy?

Every application developed in AngularJS has one `$rootScope` object and many child `$scope` objects. Whenever a new scope is created, that is added to the parent scope. This results in the creation of a hierarchical structure like the DOM structure.

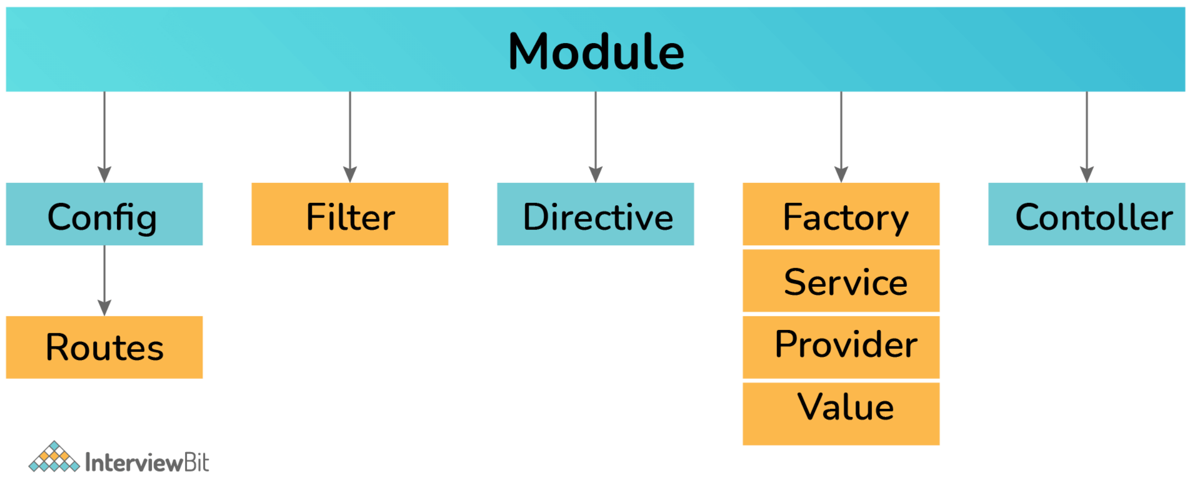


12. What is an AngularJS module?

An AngularJS module is nothing but a container for maintaining different components of the AngularJS application such as controller, filters, directives, services, etc, and a place where dependencies between them are defined. It can be treated like a main() method of Java. AngularJS module can be created by making use of the module() method of the angular object.
For example, in the below code, you are defining an `app` module for the `myFirstApp` application. You can define all the dependencies, if any, to this module within the square brackets.

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

To the `app` module, we can define all the controllers, filters, constants or directives, etc as shown in the figure:



For example, to define a controller, we follow the below approach:

```
app.controller("FirstController", ['$scope', function(obj) {
   obj.item = "Item 1";
}
]);
```

AngularJS Interview Questions for Experienced

13. Differentiate between expressions of AngularJS and JavaScript.

AngularJS expressions are placed inside double curly braces {{expression}} similar to JavaScript. The main differences between them are:

AngularJS expressions	JavaScript expressions
The expressions are evaluated against the scope object that they are part of.	The expressions are evaluated against the global window scope.
Expression evaluation nature is forgiving. If something goes wrong, it returns null or undefined.	JavaScript expressions are not forgiving in nature and return an error.
Here, loops and conditional statements cannot be added as part of expressions.	Loops and conditional statements can be part of JavaScript expressions.

14. List out the scope characteristics in AngularJS?

Scope object has 5 important characteristics.

- It provides the application with a context against which the AngularJS expressions are evaluated.
- It provides an option to observe the model changes within them using the $watch watcher service.
- The scope objects provide APIs like $apply that help in propagating the model changes throughout the application into the view from the sources like controllers, services, or various AngularJS event handlers.
- Scope objects inherit the parent properties and provide access to shared model properties.
- Scopes can even be nested to isolate directives and various AngularJS components.

15. How is the mouse double click event accomplished?

To specify any custom behaviour upon double click event on any HTML element, AngularJS makes use of the `ng-dblclick` directive. It is to be noted that the `ng-dblclick` does not override the JavaScript's `ondblclick` event. Example usage of this directive:

```
<button ng-dblclick="clicked = clicked + 1" ng-init="clicked=0">
 Double Click Here
</button>
```

The above piece of code increments the `clicked` variable by 1 upon every double click on the button.

16. How can you reset a $timeout and disable a $watch()?

In order to reset $timeout, we call the `.cancel()` method on it. as shown below:

```
var myTimer = $timeout(function() { /* your code */ }, 1000);
$timeout.cancel(myTimer);
```

To disable $watch, we can just call it as shown below:

```
var deregisterWatch = $scope.$watch(function() { /* Your code */ });
deregisterWatch(); // calling the watcher disables it.
```

17. Why is the findIndex() method used? What does it return in case the value is not found?

findIndex() method returns the position of the element in any object. In case the element is not found then the method returns -1.
For example:

```
var index = $scope.objectsList.findIndex(obj => obj.date =='2021-21-06');
```

Here, the index of the object where it has the date property value equal to 2021-21-06 is returned.

18. Is it possible for a parent controller to access the methods and properties of the child controller?

No, the parent controller can't access the properties and the methods of the child controller. But the child controller can access the parent's methods.

19. What can you tell about the given piece of code?

```
<select ng-options="employee.name for employee in employeeList">
</select>
```

The given piece of code would throw syntax error because in AngularJS it is not possible to use ng-options directives without using the ng-model directive. The ng-options dynamically generate the `<option>` elements for the given `<select>` element by evaluating the expression within it. Now upon selecting the element from the dropdown, the option value needs to be bound to a model which is defined by the ng-model directive. Absence of the ng-model results in error.

20. What is the importance of track by in the ng-repeat directive?

ng-repeat directive helps to keep track of all DOM elements dynamically to minimize DOM creation and rendering. It is achieved by storing the instances of the object whenever a new element gets added to the list or collection. AngularJS just renders the newly added element instead of re-rendering the overall collection. This helps in rendering the elements faster.

In case the ng-repeat is operated on a collection that has the objects of unique identifier, the tracking of new elements should be done based on that id instead of new instance insertion. This can be done by making use of the `track by` provided by the ng-repeat. For example, in the below piece of code:

```
<div ng-repeat="obj in objectList track by obj.id">
   <!--Some code -->
</div>
```

the tracking of new elements is done using the object's id.

21. What does the following code do? What are the permitted values of the restrict attribute?

```
app.directive('myFirstDirective', function() {
 return {
   restrict: 'E',
   scope: {
     directiveInfo: '=directiveInfo'
   },
   templateUrl: 'my-first-directive.html'
 };
});
```

- In the given piece of code, we have defined a custom directive called "myFirstDirective". Based on the value of restrict, we can say that the directive is restricted only to the element names. The directive has an isolated scope which has a property called "directiveInfo" that will be getting its value from the "directiveInfo" attribute of the element directive. The view or the template URL used for making the directive work is "my-first-directive.html".
- Following are the possible values of the restrict attribute in AngularJS directives:
    - 'A' - attribute names are matched.
    - 'E' - element names are matched.
    - 'C' - class names are matched.
    - 'M' - only comments are matched.

22. Differentiate between compile and link in AngularJS?

Compile is like a service used for traversing the HTML to find all the directives and return link functions.
The link does the task of combining the model with a view where any changes made to the model are reflected in the view and vice versa.
More information on compile and link can be found here.

23. How does routing work in AngularJS?

Routing enables us to create different URLs according to different contents in our app which in turn enables the users of the application to bookmark the contents as per their requirements. The route is that URL that can be bookmarked. Routing also helps in developing SPA (Single Page Applications) i.e create a single HTML page and update that page dynamically as and when the user interacts.

AngularJS supports routing by making use of its routing module called “ngRoute”. This module acts according to the URL. Whenever a user requests for a specific route/URL, the routing engine of the module, also called `$routeProvider` , renders the view based on that URL and defines what controller acts on this view - all based on the routing rules defined.

Consider the below snippet:

```
var myApp = angular.module('routingExample', ['ngRoute']);
myApp.config(function ($routeProvider) {
   $routeProvider.when('/', {
       templateUrl: '/login-page.html',
       controller: 'loginPageController'
   }).when('/employee/:empName', {
       templateUrl: '/employee-page.html',
       controller: 'employeeController'
   }).otherwise({
       redirectTo: "/"
   });
})
```

In the above example, we can see that to implement routing, we need to follow the below steps:

- While creating the application module, pass ‘ngRoute’ as the dependency module like below:

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

- Next, configure the routing rules for the application module by making use of the config() method by taking the $routeProvider service as the dependency.
- The `$routeProvider.when(path, route)` method is used for configuring the rules of routing where the first parameter defines what is the request URL and the second parameter defines the object containing the template, controller, and other property details that need to function upon requesting the URL.
    - In the given example, if the user requests for "/login-page.html", then inject login-page.html into the view and use the loginPageController.
    - For the URL "/employee/:empName" URL, the `:empName` refers to the URL parameter dynamically populated in the URL.
    - The otherwise() method is used for redirecting to the default or base URL for any other requests that are not part of the rules configured.

24. Write a syntax to send sample HTTP POST request in AngualrJS?

To perform any AJAX calls, AngularJS makes use of the $http service. The syntax is as below:

```
$http({
      method: "POST",
      url: "URL",
      data: JSON.stringify(value),
      contentType: 'application/json'
  }).then(function (successResponse)
      {
          // success callback action of the request
      },function (errorResponse)
      {
          // error callback action of the request
      });
```

25. What is the importance of the $location service?

`$location` is one of the built-in AngularJS services that helps to keep track of the application's URL, parses it, and makes the value available to the controller. In case the `$location` value is changed in the controller, the same is reflected on the browser's address bar. Changes to the URL on the address bar also result in reflection of the same on the `$location` service.

26. What is the importance of orderBy?

orderBy is a built-in filter in AngularJS that helps to re-order the array items based on defined criteria. For example, if we need to sort the items based on the ascending order of price, then we can follow the below code:

```
<ul>
<li ng-repeat = "item in items | orderBy:'price">
     {{ item.name + ', price:' + item.price }}
</li>
</ul>
```

27. Why do we use ng-include?

The ng-include directive is used for helping us to embed HTML pages inside a single HTML page. For example:

```
<div ng-app = "" ng-controller = "interviewBitController">
  <div ng-include = "'sample.htm'"></div>
  <div ng-include = "'example.htm'"></div>
</div>
```

In the above snippet, we are trying to include/embed sample.htm and example.htm page into our current AngularJS page.

28. Is it possible to create nested controllers?

Yes, it is possible to create nested controllers in AngularJS.
The sample code snippet can be as shown below:

```
<div ng-controller="mainController">
    <p>{{message}} {{name}}!</p>
   <div ng-controller="childController1">
       <p>Welcome to our app, {{name}}!</p>
      <div ng-controller="subChildController2">
            <p>{{message}} {{name}}! You are our esteemed guest {{name}}.</p>
      </div>
   </div>
</div>
```

29. What are AngularJS filters?

AngularJS filters are mainly used for formatting an expression while displaying it to the user. These can be used in controllers or services or views. AngularJS provides several inbuilt filters such as currency, filter, date, JSON, limitTo, etc whose purpose is to format the data without actually changing its value before merging to the expression and this is done by using the pipe character (|). AngularJS also provides support for registering and implementing custom filters and use them using the pipe symbol.

Syntax for using filters:

```
{{expression | filterName:filterInputParameter }}
```

For example, to format data to display the currency symbol before the salary value of say 18000:

```
{{salary | currency:'Rs.'}}
```

The salary will be displayed as “Rs.18,000”. Here ‘Rs.’ is the input to currency filter to define formatting. If nothing is specified, the default is considered as Dollars ($).

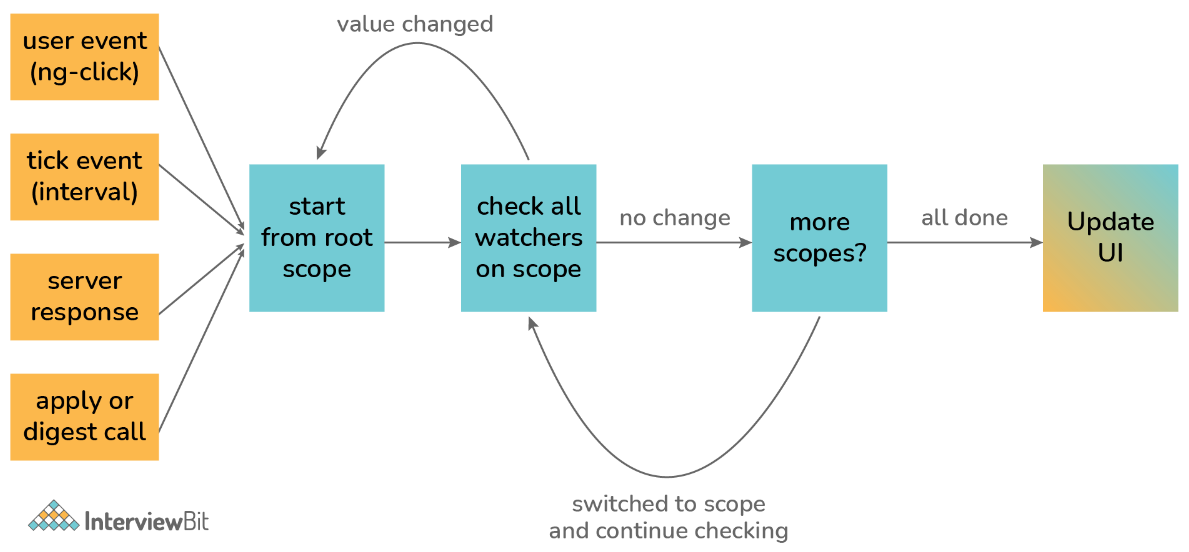
You can look into the implementation of different types of filters in the official documentation of AngularJS here.

30. What can you say about the digest phase in AngularJS?

The digest cycle or digest phase is the most important cycle required for the data binding process. It does the task of comparing the old version of a model with its new version. Whenever a change in the scope model is found during the comparison, the model watches are fired and another digest phase is initiated until the scope model is stable.

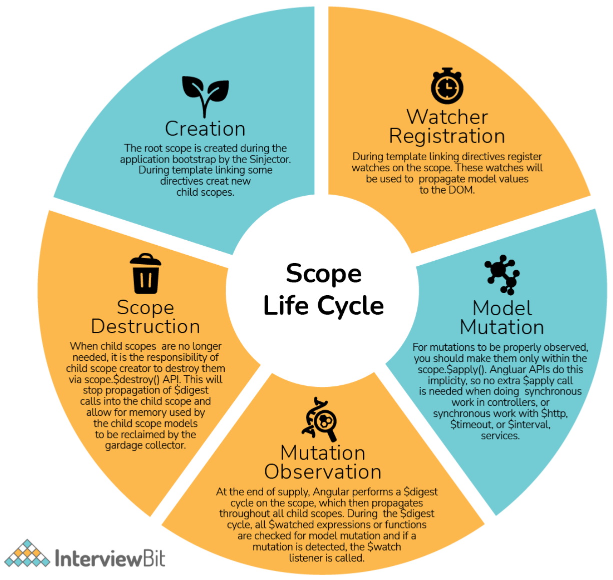
The digest cycle can be triggered manually by making use of $apply() or can be triggered automatically. The digest cycle gets triggered automatically when we use the core directives of AngularJS. In case we have any external code changes, then that would require manual triggering of the digest cycle.

The following diagram illustrates the process of the digest cycle clearly.



31. What are the different phases of the lifecycle of AngularJS Scope?

The following diagram illustrates the scope lifecycle in AngularJS:


Scope Life Cycle
Creation
The root scope is created during the application bootstrap by the Sinjector. During template linking some directives creat new child scopes.
Watcher Registration
During template linking directives register watches on the scope. These watches will be used to propagate model values to the DOM.
Model Mutation
For mutations to be properly observed, you should make them only within the scope.$apply(). Angluar APIs do this implicity, so no extra $apply call is needed when doing synchronous work in controllers, or synchronous work with $http, $timeout, or $interval, services.
Mutation Observation
At the end of supply, Angular performs a $digest cycle on the scope, which then propagates throughout all child scopes. During the $digest cycle, all $watched expressions or functions are checked for model mutation and if a mutation is detected, the $watch listener is called.
Scope Destruction
When child scopes are no longer needed, it is the responsibility of child scope creator to destroy them via scope.$destroy() API. This will stop propagation of $digest calls into the child scope and allow for memory used by the child scope models to be reclaimed by the gardage collector.
InterviewBit

- **Creation**: In this phase, the rootScope is created by $injector during the application bootstrap. During the phase of template linking, new child scopes relevant to the directives are created.
- **Watcher registration**: Here, the directives register the watches on the scope object which will be used to propagate values of models to DOM.
- Model mutation: The model mutations need to be present within the scope.$apply() for them to be properly observed. These will be done implicitly by AngularJS when working on synchronous or asynchronous work requests.
- **Mutation observation**: Once the $apply is complete, the digest cycle starts to observe for any model mutations on the scopes. Here, the $watches expressions are monitored for the model mutations and if any mutations are observed, then the $watch listener is called on the model.
- **Scope destruction**: As and when the child scopes are unnecessary, the creator of the scopes would have to destroy them by making use of scope.$destroy(). This ensures that the $digest cycle propagations are stopped and the memory once used by the child scopes is reclaimed.

32. How will you improve performance of an AngularJS application?

AngularJS makers have recommended the below two approaches for performance optimization in the production environment. They are:

- **Enable strict DI mode**: This can be achieved by making use of the directive ngStrictDi and can be implemented as:

```
<html ng-app="myFirstApp" ng-strict-di>
```

- **Disable debug data**: This can be achieved by using the debugInfoEnabled method of the $compileProvider service as shown below:

```
app.config(function ($compileProvider) {
     $compileProvider.debugInfoEnabled(false);
});
```

Apart from the above two, we can also improve the performance by following the below tips:

- Implementing one-time binding whenever possible.
- By making $httpProvider use the applyAsync feature. - - By refraining from creating many $watchers unnecessarily as too many of the watchers will lengthen the digest cycle thereby reducing the speed.
- If you have scenarios like repeated data calculations of the same nature, then we can make use of the $cacheFactory directive to store the data to avoid recalculations.
- In case we have a large number of elements to be looped, then instead of loading everything at once, pagination or infinite scroll can be implemented to reduce the data load. AngularJS provides ngInfiniteScroll directive for accomplishing the infinite scroll feature.

For more information regarding the tools that can be used to measure AngularJS performance, you can read here.

33. What is the difference between the scopes of a directive and the scopes of a controller?

Scopes of the controller and the directives are the instances of scope objects. The only difference lies in the naming convention for them. To understand the difference between scope and $scope, we need to understand directives with isolated scope using the following code:

```
app.directive('testAppDirective', function() {
 return {
   scope: {},
   link: function(myScopeVariable, elem,attr) {
     console.log(scope);
   }
 }
});
```

Here, we have defined a directive with isolated scope. The link function takes the signature scope, element, and attribute as the parameters. The name of the signature scope parameter can be anything as that parameter will be tagged to the scope of the directive's object. The $scope object that is usually injected into the controller cannot be used with another name. For example,

```
app.controller('myTestController',function(newScope)
{
});
```

results in error Error:

```
[$injector:unpr] Unknown provider: scopeProvider <- scope <- myTestController
```

Because the AngularJS dependency system tries to locate the dependency of the name newScope but fails to find it. Hence, to mark the dependency appropriately, the input to the controller function should be $scope.

34. How will you hide an HTML tag element on click of a button click in AngularJS? Write a program for the same.

This can be achieved by making use of the ng-click directive that controls the condition used for manipulating the display of the tag in the ng-hide directive.

```
<!DOCTYPE html>
<html>
<head>
   <meta chrset="UTF 8">
   <title>Button Click Hide</title>
</head>
<body>
   <script src="https://code.angularjs.org/1.6.9/angular.js"></script>
   <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
   <div ng-app="buttonDemo" ng-controller="buttonDemoController">
       <button ng-click="hideTag()">Hide IntervieBit</button>
       <div ng-hide="hideFlag">InterviewBit</div>
   </div>

   <script type="text/javascript">
       var app = angular.module('buttonDemo',[]);
       app.controller('buttonDemoController',function($scope){
           $scope.hideFlag = false;
           $scope.hideTag = function(){
               $scope.hideFlag = true;
           }
           });
   </script>
</body>
</html>
```

35. How can you maintain logs in AngularJS?

Logs in AngularJS can be maintained by using the $log built-in service. These are mainly used for troubleshooting and debugging in case of any unexpected scenarios. They are done by mainly using the below methods:

1. **log()**: To log a message onto the console. Example usage: `$log.log('Entered some function')`
2. **info()**: To write any message which represents information. Example usage: `$log.info('Data processed successfully')`
3. **warn()**: To log warnings. Example usage: `$log.warn('The value is empty.')`
4. **error()**: To log errors. Example usage: `$log.error('Oh no! Something went wrong.')`
5. **debug()**: To log any debug messages useful for debugging. Example usage: `$log.debug('Processed a variable A.')`

36. How do you achieve internationalization?

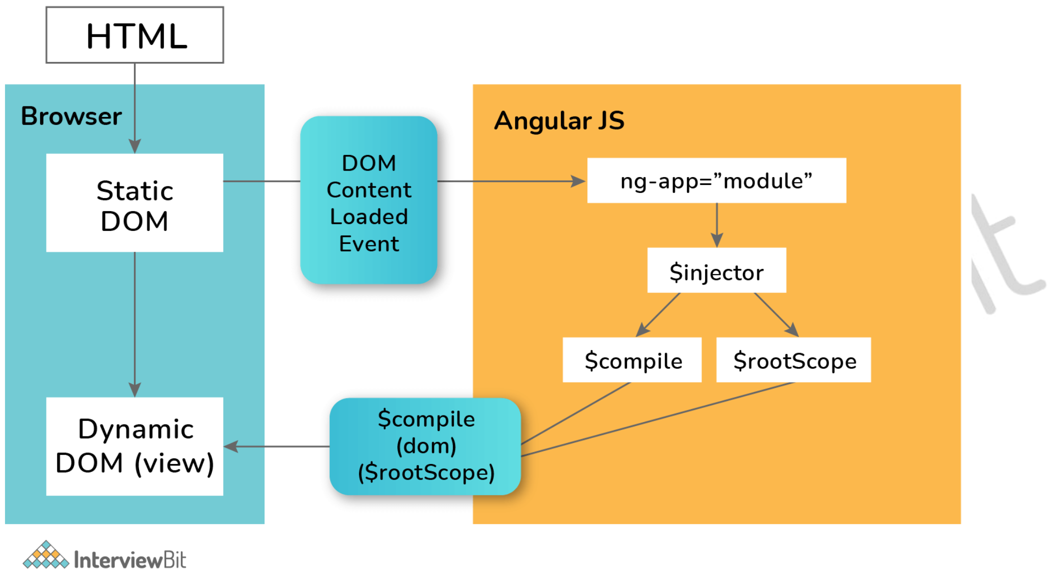
Internationalization is a way of showing locale-specific content on our applications. For example, the website in the United Kingdom needs to be displayed in English whereas the same website needs to be shown in Hindi for the users of India. By incorporating multiple languages support in our platform, we are ensuring that our website reaches a wider target audience.

Internationalization is supported in AngularJS using the `angular-translate` module which has lots of filters and directives and also the ability to asynchronously load i18n based data. The module also provides support for pluralization by making use of the highly configurable MessageFormat.

37. What is the auto bootstrap process?

Auto Bootstrapping is the process of automatically initiating the DOMContentLoaded event in the browser. The AngularJS application after downloading the angular.js library into the browser does the task of finding the ng-app directive which gives the root of the application. Once the directive is found, the following steps take place:

- The angular root module associated with the ng-app directive is loaded.
- The application injector is created which in turn creates the $compile and the $rootScope objects.
- The DOM is compiled from the ng-app element automatically and the content is rendered on the browser. This process is called auto bootstrapping.



38. What are the lifecycle hooks available?

There are many lifecycle hooks available in AngularJS and they are:

- **ngOnInit()**: This is a callback method that gets invoked as soon as the change detector detects any scope model changes for the first time and before any view has been checked. This is invoked once only when the directive is instantiated.
- **ngOnChanges()**: This callback function is triggered whenever AngularJS detects changes in the scope model and we can define the actions that need to follow up with that change in the property. It is called before ngOnInit() while instantiating the directive and is called every time the scope model changes.
- **ngDoCheck()**: This callback method does the task of change-detection and is invoked only after the default change-detector is run.
- **ngAfterContentInit()**: This is invoked once and that too as soon as AngularJS completes the initialization of all content of the directives.
- **ngAfterContentChecked()**: This callback method is invoked as soon as the default change-detector completes checking the content of the directives.
- **ngOnDestroy()**: This is used to clean up any component just before the directives or its properties are destroyed. They are useful for avoiding memory leaks and unsubscribe any unnecessary observables.
- **ngAfterViewChecked()**: This is immediately invoked once the default change-detector completes one cycle of change-check.

39. What is the difference between the $ and the $$ prefixes?

The `$$` prefix is used to define a private variable in AngularJS. This is responsible for avoiding accidental code changes due to exposure to the variables. Examples are `$$observers` , `$$watchers` , `$$childScope` etc.
The `$` prefix is used for defining built-in core AngularJS functionalities like variable, parameter, method, or any properties. Examples are `$scope` , `$http` , `$routeProvider` , `$watch` etc.

Conclusion:

In this article, we have seen the most commonly asked AngularJS interview questions for both the freshers and. the experienced developers. AngularJS was initially created to establish the concept of dependency injection and ease the development of single-page applications. Despite AngularJS being the first version, many companies have adopted this framework due to its raging benefits, easy configuration, and extensibility. Even though the framework has undergone a lot of changes in its different versions and releases, AngularJS has been the most useful framework for developing web applications in different companies.

References:

https://angularjs.org/
https://thinkster.io/a-better-way-to-learn-angularjs

Recommended Resources:

Javascript Interview
Angular Interview
Angular 8 Interview

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions