**ANGULAR:-**

**Beginners Level – Angular Interview Questions**

**1. Differentiate between Angular and AngularJS.**

|  |  |  |
| --- | --- | --- |
| **Feature** | **AngularJS** | **Angular** |
| ***Architecture*** | Supports MVC design model | Uses components and directives |
| ***Language*** | Recommended Language: JavaScript | Recommended Language: TypeScript |
| ***Expression Syntax*** | Specific ng directive is required for the image/property and an event | Uses () to bind an event and [] for property binding |
| ***Mobile Support*** | Doesn’t provide any mobile support | Provides mobile support |
| ***Routing*** | $routeprovider.when() is used for routing configs | @RouteConfig{(…)} is used for routing config |
| ***Dependency Injection*** | Doesn’t supports the concept of Dependency Injection | Supports hierarchical Dependency Injection with a unidirectional tree-based change detection |
| ***Structure*** | Less manageable | Simplified structure and makes the development and maintenance of large applications easier |
| ***Speed*** | With two-way data binding development effort and time are reduced | Faster than AngularJS with upgraded features |
| ***Support*** | No support or new updates are provided anymore | Active support and frequent new updates are made |

**2. What is Angular?**

Angular is an open-source front-end web framework. It is one of the most popular JavaScript frameworks that is mainly maintained by Google. It provides a platform for easy development of web-based applications and empowers the front end developers in curating cross-platform applications. It integrates powerful features like declarative templates, an end to end tooling, dependency injection and various other best practices that smoothens the development path.

**3. What are the advantages of using Angular?**

A few of the major advantages of using Angular framework are listed below:

* It supports two-way data-binding
* It follows MVC pattern architecture
* It supports static template and Angular template
* You can add a custom directive
* It also supports RESTfull services
* Validations are supported
* Client and server communication is facilitated
* Support for dependency injection
* Has strong features like Event Handlers, Animation, etc.

**4. What is Angular mainly used for?**

Angular is typically used for the development of SPA which stands for Single Page Applications. **Angular** provides a set of ready-to-use modules that simplify the development of single page applications. Not only this, with features like built-in data streaming, type safety, and a modular CLI,  Angular is regarded as a full-fledged web framework.

**5. What are Angular expressions?**

Angular expressions are code snippets that are usually placed in binding such as {{ expression }}similar to JavaScript. These expressions are used to bind application data to HTML

Syntax: **{{ expression }}**

**6. What are templates in Angular?**

Templates in Angular are written with HTML that contains Angular-specific elements and attributes. These templates are combined with information coming from the model and controller which are further rendered to provide the dynamic view to the user.

**7. In Angular what is string interpolation?**

String interpolation in Angular is a special syntax that uses template expressions within double curly**{{ }}** braces for displaying the component data. It is also known as **moustache syntax.**The JavaScript expressions are included within the curly braces to be executed by Angular and the relative output is then embedded into the HTML code. These expressions are usually updated and registered like watches, as a part of the digest cycle.

**8. What is the difference between an Annotation and a Decorator in Angular?**

Annotations in angular are “only” metadata set of the class using the Reflect Metadata library. They are used to create an “annotation” array. On the other hand, decorators are the design patterns that are used for separating decoration or modification of a class without actually altering the original source code.

**9. What do you understand by controllers in Angular?**

Controllers are JavaScript functions which provide data and logic to HTML UI. As the name suggests, they control how data flows from the server to HTML UI.

**10. What is scope in Angular?**

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

**11. What are directives in Angular?**

A core feature of Angular, directives are attributes that allow you to write new HTML syntax, specific to your application. They are essentially functions that execute when the Angular compiler finds them in the DOM.  The Angular directives are segregated into 3 parts:

1. Component Directives
2. Structural Directives
3. Attribute Directives

**12. What is data binding?**

In Angular, data binding is one of the most powerful and important features that allow you to define the communication between the component and DOM(Document Object Model). It basically simplifies the process of defining interactive applications without having to worry about pushing and pulling data between your view or template and component. In Angular, there are four forms of data binding:

1. String Interpolation
2. Property Binding
3. Event Binding
4. Two-Way Data Binding

**13. What is the purpose of a filter in Angular?**

Filters in Angular are used for formatting the value of an expression in order to display it to the user. These filters can be added to the templates, directives, controllers or services. Not just this, you can create your own custom filters. Using them, you can easily organize data in such a way that the data is displayed only if it fulfills certain criteria. Filters are added to the expressions by using the pipe character |, followed by a filter.

**14. What are the differences between Angular and jQuery?**

|  |  |  |
| --- | --- | --- |
| **Features** | **jQuery** | **Angular** |
| ***DOM Manipulation*** | **Yes** | **Yes** |
| ***RESTful API*** | **No** | **Yes** |
| ***Animation Support*** | **Yes** | **Yes** |
| ***Deep Linking Routing*** | **No** | **Yes** |
| ***Form Validation*** | **No** | **Yes** |
| ***Two Way Data Binding*** | **No** | **Yes** |
| ***AJAX/JSONP*** | **Yes** | **Yes** |

**15. What is a provider in Angular?**

A provider is a configurable service in Angular. It is an instruction to the Dependency Injection system that provides information about the way to obtain a value for a dependency. It is an object that has a $get() method which is called to create a new instance of a service. A Provider can also contain additional methods and uses $provide in order to register new providers.

**Intermediate Level – Angular Interview Questions**

**16.** **Does Angular support nested controllers?**

Yes, Angular does support the concept of nested controllers. The nested controllers are needed to be defined in a hierarchical manner for using it in the View.

**17. How can you differentiate between Angular expressions and JavaScript expressions?**

|  |  |
| --- | --- |
| **Angular Expressions** | **JavaScript Expressions** |
| 1. They can contain literals, operators, and variables. | 1. They can contain literals, operators, and variables. |
| 2. They can be written inside the HTML tags. | 2. They can’t be written inside the HTML tags. |
| 3. They do not support conditionals, loops, and exceptions. | 3. They do support conditionals, loops, and exceptions. |
| 4.  They support filters. | 4.  They do not support filters. |

**18. List at down the ways in which you can communicate between applications modules using core Angular functionality.**

Below are the most general ways for communicating between application modules using core Angular functionality :

* Using events
* Using services
* By assigning models on **$rootScope**
* Directly between controllers [**$parent**, **$$childHead**, **$$nextSibling**, etc.]
* Directly between controllers [**ControllerAs**, or other forms of inheritance]

**19. What is the difference between a service() and a factory()?**

A service() in Angular is a function that is used for the business layer of the application. It operates as a constructor function and is invoked once at the runtime using the ‘new’ keyword. Whereas factory() is a function which works similar to the service() but is much more powerful and flexible. factory() are design patterns which help in creating Objects.

**20. What is the difference between $scope and scope in Angular?**

* $**scope** in Angular is used for implementing the concept of dependency injection (D.I) on the other hand **scope** is used for directive linking.
* $**scope** is the service provided by $scopeProviderwhich can be injected into controllers, directives or other services whereas **Scope** can be anything such as a function parameter name, etc.

**21. Explain the concept of scope hierarchy?**

The $scope objects in Angular are organized into a hierarchy and are majorly used by views. It contains a root scope which can further contain scopes known as child scopes. One root scope can contain more than one child scopes. Here each view has its own $scope thus the variables set by its view controller will remain hidden to the other controllers. The Scope hierarchy generally looks like:

* Root $scope
  + $scope for Controller 1
  + $scope for Controller 2
  + ..
  + $scope for Controller ‘n’

**22. What is AOT?**

AOT stands for Angular Ahead-of-Time compiler. It is used for pre-compiling the application components and along with their templates during the build process. Angular applications which are compiled with AOT has a smaller launching time. Also, components of these applications can execute immediately, without needing any client-side compilation. Templates in these applications are embedded as code within their components. It reduces the need for downloading the Angular compiler which saves you from a cumbersome task. AOT compiler can discard the unused directives which are further thrown out using a tree-shaking tool.

**23. Explain jQLite.**

jQlite is also known as **jQuery lite** is a subset of jQuery and contains all its features. It is packaged within Angular, by default. It helps Angular to manipulate the DOM in a way that is compatible cross-browser. **jQLite** basically implements only the most commonly needed functionality which results in having a small footprint.

[*Powered by Edureka*](https://www.edureka.co/mymock-interview-service)

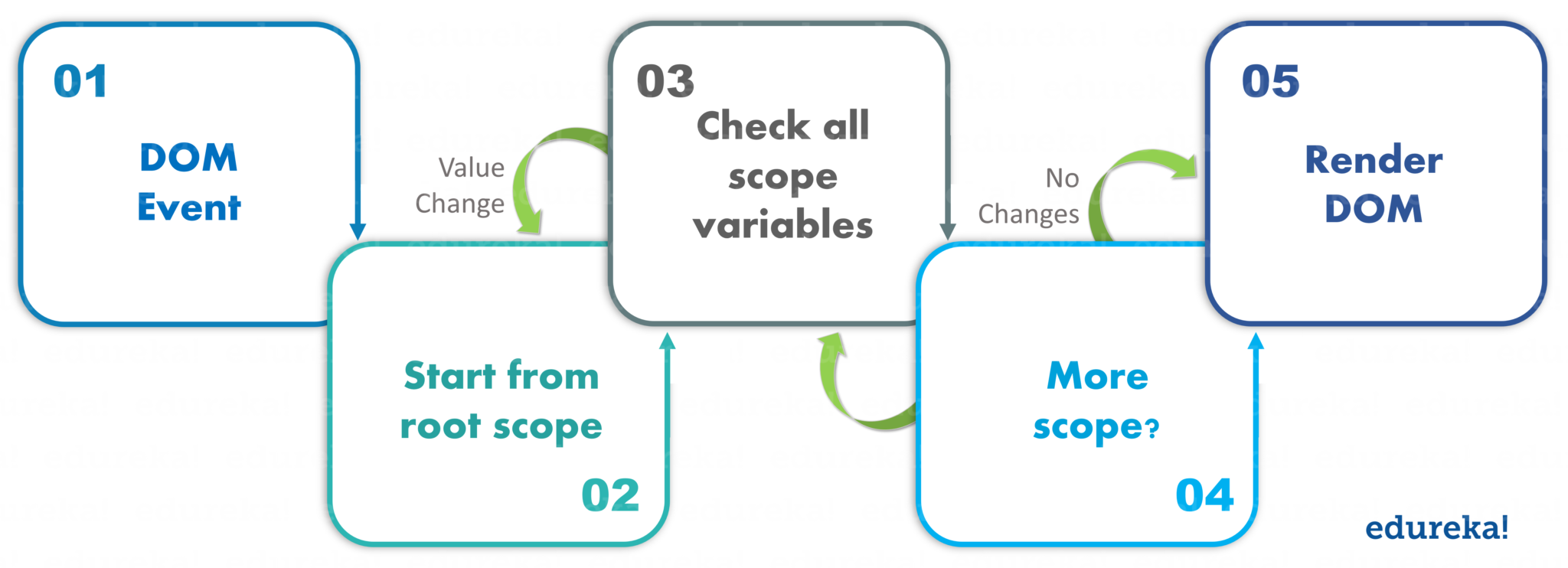
[**Angular Mock interviews for you**](https://www.edureka.co/mymock-interview-service)

* [**Interviews by Industry Experts**](https://www.edureka.co/mymock-interview-service)
* [**Personalized detailed interview feedback**](https://www.edureka.co/mymock-interview-service)
* [**Access to exclusive and curated content**](https://www.edureka.co/mymock-interview-service)

[SCHEDULE NOW](https://www.edureka.co/mymock-interview-service)

**24. Explain the process of digest cycle in Angular?**

The digest cycle in Angular is a process of monitoring the watchlist for keeping a track of changes in the value of the watch variable. In each digest cycle, Angular compares the previous and the new version of the scope model values. Generally, this process is triggered implicitly but you can activate it manually as well by using **$apply()**.



**25. What are the Angular Modules?**

All the Angular apps are modular and follow a modularity system known as *NgModules*. These are the containers which hold a cohesive block of code dedicated specifically to an application domain, a workflow, or some closely related set of capabilities. These modules generally contain components, service providers, and other code files whose scope is defined by the containing NgModule.  With modules makes the code becomes more maintainable, testable, and readable. Also, all the dependencies of your applications are generally defined in modules only.

**26. On which types of the component can we create a custom directive?**

Angular provides support to create custom directives for the following:

* **Element directives** − Directive activates when a matching element is encountered.
* **Attribute** − Directive activates when a matching attribute is encountered.
* **CSS** − Directive activates when a matching CSS style is encountered.
* **Comment** − Directive activates when a matching comment is encountered

**27. What are the different types of filters in Angular?**

Below are the various filters supported by Angular:

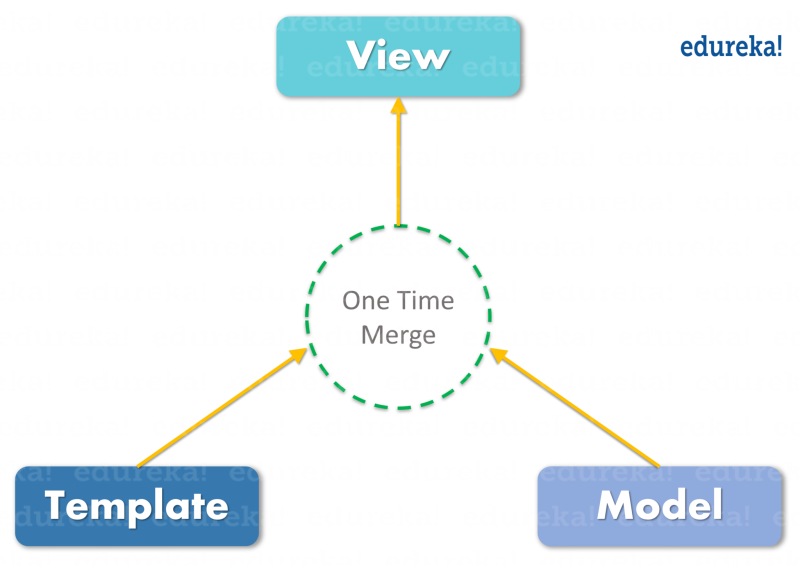
* **currency:** Format a number to a currency format.
* **date:** Format a date to a specified format.
* **filter:** Select a subset of items from an array.
* **json:** Format an object to a JSON string.
* **limit:**To Limits an array/string, into a specified number of elements/characters.
* **lowercase:** Format a string to lower case.
* **number:** Format a number to a string.
* **orderBy:** Orders an array by an expression.
* **uppercase:** Format a string to upper case.

**28. What is Dependency Injection in Angular?**

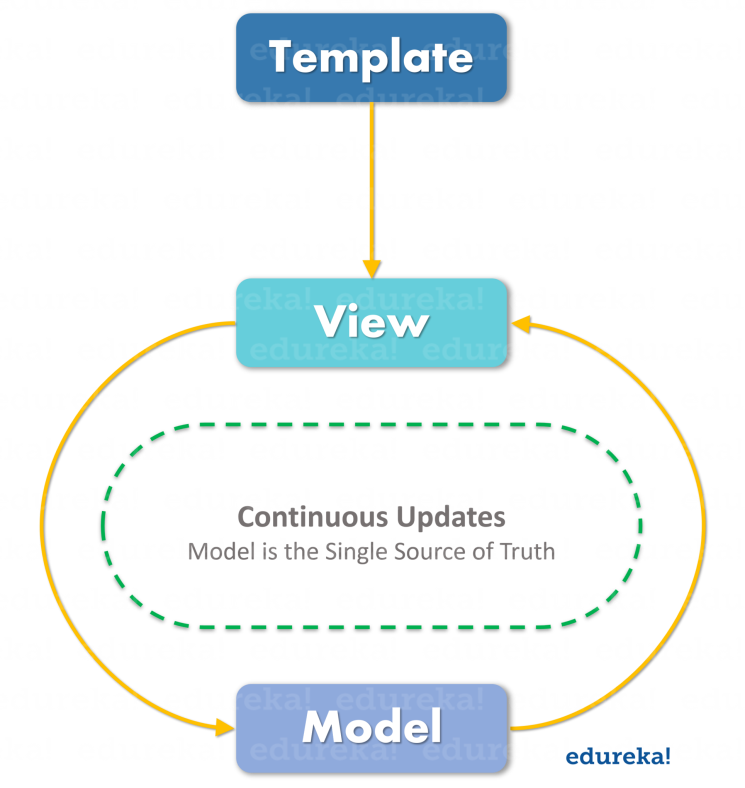
Dependency Injection (DI) is a software design pattern where the objects are passed as dependencies rather than hard-coding them within the component. The concept of Dependency Injection comes in handy when you are trying to separate the logic of object creation to that of its consumption. The ‘config’ operation makes use of DI that must be configured beforehand while the module gets loaded to retrieve the elements of the application. With this feature, a user can change dependencies as per his requirements.

**29. Differentiate between one-way binding and two-way data binding.**

In **One-Way** data binding, the View or the UI part does not update automatically whenever the data model changes. You need to manually write custom code in order to update it every time the view changes.



Whereas, in **Two-way** data binding, the View or the UI part is updated implicitly as soon as the data model changes. It is a synchronization process, unlike One-way data binding.



**30. What are the lifecycle hooks for components and directives?**

An Angular component has a discrete life-cycle which contains different phases as it transits through birth till death. In order to gain better control of these phases, we can hook into them using the following:

* **constructor:** It is invoked when a component or directive is created by calling new on the class.
* **ngOnChanges:** It is invoked whenever there is a change or update in any of the input properties of the component.
* **ngOnInit:** It is invoked every time a given component is initialized. This hook is only once called in its lifetime after the first ngOnChanges.
* **ngDoCheck:** It is invoked whenever the change detector of the given component is called. This allows you to implement your own change detection algorithm for the provided component.
* **ngOnDestroy:** It is invoked right before the component is destroyed by Angular. You can use this hook in order to unsubscribe observables and detach event handlers for avoiding any kind of memory leaks.

**31. What do you understand by dirty checking in Angular?**

In Angular, the digest process is known as **dirty checking**. It is called so as it scans the entire scope for changes. In other words, it compares all the new scope model values with the previous scope values. Since all the watched variables are contained in a single loop, any change/update in any of the variable leads to reassigning of rest of the watched variables present inside the DOM. A watched variable is in a single loop(digest cycle), any value change of any variable forces to reassign values of other watched variables in DOM

**32. Differentiate between DOM and BOM.**

|  |  |
| --- | --- |
| **DOM** | **BOM** |
| 1. Stands for Document Object Model | 1. Stands for Browser Object Model |
| 2. Represents the contents of a web page | 2. Works a level above web page and includes browser attributes |
| 3. All the Objects are arranged in a tree structure and the document can be manipulated & accessed via provided APIs only | 3. All global JavaScript objects, variables & functions become members of the window object implicitly |
| 4. Manipulates HTML documents | 4. Access and manipulate the browser window |
| 5. W3C Recommended standard specifications | 5. Each browser has its own implementation |

**33. What is Transpiling in Angular?**  
Transpiling in Angular refers to the process of conversion of the source code from one programming language to another. In Angular, generally, this conversion is done from TypeScript to JavaScript. It is an implicit process and happens internally.

**34. How to perform animation in Angular?**

In order to perform animation in an Angular application, you need to include a special Angular library known as Animate Library and then refer to the ngAnimate module into your application or add the ngAnimate as a dependency inside your application module.

**35. What is transclusion in Angular?**

The transclusion in Angular allows you to shift the original children of a directive into a specific location within a new template. The ng directive indicates the insertion point for a transcluded DOM of the nearest parent directive that is using transclusion. Attribute directives like **ng-transclude** or **ng-transclude-slot** are mainly used for transclusion.

**36. What are events in Angular?**

Events in Angular are specific directives that help in customizing the behavior of various DOM events. Few of the events supported by Angular are listed below:

* ng-click
* ng-copy
* ng-cut
* ng-dblclick
* ng-keydown
* ng-keypress
* ng-keyup
* ng-mousedown
* ng-mouseenter
* ng-mouseleave
* ng-mousemove
* ng-mouseover
* ng-mouseup
* ng-blur

**37. List some tools for testing angular applications?**

1. Karma
2. Angular Mocks
3. Mocha
4. Browserify
5. Sion

**38. How to create a service in Angular?**

In Angular, a service is a substitutable object that is wired together using dependency injection. A service is created by registering it in the module it is going to be executed within. There are basically three ways in which you can create an angular service. They are basically three ways in which a service can be created in Angular:

* Factory
* Service
* Provider

**39. What is a singleton pattern and where we can find it in Angular?**

Singleton pattern in Angular is a great pattern which restricts a class from being used more than once. Singleton pattern in Angular is majorly implemented on dependency injection and in the services. Thus, if you use ‘new Object()’ without making it a singleton, then two different memory locations will be allocated for the same object. Whereas, if the object is declared as a singleton, in case it already exists in the memory then simply it will be reused.

**40. What do you understand by REST in Angular?**

REST stands for **RE**presentational **S**tate **T**ransfer. REST is an API (Application Programming Interface) style that works on the HTTP request. In this, the requested URL pinpoints the data that needs to be processed. Further ahead, an HTTP method then identifies the specific operation that needs to be performed on that requested data. Thus, the APIs which follows this approach are known as RESTful APIs.

**41. What is bootstrapping in Angular?**

Bootstrapping in Angular is nothing but initializing, or starting the Angular app. Angular supports automatic and manual bootstrapping.

* ***Automatic Bootstrapping:*** this is done by adding the ng-app directive to the root of the application, typically on the tag or tag if you want angular to bootstrap your application automatically. When Angular finds ng-app directive, it loads the module associated with it and then compiles the DOM.
* ***Manual Bootstrapping:***Manual bootstrapping provides you more control on how and when to initialize your Angular application. It is useful where you want to perform any other operation before Angular wakes up and compile the page.

**42. What is the difference between a link and compile in Angular?**

* Compile function is used for template DOM Manipulation and to collect all the directives.
* Link function is used for registering DOM listeners as well as instance DOM manipulation and is executed once the template has been cloned.

**43.** **What do you understand by constants in Angular?**

In Angular, constants are similar to the services which are used to define the global data. Constants are declared using the keyword “constant”. They are created using constant dependency and can be injected anywhere in controller or services.

**44. What is the difference between a provider, a service and a factory in Angular?**

|  |  |  |
| --- | --- | --- |
| **Provider** | **Service** | **Factory** |
| A provider is a method using which you can pass a portion of your application into app.config | A service is a method that is used to create a service instantiated with the ‘new’ keyword. | It is a method that is used for creating and configuring services. Here you create an object, add properties to it and then return the same object and pass the factory method into your controller. |

**45.** **What are Angular Global APIs?**

Angular Global API is a combination of global JavaScript functions for performing various common tasks like:

* Comparing objects
* Iterating objects
* Converting data

There are some common Angular Global API functions like:

* **angular. lowercase:** Converts a string to lowercase string.
* **angular. uppercase:** Converts a string to uppercase string.
* **angular. isString:**Returns true if the current reference is a string.
* **angular. isNumber:** Returns true if the current reference is a number.

**Advanced Level – Angular Interview Questions**

**46. In Angular, describe how will you set, get and clear cookies?**

For using cookies in Angular, you need to include a  module called ngCookies angular-cookies.js.

**To set Cookies** – For setting the cookies in a key-value format ‘put’ method is used.

cookie.set('nameOfCookie',"cookieValue");

**To get Cookies –** For retrieving the cookies ‘get’ method is used.

cookie.get(‘nameOfCookie’);

**To clear Cookies –** For removing cookies ‘remove’ method is used.

cookie.delete(‘nameOfCookie’);

**47.  If your data model is updated outside the ‘Zone’, explain the process how will you the view?**

You can update your view using any of the following:

1. **ApplicationRef.prototype.tick()**: It will perform change detection on the complete component tree.
2. **NgZone.prototype.run():** It will perform the change detection on the entire component tree. Here, the run() under the hood will call the tick itself and then parameter will take the function before tick and executes it.
3. **ChangeDetectorRef.prototype.detectChanges():**It will launch the change detection on the current component and its children.

**48. Explain ng-app directive in Angular.**

ng-app directive is used to define Angular applications which let us use the auto-bootstrap in an Angular application. It represents the root element of an Angular application and is generally declared near <html> or <body> tag. Any number of ng-app directives can be defined within an HTML document but just a single Angular application can be officially bootstrapped implicitly. Rest of the applications must be manually bootstrapped.

**Example**

<div ng-app=“myApp” ng-controller=“myCtrl”>  
First Name :  
<input type=“text” ng-model=“firstName”>  
<br />  
Last Name :  
<input type=“text” ng-model=“lastName”>  
<br>  
Full Name: {{firstName + ” ” + lastName }}  
</div>

**49. What is the process of inserting an embedded view from a prepared TemplateRef?**

@Component({

selector: 'app-root',

template: `

<ng-template #template let-name='fromContext'><div>{{name}}</ng-template>

`

})

export class AppComponent implements AfterViewChecked {

@ViewChild('template', { read: TemplateRef }) \_template: TemplateRef<any>;

constructor() { }

ngAfterViewChecked() {

this.vc.createEmbeddedView(this.\_template, {fromContext: 'John'});

}

}

**50. How can you hide an HTML element just by a button click in angular?**

An HTML element can be easily hidden using the ng-hide directive in conjunction along with a controller to hide an HTML element on button click.

**View**

<div ng-controller="MyController">

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

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

</div>

**Controller**

controller: function() {

this.isHide = false;

this.hide = function(){

this.isHide = true; }; }

**Question: What is Angular?**  
**Answer:**Angular is a TypeScript-based open-source web application framework, developed and maintained by Google. It offers an easy and powerful way of building front end web-based applications.

Angular integrates a range of features like declarative templates, dependency injection, end-to-end tooling, etc. that facilitates web application development.

**Question**: **Define the ng-content Directive?**  
**Answer**: Conventional HTML elements have some content between the tags. For instance:

<p>Put your paragraph here</p>

Now consider the following example of having custom text between angular tags:

<app-work>This won’t work like HTML until you use ng-content Directive</app-work>

However, doing so won’t work the way it worked for HTML elements. In order to make it work just like the HTML example mentioned above, we need to use the ng-content Directive. Moreover, it is helpful in building reusable components.

Know more about the [ng-content directive](https://blog.angular-university.io/angular-ng-content/).

**Question: Please explain the various features of Angular.**  
**Answer:** There are several features of Angular that makes it an ideal front end JavaScript framework. Most important of them are described as follows:

* **Accessibility Applications**

Angular allows creating accessible applications using ARIA-enabled components, built-in a11y test infrastructure, and developer guides.

* **Angular CLI**

Angular provides support for command line interface tools. These tools can be used for adding components, testing, instant deploying, etc.

* **Animation Support**

Angular’s intuitive API allows the creation of high-performance, complex animation timelines with very little code.

* **Cross-Platform App Development**

Angular can be used for building an efficient and powerful desktop, native, and progressive web apps. Angular provides support for building native mobile applications using Cordova, Ionic, or NativeScript.

Angular allows creating high performance, offline, and zero-step installation progressive web apps using modern web platform capabilities. The popular JS framework can also be used for building desktop apps for Linux, macOS, and Windows.

* **Code Generation**

Angular is able to convert templates into highly-optimized code for modern JavaScript virtual machines.

* **Code Splitting**

With the new Component Router, Angular apps load quickly. The Component Router offers automatic code-splitting so that only the code required to render the view that is requested by a user is loaded.

* **Synergy with Popular Code Editors and IDEs**

Angular offers code completion, instant errors, etc. with popular source code editors and IDEs.

* **Templates**

Allows creating UI views with a simple and powerful template syntax.

* **Testing**

Angular lets you carry out frequent unit tests using Karma. The Protractor allows running faster scenario tests in a stable way.

**Question**: **Demonstrate navigating between different routes in an Angular application.**  
**Answer**: Following code demonstrates how to navigate between different routes in an Angular app dubbed “Some Search App”:

import {Router} from "@angular/router";

.

.

.

@Component({

 selector: 'app-header',

 template: `

<nav class="navbar navbar-light bg-faded">

 <a class="navbar-brand" (click)="goHome()">Some Search App</a>

 <ul class="nav navbar-nav">

   <li class="nav-item">

     <a class="nav-link" (click)="goHome()">Home</a>

   </li>

   <li class="nav-item">

     <a class="nav-link" (click)="goSearch()">Search</a>

   </li>

 </ul>

</nav>

`

})

class HeaderComponent {

 constructor(private router: Router) {}

 goHome() {

   this.router.navigate(['']);

 }

 goSearch() {

   this.router.navigate(['search']);

 }

}

**Question: Could you explain services in Angular?**  
**Answer:** Singleton objects in Angular that get instantiated only once during the lifetime of an application are called services. An Angular service contains methods that maintain the data throughout the life of an application.

The primary intent of an Angular service is to organize as well as share business logic, models, or data and functions with various components of an Angular application.

The functions offered by an Angular service can be invoked from any Angular component, such as a controller or directive.

**Question: Discuss the advantages and disadvantages of using Angular?**  
**Answer:** Following are the various advantages of using Angular:

* Ability to add a custom directive
* Exceptional community support
* Facilitates client and server communication
* Features strong features, such as Animation and Event Handlers
* Follows the MVC pattern architecture
* Offers support for static template and Angular template
* Support for two-way data-binding
* Supports dependency injection, RESTful services, and validations

Disadvantages of using Angular are enumerated as follows:

* Complex SPAs can be inconvenient and laggy to use due to their size
* Dynamic applications do not always perform well
* Learning Angular requires a decent effort and time

**Question**: **Enumerate some salient features of Angular 7.**  
**Answer**: Unlike the previous versions of Angular, the 7th major release comes with splitting in @angular/core. This is done in order to reduce the size of the same. Typically, not each and every module is required by an Angular developer. Therefore, in Angular 7 each split of the @angular/core will have no more than 418 modules.

Also, Angular 7 brings drag-and-drop and virtual scrolling into play. The latter enables loading as well as unloading elements from the DOM. For virtual scrolling, the latest version of Angular comes with the <cdk-virtual-scroll-viewport> package. Furthermore, Angular 7 comes with a new and enhanced version of the ng-compiler.

**Question: What is string interpolation in Angular?**  
**Answer:** Also referred to as moustache syntax, string interpolation in Angular refers to a special type of syntax that makes use of template expressions in order to display the component data. These template expressions are enclosed within double curly braces i.e. {{ }}.

The JavaScript expressions that are to be executed by Angular are added within the curly braces and the corresponding output is embedded into the HTML code. Typically, these expressions are updated and registered like watches as a part of the digest cycle.

**Question**: **Explain Angular Authentication and Authorization.**  
**Answer**: The user login credentials are passed to an authenticate API, which is present on the server. Post server-side validation of the credentials, a JWT (JSON Web Token) is returned. The JWT has information or attributes regarding the current user. The user is then identified with the given JWT. This is called authentication.

Post logging-in successfully, different users have a different level of access. While some may access everything, access for others might be restricted to only some resources. The level of access is authorization.

Here is a detailed post on Angular 7 – JWT Authentication Example & Tutorial: http://jasonwatmore.com/post/2018/11/16/angular-7-jwt-authentication-example-tutorial

**Question: Can you explain the concept of scope hierarchy in Angular?**  
**Answer:** Angular organizes the $scope objects into a hierarchy that is typically used by views. This is known as the scope hierarchy in Angular. It has a root scope that can further contain one or several scopes called child scopes.

In a scope hierarchy, each view has its own $scope. Hence, the variables set by a view’s view controller will remain hidden to other view controllers. Following is a typical representation of a Scope Hierarchy:

* Root $scope
  + $scope for Controller 1
  + $scope for Controller 2
  + …
  + ..
  + .
  + $scope for Controller n

**Question**: **How to generate a class in Angular 7 using CLI?**  
**Answer**:

ng generate class Dummy [options]

This will generate a class named Dummy.

**Question: Explain what is the difference between Angular and backbone.js?**  
**Answer:** Following are the various notable differences between Angular and Backbone.js

* **Architecture**

Backbone.js makes use of the MVP architecture and doesn’t offer any data binding process. Angular, on the contrary, works on the MVC architecture and makes use of two-way data binding for driving application activity.

* **Community Support**

Being backed by Google greatly ups the community support received by the Angular framework. Also, extensive documentation is available. Although Backbone.js has a good level of community support, it only documents on Underscore.js templates, not much else.

* **Data Binding**

Angular uses two-way data binding process and thus is a bit complex. Backbone.js, on the contrary, doesn’t have any data binding process and thus, has a simplistic API.

* **DOM**

The prime focus of Angular JS is upon valid HTML and dynamic elements that imitate the underlying data for rebuilding the DOM as per the specified rules and then works on the updated data records.

Backbone.js follows the direct DOM manipulation approach for representing data and application architecture changes.

* **Performance**

Thanks to its two-way data binding functionality, Angular offers an impactful performance for both small and large projects.

Backbone.js has a significant upper hand in performance over Angular in small data sets or small webpages. However, it is not recommended for larger webpages or large data sets due to the absence of any data binding process.

* **Templating**

Angular supports templating via dynamic HTML attributes. These are added to the document to develop an easy to understand application at a functional level. Unlike Angular, Backbone.js uses [Underscore.js](https://en.wikipedia.org/wiki/Underscore.js) templates that aren’t fully-featured as Angular templates.

* **The Approach to Testing**

The approach to testing varies greatly between Angular and Backbone.js due to the fact that while the former is preferred for building large applications the latter is ideal for developing smaller webpages or applications.

For Angular, unit testing is preferred and the testing process is smoother through the framework. In the case of Backbone.js, the absence of a data binding process allows for a swift testing experience for a single page and small applications.

* **Type**

Angular is an open-source JS-based front-end web application framework that extends HTML with new attributes. On the other hand, Backbone.js is a lightweight JavaScript library featuring a RESTful JSON interface and MVP framework.

**Question**: **How do Observables differ from Promises?**  
**Answer**: As soon as a [promise](http://andyshora.com/promises-angularjs-explained-as-cartoon.html) is made, the execution takes place. However, this is not the case with observables because they are lazy. This means that nothing happens until a subscription is made. While promises handle a single event, observable is a stream that allows passing of more than one event. A callback is made for each event in an observable.

**Question: Please explain the difference between Angular and AngularJS?**  
**Answer:** Various differences between Angular and AngularJS are stated as follows:

* **Architecture –**AngularJS supports the MVC design model. Angular relies on components and directives instead
* **Dependency Injection (DI) –**Angular supports a hierarchical Dependency Injection with unidirectional tree-based change detection. AngularJS doesn’t support DI
* **Expression Syntax –** In AngularJS, a specific ng directive is required for the image or property and an event. Angular, on the other hand, use () and [] for blinding an event and accomplishing property binding, respectively
* **Mobile Support –** AngularJS doesn’t have mobile support while Angular does have
* **Recommended Language –** While JavaScript is the recommended language for AngularJS, TypeScript is the recommended language for Angular
* **Routing –** For routing, AngularJS uses $routeprovider.when() whereas Angular uses @RouteConfig{(…)}
* **Speed –** The development effort and time are reduced significantly thanks to support for two-way data binding in AngularJS. Nonetheless, Angular is faster thanks to upgraded features
* **Structure –** With a simplified structure, Angular makes the development and maintenance of large applications easier. Comparatively, AngularJS has a less manageable structure
* **Support –** No official support or updates are available for the AngularJS. On the contrary, Angular has active support with updates rolling out every now and then

**Question**: **Observe the following image:**



**Question: Could you explain the concept of templates in Angular?**  
**Answer:** Written with HTML, templates in Angular contains Angular-specific attributes and elements. Combined with information coming from the controller and model, templates are then further rendered to cater the user with the dynamic view.

**Question: What should replace the “?”?**  
**Answer**: Directives. The image represents the types of directives in Angular; Attribute, structural, and custom.

**Question: Explain the difference between an Annotation and a Decorator in Angular?**  
**Answer:** In Angular, annotations are used for creating an annotation array. They are only metadata set of the class using the Reflect Metadata library.

Decorators in Angular are design patterns used for separating decoration or modification of some class without changing the original source code.

**Question: What are directives in Angular?**  
**Answer:** Directives are one of the core features of Angular. They allow an Angular developer to write new, application-specific HTML syntax. In actual, directives are functions that are executed by the Angular compiler when the same finds them in the DOM. Directives are of three types:

* Attribute Directives
* Component Directives
* Structural Directives

**Question**: **What are the building blocks of Angular?**  
**Answer**: There are essentially 9 building blocks of an Angular application. These are:

1. **Components –** A component controls one or more views. Each view is some specific section of the screen. Every Angular application has at least one component, known as the [root component](https://www.learnhowtoprogram.com/javascript/angular/angular-2-setup-root-component-root-module-and-more). It is bootstrapped inside the main module, known as the root module. A component contains application logic defined inside a class. This class is responsible for interacting with the view via an API of properties and methods.
2. **Data Binding –** The mechanism by which parts of a template coordinates with parts of a component is known as data binding. In order to let Angular know how to connect both sides (template and its component), the binding markup is added to the template HTML.
3. **Dependency Injection (DI) –** Angular makes use of DI to provide required dependencies to new components. Typically, dependencies required by a component are services. A component’s constructor parameters tell Angular about the services that a component requires. So, a dependency injection offers a way to supply fully-formed dependencies required by a new instance of a class.
4. **Directives –** The templates used by Angular are dynamic in nature. Directives are responsible for instructing Angular about how to transform the DOM when rendering a template. Actually, components are directives with a template. Other [types of directives](https://angular.io/guide/attribute-directives) are attribute and structural directives.
5. **Metadata –** In order to let Angular know how to process a class, metadata is attached to the class. For doing so decorators are used.
6. **Modules –** Also known as NgModules, a module is an organized block of code with a specific set of capabilities. It has a specific application domain or a workflow. Like components, any Angular application has at least one module. This is known as the root module. Typically, an Angular application has several modules.
7. **Routing –** An Angular router is responsible for interpreting a browser URL as an instruction to navigate to a client-generated view. The router is bound to links on a page to tell Angular to navigate the application view when a user clicks on it.
8. **Services –** A very broad category, a service can be anything ranging from a value and function to a feature that is required by an Angular app. Technically, a service is a class with a well-defined purpose.
9. **Template –** Each component’s view is associated with its companion template. A template in Angular is a form of HTML tags that lets Angular know that how it is meant to render the component.

**Question: Please explain the differences between Angular and jQuery?**  
**Answer:** The single biggest difference between Angular and jQuery is that while the former is a JS frontend framework, the latter is a JS library. Despite this, there are some similarities between the two, such as both features DOM manipulation and provides support for animation.

Nonetheless, notable differences between Angular and jQuery are:

* Angular has two-way data binding, jQuery does not
* Angular provides support for RESTful API while jQuery doesn’t
* jQuery doesn’t offer deep linking routing though Angular supports it
* There is no form validation in jQuery whereas it is present in Angular

**Question: Could you explain the difference between Angular expressions and JavaScript expressions?**  
**Answer:** Although both Angular expressions and JavaScript expressions can contain literals, operators, and variables, there are some notable dissimilarities between the two. Important differences between Angular expressions and JavaScript expressions are enlisted below:

* Angular expressions support filters while JavaScript expressions do not
* It is possible to write Angular expressions inside the HTML tags. JavaScript expressions, contrarily, can’t  be written inside the HTML tags
* While JavaScript expressions support conditionals, exceptions, and loops, Angular expressions don’t

**Question: Can you give us an overview of Angular architecture?**  
**Answer**: You can draw some like this:



Here is Angular Architecture in detail: https://angular.io/guide/architecture

**Question**: **What is Angular Material?**  
**Answer**: It is a UI component library. [Angular Material](https://material.angular.io/) helps in creating attractive, consistent, and fully functional web pages as well as web applications. It does so while following modern web design principles, including browser portability and graceful degradation.

**Question**: **What is AOT (Ahead-Of-Time) Compilation?**  
**Answer**: Each Angular app gets compiled internally. The Angular compiler takes in the JS code, compiles it and then produces some JS code. This happens only once per occasion per user. It is known as AOT (Ahead-Of-Time) compilation.

**Question**: **What is Data Binding? How many ways it can be done?**  
**Answer**: In order to connect application data with the DOM (Data Object Model), data binding is used. It happens between the template (HTML) and component (TypeScript). There are 3 ways to achieve data binding:

1. Event Binding – Enables the application to respond to user input in the target environment
2. Property Binding – Enables interpolation of values computed from application data into the HTML
3. Two-way Binding – Changes made in the application state gets automatically reflected in the view and vice-versa. The ngModel directive is used for achieving this type of data binding.

**Question**: **What is demonstrated by the arrow in the following image?**

  
**Answer**: This represents a dependency injection or DI.

**Question: Can you draw a comparison between the service() and the factory() functions?**  
**Answer:** Used for the business layer of the application, the *service()* function operates as a constructor function. The function is invoked at runtime using the *new* keyword.

Although the *factory()* function works in pretty much the same way as the *service()* function does, the former is more flexible and powerful. In actual, the *factory()* function are design patterns that help in creating objects.

**Question: Please explain the digest cycle in Angular?**  
**Answer:** The process of monitoring the watchlist in order to track changes in the value of the watch variable is termed the digest cycle in Angular. The previous and present versions of the scope model values are compared in each digest cycle.

Although the digest cycle process gets triggered implicitly, it is possible to start it manually by using the *$apply()* function.

**Question: Could you explain the various types of filters in Angular.**  
**Answer:** In order to format the value of expression so that it can be displayed to the user, Angular has filters. It is possible to add these filters to the controllers, directives, services, or templates. Angular also provides support for creating custom filters.

Organizing data in such a way so that it is displayed only when certain criteria are fulfilled is made possible using filters. Filters are added to the expressions using the pipe ‘|’ character. Various types of Angular filters are enumerated as follows:

* *currency* – Formats a number to the currency format
* *date* – Formats a data to some specific format
* *filter* – Selects a subset of items from an array
* *json* – Formats an object to a JSON string
* *limitTo* – Limits an array or string into a specified number of characters or elements
* *lowercase* – Formats a string to lowercase
* *number* – Formats a number to a string
* *orderBy* – Orders an array by an expression

**Question**: **What is new in Angular 6?**  
**Answer**: Here are some of the new aspects introduced in Angular 6:

* Angular Elements – It allows converting Angular components into web components and embeds the same in some non-Angular application
* Tree Shakeable Provider – Angular 6 introduces a new way of registering a provider directly inside the @Injectable() decorator. It is achieved by using the providedIn attribute
* RxJS 6 – Angular 6 makes use of RxJS 6 internally
* i18n (internationalization) – Without having to build the application once per locale, any Angular application can have “runtime i18n”

**Question**: **What is ngOnInit ()? How to define it?**  
**Answer**: ngOnInit () is a lifecycle hook that is called after Angular has finished initializing all data-bound properties of a directive. It is defined as:

Interface OnInit {

          ngOnInit () : void

     }

**Question**: **What is SPA** **(Single Page Application) in Angular? Contrast SPA technology with traditional web technology?**  
**Answer**: In the SPA technology, only a single page, which is index.HTML, is maintained although the URL keeps on changing. Unlike the traditional web technology, SPA technology is faster and easy to develop as well.

In the conventional web technology, as soon as a client requests a webpage, the server sends the resource. However, when again the client requests for another page, the server responds again with sending the requested resource. The problem with this technology is that it requires a lot of time.

**Question**: **What is the code for creating a decorator?**  
**Answer**: We create a decorator called Dummy:

    function Dummy(target) {

       dummy.log('This decorator is Dummy', target);

    }

**Question**: **What is the process called by which TypeScript code is converted into JavaScript code?**  
**Answer**: It is called Transpiling. Even though TypeScript is used for writing code in Angular applications, it gets internally transpiled into equivalent JavaScript.

**Question**: **What is ViewEncapsulation and how many ways are there do to do it in Angular?**  
**Answer**: To put simply, ViewEncapsulation determines whether the styles defined in a particular component will affect the entire application or not. Angular supports 3 types of ViewEncapsulation:

* Emulated – Styles used in other HTML spread to the component
* Native – Styles used in other HTML doesn’t spread to the component
* None – Styles defined in a component are visible to all components of the application

1. **What is Angular Framework?**

Angular is a **TypeScript-based open-source** front-end platform that makes it easy to build applications with in web/mobile/desktop. The major features of this framework such as declarative templates, dependency injection, end to end tooling, and many more other features are used to ease the development.

1. **What is the difference between AngularJS and Angular?**

Angular is a completely revived component-based framework in which an application is a tree of individual components.

Some of the major difference in tabular form

| **AngularJS** | **Angular** |
| --- | --- |
| It is based on MVC architecture | This is based on Service/Controller |
| This uses use JavaScript to build the application | Introduced the typescript to write the application |
| Based on controllers concept | This is a component based UI approach |
| Not a mobile friendly framework | Developed considering mobile platform |
| Difficulty in SEO friendly application development | Ease to create SEO friendly applications |

1. **What is TypeScript?**

TypeScript is a typed superset of JavaScript created by Microsoft that adds optional types, classes, async/await, and many other features, and compiles to plain JavaScript. Angular built entirely in TypeScript and used as a primary language. You can install it globally as

npm install -g typescript

Let's see a simple example of TypeScript usage,

function greeter(person: string) {

return "Hello, " + person;

}

let user = "Sudheer";

document.body.innerHTML = greeter(user);

The greeter method allows only string type as argument.

1. **Write a pictorial diagram of Angular architecture?**

The main building blocks of an Angular application is shown in the below diagram

1. **What are the key components of Angular?**

Angular has the below key components,

* 1. **Component:** These are the basic building blocks of angular application to control HTML views.
  2. **Modules:** An angular module is set of angular basic building blocks like component, directives, services etc. An application is divided into logical pieces and each piece of code is called as "module" which perform a single task.
  3. **Templates:** This represent the views of an Angular application.
  4. **Services:** It is used to create components which can be shared across the entire application.
  5. **Metadata:** This can be used to add more data to an Angular class.

1. **What are directives?**

Directives add behaviour to an existing DOM element or an existing component instance.

import { Directive, ElementRef, Input } from '@angular/core';

@Directive({ selector: '[myHighlight]' })

export class HighlightDirective {

constructor(el: ElementRef) {

el.nativeElement.style.backgroundColor = 'yellow';

}

}

Now this directive extends HTML element behavior with a yellow background as below

<p myHighlight>Highlight me!</p>

1. **What are components?**

Components are the most basic UI building block of an Angular app which formed a tree of Angular components. These components are subset of directives. Unlike directives, components always have a template and only one component can be instantiated per an element in a template. Let's see a simple example of Angular component

import { Component } from '@angular/core';

@Component ({

selector: 'my-app',

template: ` <div>

<h1>{{title}}</h1>

<div>Learn Angular6 with examples</div>

</div> `,

})

export class AppComponent {

title: string = 'Welcome to Angular world';

}

1. **What are the differences between Component and Directive?**

In a short note, A component(@component) is a directive-with-a-template.

Some of the major differences are mentioned in a tabular form

| **Component** | **Directive** |
| --- | --- |
| To register a component we use @Component meta-data annotation | To register directives we use @Directive meta-data annotation |
| Components are typically used to create UI widgets | Directive is used to add behavior to an existing DOM element |
| Component is used to break up the application into smaller components | Directive is use to design re-usable components |
| Only one component can be present per DOM element | Many directives can be used per DOM element |
| @View decorator or templateurl/template are mandatory | Directive doesn't use View |

1. **What is a template?**

A template is a HTML view where you can display data by binding controls to properties of an Angular component. You can store your component's template in one of two places. You can define it inline using the template property, or you can define the template in a separate HTML file and link to it in the component metadata using the @Component decorator's templateUrl property. **Using inline template with template syntax,**

import { Component } from '@angular/core';

@Component ({

selector: 'my-app',

template: '

<div>

<h1>{{title}}</h1>

<div>Learn Angular</div>

</div>

'

})

export class AppComponent {

title: string = 'Hello World';

}

**Using separate template file such as app.component.html**

import { Component } from '@angular/core';

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

})

export class AppComponent {

title: string = 'Hello World';

}

1. **What is a module?**

Modules are logical boundaries in your application and the application is divided into separate modules to separate the functionality of your application. Lets take an example of **app.module.ts** root module declared with **@NgModule**decorator as below,

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

@NgModule ({

imports: [ BrowserModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

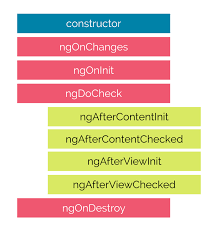
})

export class AppModule { }

The NgModule decorator has three options

* 1. The imports option is used to import other dependent modules. The BrowserModule is required by default for any web based angular application
  2. The declarations option is used to define components in the respective module
  3. The bootstrap option tells Angular which Component to bootstrap in the application

1. **What are lifecycle hooks available?**

Angular application goes through an entire set of processes or has a lifecycle right from its initiation to the end of the application. The representation of lifecycle in pictorial representation as follows, 

The description of each lifecycle method is as below,

* 1. **ngOnChanges:** When the value of a data bound property changes, then this method is called.
  2. **ngOnInit:** This is called whenever the initialization of the directive/component after Angular first displays the data-bound properties happens.
  3. **ngDoCheck:** This is for the detection and to act on changes that Angular can't or won't detect on its own.
  4. **ngAfterContentInit:** This is called in response after Angular projects external content into the component's view.
  5. **ngAfterContentChecked:** This is called in response after Angular checks the content projected into the component.
  6. **ngAfterViewInit:** This is called in response after Angular initializes the component's views and child views.
  7. **ngAfterViewChecked:** This is called in response after Angular checks the component's views and child views.
  8. **ngOnDestroy:** This is the cleanup phase just before Angular destroys the directive/component.

1. **What is a data binding?**

Data binding is a core concept in Angular and allows to define communication between a component and the DOM, making it very easy to define interactive applications without worrying about pushing and pulling data. There are four forms of data binding(divided as 3 categories) which differ in the way the data is flowing.

* 1. **From the Component to the DOM:** **Interpolation:** {{ value }}: Adds the value of a property from the component

<li>Name: {{ user.name }}</li>

<li>Address: {{ user.address }}</li>

**Property binding:** [property]=”value”: The value is passed from the component to the specified property or simple HTML attribute

<input type="email" [value]="user.email">

* 1. **From the DOM to the Component:** **Event binding: (event)=”function”:** When a specific DOM event happens (eg.: click, change, keyup), call the specified method in the component

<button (click)="logout()"></button>

* 1. **Two-way binding:** **Two-way data binding:** [(ngModel)]=”value”: Two-way data binding allows to have the data flow both ways. For example, in the below code snippet, both the email DOM input and component email property are in sync

<input type="email" [(ngModel)]="user.email">

1. **What is metadata?**

Metadata is used to decorate a class so that it can configure the expected behavior of the class. The metadata is represented by decorators

* 1. **Class decorators**, e.g. @Component and @NgModule

import { NgModule, Component } from '@angular/core';

@Component({

selector: 'my-component',

template: '<div>Class decorator</div>',

})

export class MyComponent {

constructor() {

console.log('Hey I am a component!');

}

}

@NgModule({

imports: [],

declarations: [],

})

export class MyModule {

constructor() {

console.log('Hey I am a module!');

}

}

* 1. **Property decorators** Used for properties inside classes, e.g. @Input and @Output

import { Component, Input } from '@angular/core';

@Component({

selector: 'my-component',

template: '<div>Property decorator</div>'

})

export class MyComponent {

@Input()

title: string;

}

* 1. **Method decorators** Used for methods inside classes, e.g. @HostListener

import { Component, HostListener } from '@angular/core';

@Component({

selector: 'my-component',

template: '<div>Method decorator</div>'

})

export class MyComponent {

@HostListener('click', ['$event'])

onHostClick(event: Event) {

// clicked, `event` available

}

}

* 1. **Parameter decorators** Used for parameters inside class constructors, e.g. @Inject

import { Component, Inject } from '@angular/core';

import { MyService } from './my-service';

@Component({

selector: 'my-component',

template: '<div>Parameter decorator</div>'

})

export class MyComponent {

constructor(@Inject(MyService) myService) {

console.log(myService); // MyService

}

}

1. **What is angular CLI?**

Angular CLI(**Command Line Interface**) is a command line interface to scaffold and build angular apps using nodejs style (commonJs) modules. You need to install using below npm command,

npm install @angular/cli@latest

Below are the list of few commands, which will come handy while creating angular projects

* 1. **Creating New Project:** ng new
  2. **Generating Components, Directives & Services:** ng generate/g The different types of commands would be,
  3. ng generate class my-new-class: add a class to your application
  4. ng generate component my-new-component: add a component to your application
  5. ng generate directive my-new-directive: add a directive to your application
  6. ng generate enum my-new-enum: add an enum to your application
  7. ng generate module my-new-module: add a module to your application
  8. ng generate pipe my-new-pipe: add a pipe to your application
  9. ng generate service my-new-service: add a service to your application
  10. **Running the Project:** ng serve

1. **What is the difference between constructor and ngOnInit?**

TypeScript classes has a default method called constructor which is normally used for the initialization purpose. Whereas ngOnInit method is specific to Angular, especially used to define Angular bindings. Even though constructor getting called first, it is preferred to move all of your Angular bindings to ngOnInit method. In order to use ngOnInit, you need to implement OnInit interface as below,

export class App implements OnInit{

constructor(){

//called first time before the ngOnInit()

}

ngOnInit(){

//called after the constructor and called after the first ngOnChanges()

}

}

1. **What is a service?**

A service is used when a common functionality needs to be provided to various modules. Services allow for greater separation of concerns for your application and better modularity by allowing you to extract common functionality out of components. Let's create a repoService which can be used across components,

import { Injectable } from '@angular/core';

import { Http } from '@angular/http';

@Injectable({ // The Injectable decorator is required for dependency injection to work

// providedIn option registers the service with a specific NgModule

providedIn: 'root', // This declares the service with the root app (AppModule)

})

export class RepoService{

constructor(private http: Http){

}

fetchAll(){

return this.http.get('https://api.github.com/repositories');

}

}

The above service uses Http service as a dependency.

1. **What is dependency injection in Angular?**

Dependency injection (DI), is an important application design pattern in which a class asks for dependencies from external sources rather than creating them itself. Angular comes with its own dependency injection framework for resolving dependencies( services or objects that a class needs to perform its function).So you can have your services depend on other services throughout your application.

1. **How is Dependency Hierarchy formed?**
2. **What is the purpose of async pipe?**

The AsyncPipe subscribes to an observable or promise and returns the latest value it has emitted. When a new value is emitted, the pipe marks the component to be checked for changes. Let's take a time observable which continuously updates the view for every 2 seconds with the current time.

@Component({

selector: 'async-observable-pipe',

template: `<div><code>observable|async</code>:

Time: {{ time | async }}</div>`

})

export class AsyncObservablePipeComponent {

time = new Observable(observer =>

setInterval(() => observer.next(new Date().toString()), 2000)

);

}

1. **What is the option to choose between inline and external template file?**

You can store your component's template in one of two places. You can define it inline using the **template** property, or you can define the template in a separate HTML file and link to it in the component metadata using the **@Component**decorator's **templateUrl** property. The choice between inline and separate HTML is a matter of taste, circumstances, and organization policy. But normally we use inline template for small portion of code and external template file for bigger views. By default, the Angular CLI generates components with a template file. But you can override that with the below command,

ng generate component hero -it

1. **What is the purpose of ngFor directive?**

We use Angular ngFor directive in the template to display each item in the list. For example, here we iterate over list of users,

<li \*ngFor="let user of users">

{{ user }}

</li>

The user variable in the ngFor double-quoted instruction is a **template input variable**

1. **What is the purpose of ngIf directive?**

Sometimes an app needs to display a view or a portion of a view only under specific circumstances. The Angular ngIf directive inserts or removes an element based on a truthy/falsy condition. Let's take an example to display a message if the user age is more than 18,

<p \*ngIf="user.age > 18">You are not eligible for student pass!</p>

**Note:** Angular isn't showing and hiding the message. It is adding and removing the paragraph element from the DOM. That improves performance, especially in the larger projects with many data bindings.

1. **What happens if you use script tag inside template?**

Angular recognizes the value as unsafe and automatically sanitizes it, which removes the **<script>** tag but keeps safe content such as the text content of the <script> tag. This way it eliminates the risk of script injection attacks. If you still use it then it will be ignored and a warning appears in the browser console. Let's take an example of innerHtml property binding which causes XSS vulnerability,

export class InnerHtmlBindingComponent {

// For example, a user/attacker-controlled value from a URL.

htmlSnippet = 'Template <script>alert("0wned")</script> <b>Syntax</b>';

}

1. **What is interpolation?**

Interpolation is a special syntax that Angular converts into property binding. It’s a convenient alternative to property binding. It is represented by double curly braces({{}}). The text between the braces is often the name of a component property. Angular replaces that name with the string value of the corresponding component property. Let's take an example,

<h3>

{{title}}

<img src="{{url}}" style="height:30px">

</h3>

In the example above, Angular evaluates the title and url properties and fills in the blanks, first displaying a bold application title and then a URL.

1. **What are template expressions?**

A template expression produces a value similar to any Javascript expression. Angular executes the expression and assigns it to a property of a binding target; the target might be an HTML element, a component, or a directive. In the property binding, a template expression appears in quotes to the right of the = symbol as in [property]="expression". In interpolation syntax, the template expression is surrounded by double curly braces. For example, in the below interpolation, the template expression is {{username}},

<h3>{{username}}, welcome to Angular</h3>

The below javascript expressions are prohibited in template expression

* 1. assignments (=, +=, -=, ...)
  2. new
  3. chaining expressions with ; or ,
  4. increment and decrement operators (++ and --)

1. **What are template statements?**

A template statement responds to an event raised by a binding target such as an element, component, or directive. The template statements appear in quotes to the right of the = symbol like **(event)="statement"**. Let's take an example of button click event's statement

<button (click)="editProfile()">Edit Profile</button>

In the above expression, editProfile is a template statement. The below JavaScript syntax expressions are not allowed.

* 1. new
  2. increment and decrement operators, ++ and --
  3. operator assignment, such as += and -=
  4. the bitwise operators | and &
  5. the template expression operators

1. **How do you categorize data binding types?**

Binding types can be grouped into three categories distinguished by the direction of data flow. They are listed as below,

* 1. From the source-to-view
  2. From view-to-source
  3. View-to-source-to-view

The possible binding syntax can be tabularized as below,

| **Data direction** | **Syntax** | **Type** |
| --- | --- | --- |
| From the source-to-view(One-way) | 1. {{expression}} 2. [target]="expression" 3. bind-target="expression" | Interpolation, Property, Attribute, Class, Style |
| From view-to-source(One-way) | 1. (target)="statement" 2. on-target="statement" | Event |
| View-to-source-to-view(Two-way) | 1. [(target)]="expression" 2. bindon-target="expression" | Two-way |

1. **What are pipes?**

A pipe takes in data as input and transforms it to a desired output. For example, let us take a pipe to transform a component's birthday property into a human-friendly date using **date** pipe.

import { Component } from '@angular/core';

@Component({

selector: 'app-birthday',

template: `<p>Birthday is {{ birthday | date }}</p>`

})

export class BirthdayComponent {

birthday = new Date(1987, 6, 18); // June 18, 1987

}

1. **What is a parameterized pipe?**

A pipe can accept any number of optional parameters to fine-tune its output. The parameterized pipe can be created by declaring the pipe name with a colon ( : ) and then the parameter value. If the pipe accepts multiple parameters, separate the values with colons. Let's take a birthday example with a particular format(dd/mm/yyyy):

import { Component } from '@angular/core';

@Component({

selector: 'app-birthday',

template: `<p>Birthday is {{ birthday | date:'dd/mm/yyyy'}}</p>` // 18/06/1987

})

export class BirthdayComponent {

birthday = new Date(1987, 6, 18);

}

**Note:** The parameter value can be any valid template expression, such as a string literal or a component property.

1. **How do you chain pipes?**

You can chain pipes together in potentially useful combinations as per the needs. Let's take a birthday property which uses date pipe(along with parameter) and uppercase pipes as below

import { Component } from '@angular/core';

@Component({

selector: 'app-birthday',

template: `<p>Birthday is {{ birthday | date:'fullDate' | uppercase}} </p>` // THURSDAY, JUNE 18, 1987

})

export class BirthdayComponent {

birthday = new Date(1987, 6, 18);

}

1. **What is a custom pipe?**

Apart from built-inn pipes, you can write your own custom pipe with the below key characteristics,

* 1. A pipe is a class decorated with pipe metadata **@Pipe** decorator, which you import from the core Angular library For example,

@Pipe({name: 'myCustomPipe'})

* 1. The pipe class implements the **PipeTransform** interface's transform method that accepts an input value followed by optional parameters and returns the transformed value. The structure of pipeTransform would be as below,

interface PipeTransform {

transform(value: any, ...args: any[]): any

}

* 1. The @Pipe decorator allows you to define the pipe name that you'll use within template expressions. It must be a valid JavaScript identifier.

template: `{{someInputValue | myCustomPipe: someOtherValue}}`

1. **Give an example of custom pipe?**

You can create custom reusable pipes for the transformation of existing value. For example, let us create a custom pipe for finding file size based on an extension,

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({name: 'customFileSizePipe'})

export class FileSizePipe implements PipeTransform {

transform(size: number, extension: string = 'MB'): string {

return (size / (1024 \* 1024)).toFixed(2) + extension;

}

}

Now you can use the above pipe in template expression as below,

template: `

<h2>Find the size of a file</h2>

<p>Size: {{288966 | customFileSizePipe: 'GB'}}</p>

`

1. **What is the difference between pure and impure pipe?**

A pure pipe is only called when Angular detects a change in the value or the parameters passed to a pipe. For example, any changes to a primitive input value (String, Number, Boolean, Symbol) or a changed object reference (Date, Array, Function, Object). An impure pipe is called for every change detection cycle no matter whether the value or parameters changes. i.e, An impure pipe is called often, as often as every keystroke or mouse-move.

1. **What is a bootstrapping module?**

Every application has at least one Angular module, the root module that you bootstrap to launch the application is called as bootstrapping module. It is commonly known as AppModule. The default structure of AppModule generated by AngularCLI would be as follows,

/\* JavaScript imports \*/

import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';

import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';

/\* the AppModule class with the @NgModule decorator \*/

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

FormsModule,

HttpClientModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

1. **What are observables?**

Observables are declarative which provide support for passing messages between publishers and subscribers in your application. They are mainly used for event handling, asynchronous programming, and handling multiple values. In this case, you define a function for publishing values, but it is not executed until a consumer subscribes to it. The subscribed consumer then receives notifications until the function completes, or until they unsubscribe.

1. **What is HttpClient and its benefits?**

Most of the Front-end applications communicate with backend services over HTTP protocol using either XMLHttpRequest interface or the fetch() API. Angular provides a simplified client HTTP API known as **HttpClient** which is based on top of XMLHttpRequest interface. This client is avaialble from @angular/common/http package. You can import in your root module as below,

import { HttpClientModule } from '@angular/common/http';

The major advantages of HttpClient can be listed as below,

* 1. Contains testability features
  2. Provides typed request and response objects
  3. Intercept request and response
  4. Supports Observalbe APIs
  5. Supports streamlined error handling

1. **Explain on how to use HttpClient with an example?**

Below are the steps need to be followed for the usage of HttpClient.

* 1. Import HttpClient into root module:

import { HttpClientModule } from '@angular/common/http';

@NgModule({

imports: [

BrowserModule,

// import HttpClientModule after BrowserModule.

HttpClientModule,

],

......

})

export class AppModule {}

* 1. Inject the HttpClient into the application: Let's create a userProfileService(userprofile.service.ts) as an example. It also defines get method of HttpClient

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

const userProfileUrl: string = 'assets/data/profile.json';

@Injectable()

export class UserProfileService {

constructor(private http: HttpClient) { }

getUserProfile() {

return this.http.get(this.userProfileUrl);

}

}

* 1. Create a component for subscribing service: Let's create a component called UserProfileComponent(userprofile.component.ts) which inject UserProfileService and invokes the service method,

fetchUserProfile() {

this.userProfileService.getUserProfile()

.subscribe((data: User) => this.user = {

id: data['userId'],

name: data['firstName'],

city: data['city']

});

}

Since the above service method returns an Observable which needs to be subscribed in the component.

1. **How can you read full response?**

The response body doesn't may not return full response data because sometimes servers also return special headers or status code which which are important for the application workflow. Inorder to get full response, you should use observe option from HttpClient,

getUserResponse(): Observable<HttpResponse<User>> {

return this.http.get<User>(

this.userUrl, { observe: 'response' });

}

Now HttpClient.get() method returns an Observable of typed HttpResponse rather than just the JSON data.

1. **How do you perform Error handling?**

If the request fails on the server or failed to reach the server due to network issues then HttpClient will return an error object instead of a successful reponse. In this case, you need to handle in the component by passing error object as a second callback to subscribe() method. Let's see how it can be handled in the component with an example,

fetchUser() {

this.userService.getProfile()

.subscribe(

(data: User) => this.userProfile = { ...data }, // success path

error => this.error = error // error path

);

}

It is always a good idea to give the user some meaningful feedback instead of displaying the raw error object returned from HttpClient.

1. **What is RxJS?**

RxJS is a library for composing asynchronous and callback-based code in a functional, reactive style using Observables. Many APIs such as HttpClient produce and consume RxJS Observables and also uses operators for processing observables. For example, you can import observables and operators for using HttpClient as below,

import { Observable, throwError } from 'rxjs';

import { catchError, retry } from 'rxjs/operators';

1. **What is subscribing?**

An Observable instance begins publishing values only when someone subscribes to it. So you need to subscribe by calling the **subscribe()** method of the instance, passing an observer object to receive the notifications. Let's take an example of creating and subscribing to a simple observable, with an observer that logs the received message to the console.

Creates an observable sequence of 5 integers, starting from 1

const source = range(1, 5);

// Create observer object

const myObserver = {

next: x => console.log('Observer got a next value: ' + x),

error: err => console.error('Observer got an error: ' + err),

complete: () => console.log('Observer got a complete notification'),

};

// Execute with the observer object and Prints out each item

myObservable.subscribe(myObserver);

// => Observer got a next value: 1

// => Observer got a next value: 2

// => Observer got a next value: 3

// => Observer got a next value: 4

// => Observer got a next value: 5

// => Observer got a complete notification

1. **What is an observable?**

An Observable is a unique Object similar to a Promise that can help manage async code. Observables are not part of the JavaScript language so we need to rely on a popular Observable library called RxJS. The observables are created using new keyword. Let see the simple example of observable,

import { Observable } from 'rxjs';

const observable = new Observable(observer => {

setTimeout(() => {

observer.next('Hello from a Observable!');

}, 2000);

});

1. **What is an observer?**

Observer is an interface for a consumer of push-based notifications delivered by an Observable. It has below structure,

interface Observer<T> {

closed?: boolean;

next: (value: T) => void;

error: (err: any) => void;

complete: () => void;

}

A handler that implements the Observer interface for receiving observable notifications will be passed as a parameter for observable as below,

myObservable.subscribe(myObserver);

**Note:** If you don't supply a handler for a notification type, the observer ignores notifications of that type.

1. **What is the difference between promise and observable?**

Below are the list of differences between promise and observable,

| **Observable** | **Promise** |
| --- | --- |
| Declarative: Computation does not start until subscription so that they can be run whenever you need the result | Execute immediately on creation |
| Provide multiple values over time | Provide only one |
| Subscribe method is used for error handling which makes centralized and predictable error handling | Push errors to the child promises |
| Provides chaining and subscription to handle complex applications | Uses only .then() clause |

1. **What is multicasting?**

Multi-casting is the practice of broadcasting to a list of multiple subscribers in a single execution. Let's demonstrate the multi-casting feature,

var source = Rx.Observable.from([1, 2, 3]);

var subject = new Rx.Subject();

var multicasted = source.multicast(subject);

// These are, under the hood, `subject.subscribe({...})`:

multicasted.subscribe({

next: (v) => console.log('observerA: ' + v)

});

multicasted.subscribe({

next: (v) => console.log('observerB: ' + v)

});

// This is, under the hood, `s

1. **How do you perform error handling in observables?**

You can handle errors by specifying an **error callback** on the observer instead of relying on try/catch which are ineffective in asynchronous environment. For example, you can define error callback as below,

myObservable.subscribe({

next(num) { console.log('Next num: ' + num)},

error(err) { console.log('Received an errror: ' + err)}

});

1. **What is the short hand notation for subscribe method?**

The subscribe() method can accept callback function definitions in line, for next, error, and complete handlers is known as short hand notation or Subscribe method with positional arguments. For example, you can define subscribe method as below,

myObservable.subscribe(

x => console.log('Observer got a next value: ' + x),

err => console.error('Observer got an error: ' + err),

() => console.log('Observer got a complete notification')

);

1. **What are the utility functions provided by RxJS?**

The RxJS library also provides below utility functions for creating and working with observables.

* 1. Converting existing code for async operations into observables
  2. Iterating through the values in a stream
  3. Mapping values to different types
  4. Filtering streams
  5. Composing multiple streams

1. **What are observable creation functions?**

RxJS provides creation functions for the process of creating observables from things such as promises, events, timers and Ajax requests. Let us explain each of them with an example,

* 1. Create an observable from a promise

import { from } from 'rxjs'; // from function

const data = from(fetch('/api/endpoint')); //Created from Promise

data.subscribe({

next(response) { console.log(response); },

error(err) { console.error('Error: ' + err); },

complete() { console.log('Completed'); }

});

* 1. Create an observable that creates an AJAX request

import { ajax } from 'rxjs/ajax'; // ajax function

const apiData = ajax('/api/data'); // Created from AJAX request

// Subscribe to create the request

apiData.subscribe(res => console.log(res.status, res.response));

* 1. Create an observable from a counter

import { interval } from 'rxjs'; // interval function

const secondsCounter = interval(1000); // Created from Counter value

secondsCounter.subscribe(n =>

console.log(`Counter value: ${n}`));

* 1. Create an observable from an event

import { fromEvent } from 'rxjs';

const el = document.getElementById('custom-element');

const mouseMoves = fromEvent(el, 'mousemove');

const subscription = mouseMoves.subscribe((e: MouseEvent) => {

console.log(`Coordnitaes of mouse pointer: ${e.clientX} \* ${e.clientY}`);

});

1. **What will happen if you do not supply handler for observer?**

Normally an observer object can define any combination of next, error and complete notification type handlers. If you don't supply a handler for a notification type, the observer just ignores notifications of that type.

1. **What are angular elements?**

Angular elements are Angular components packaged as **custom elements**(a web standard for defining new HTML elements in a framework-agnostic way). Angular Elements hosts an Angular component, providing a bridge between the data and logic defined in the component and standard DOM APIs, thus, providing a way to use Angular components in non-Angular environments.

1. **What is the browser support of Angular Elements?**

Since Angular elements are packaged as custom elements the browser support of angular elements is same as custom elements support. This feature is is currently supported natively in a number of browsers and pending for other browsers.

| **Browser** | **Angular Element Support** |
| --- | --- |
| Chrome | Natively supported |
| Opera | Natively supported |
| Safari | Natively supported |
| Firefox | Natively supported from 63 version onwards. You need to enable dom.webcomponents.enabled and dom.webcomponents.customelements.enabled in older browsers |
| Edge | Currently it is in progress |

1. **What are custom elements?**

Custom elements (or Web Components) are a Web Platform feature which extends HTML by allowing you to define a tag whose content is created and controlled by JavaScript code. The browser maintains a CustomElementRegistry of defined custom elements, which maps an instantiable JavaScript class to an HTML tag. Currently this feature is supported by Chrome, Firefox, Opera, and Safari, and available in other browsers through polyfills.

1. **Do I need to bootstrap custom elements?**

No, custom elements bootstrap (or start) automatically when they are added to the DOM, and are automatically destroyed when removed from the DOM. Once a custom element is added to the DOM for any page, it looks and behaves like any other HTML element, and does not require any special knowledge of Angular.

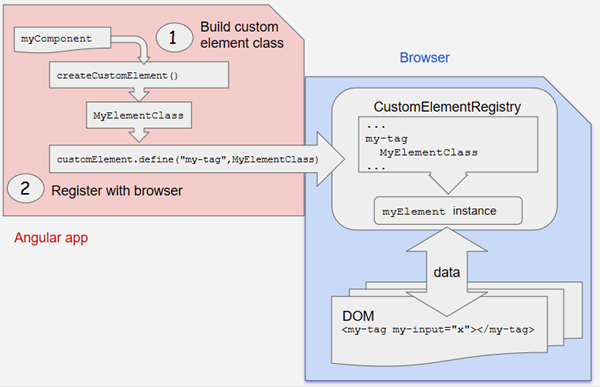
1. **Explain how custom elements works internally?**

Below are the steps in an order about custom elements functionality,

* 1. **App registers custom element with browser:** Use the createCustomElement() function to convert a component into a class that can be registered with the browser as a custom element.
  2. **App adds custom element to DOM:** Add custom element just like a built-in HTML element directly into the DOM.
  3. **Browser instantiate component based class:** Browser creates an instance of the registered class and adds it to the DOM.
  4. **Instance provides content with data binding and change detection:** The content with in template is rendered using the component and DOM data. The flow chart of the custom elements functionality would be as follows, 

1. **How to transfer components to custom elements?**

Transforming components to custom elements involves **two** major steps,

* 1. **Build custom element class:** Angular provides the createCustomElement() function for converting an Angular component (along with its dependencies) to a custom element. The conversion process implements NgElementConstructor interface, and creates a constructor class which is used to produce a self-bootstrapping instance of Angular component.
  2. **Register element class with browser:** It uses customElements.define() JS function, to register the configured constructor and its associated custom-element tag with the browser's CustomElementRegistry. When the browser encounters the tag for the registered element, it uses the constructor to create a custom-element instance. The detailed structure would be as follows, 

1. **What are the mapping rules between Angular component and custom element?**

The Component properties and logic maps directly into HTML attributes and the browser's event system. Let us describe them in two steps,

* 1. The createCustomElement() API parses the component input properties with corresponding attributes for the custom element. For example, component @Input('myInputProp') converted as custom element attribute my-input-prop.
  2. The Component outputs are dispatched as HTML Custom Events, with the name of the custom event matching the output name. For example, component @Output() valueChanged = new EventEmitter() converted as custom element with dispatch event as "valueChanged".

1. **How do you define typings for custom elements?**

You can use the NgElement and WithProperties types exported from @angular/elements. Let's see how it can be applied by comparing with Angular component, The simple container with input property would be as below,

@Component(...)

class MyContainer {

@Input() message: string;

}

After applying types typescript validates input value and their types,

const container = document.createElement('my-container') as NgElement & WithProperties<{message: string}>;

container.message = 'Welcome to Angular elements!';

container.message = true; // <-- ERROR: TypeScript knows this should be a string.

container.greet = 'News'; // <-- ERROR: TypeScript knows there is no `greet` property on `container`.

1. **What are dynamic components?**

Dynamic components are the components in which components location in the application is not defined at build time.i.e, They are not used in any angular template. But the component is instantiated and placed in the application at runtime.

1. **What are the various kinds of directives?**

There are mainly three kinds of directives.

* 1. **Components** — These are directives with a template.
  2. **Structural directives** — These directives change the DOM layout by adding and removing DOM elements.
  3. **Attribute directives** — These directives change the appearance or behavior of an element, component, or another directive.

1. **How do you create directives using CLI?**

You can use CLI command ng generate directive to create the directive class file. It creates the source file(src/app/components/directivename.directive.ts), the respective test file(.spec.ts) and declare the directive class file in root module.

1. **Give an example for attribute directives?**

Let's take simple highlighter behavior as a example directive for DOM element. You can create and apply the attribute directive using below steps,

* 1. Create HighlightDirective class with the file name src/app/highlight.directive.ts. In this file, we need to import **Directive** from core library to apply the metadata and **ElementRef** in the directive's constructor to inject a reference to the host DOM element ,

import { Directive, ElementRef } from '@angular/core';

@Directive({

selector: '[appHighlight]'

})

export class HighlightDirective {

constructor(el: ElementRef) {

el.nativeElement.style.backgroundColor = 'red';

}

}

* 1. Apply the attribute directive as an attribute to the host element(for example,

)

<p appHighlight>Highlight me!</p>

* 1. Run the application to see the highlight behavior on paragraph element

ng serve

1. **What is Angular Router?**

Angular Router is a mechanism in which navigation happens from one view to the next as users perform application tasks. It borrows the concepts or model of browser's application navigation.

1. **What is the purpose of base href tag?**

The routing application should add element to the index.html as the first child in the tag inorder to indicate how to compose navigation URLs. If app folder is the application root then you can set the href value as below

<base href="/">

1. **What are the router imports?**

The Angular Router which represents a particular component view for a given URL is not part of Angular Core. It is available in library named @angular/router to import required router components. For example, we import them in app module as below,

import { RouterModule, Routes } from '@angular/router';

1. **What is router outlet?**

The RouterOutlet is a directive from the router library and it acts as a placeholder that marks the spot in the template where the router should display the components for that outlet. Router outlet is used like a component,

<router-outlet></router-outlet>

<!-- Routed components go here -->

1. **What are router links?**

The RouterLink is a directive on the anchor tags give the router control over those elements. Since the navigation paths are fixed, you can assign string values to router-link directive as below,

<h1>Angular Router</h1>

<nav>

<a routerLink="/todosList" >List of todos</a>

<a routerLink="/completed" >Completed todos</a>

</nav>

<router-outlet></router-outlet>

1. **What are active router links?**

RouterLinkActive is a directive that toggles css classes for active RouterLink bindings based on the current RouterState. i.e, the Router will add CSS classes when this link is active and and remove when the link is inactive. For example, you can add them to RouterLinks as below

<h1>Angular Router</h1>

<nav>

<a routerLink="/todosList" routerLinkActive="active">List of todos</a>

<a routerLink="/completed" routerLinkActive="active">Completed todos</a>

</nav>

<router-outlet></router-outlet>

1. **What is router state?**

RouterState is a tree of activated routes. Every node in this tree knows about the "consumed" URL segments, the extracted parameters, and the resolved data. You can access the current RouterState from anywhere in the application using the Router service and the routerState property.

@Component({templateUrl:'template.html'})

class MyComponent {

constructor(router: Router) {

const state: RouterState = router.routerState;

const root: ActivatedRoute = state.root;

const child = root.firstChild;

const id: Observable<string> = child.params.map(p => p.id);

//...

}

}

1. **What are router events?**

During each navigation, the Router emits navigation events through the Router.events property allowing you to track the lifecycle of the route. The sequence of router events is as below,

* 1. NavigationStart,
  2. RouteConfigLoadStart,
  3. RouteConfigLoadEnd,
  4. RoutesRecognized,
  5. GuardsCheckStart,
  6. ChildActivationStart,
  7. ActivationStart,
  8. GuardsCheckEnd,
  9. ResolveStart,
  10. ResolveEnd,
  11. ActivationEnd
  12. ChildActivationEnd
  13. NavigationEnd,
  14. NavigationCancel,
  15. NavigationError
  16. Scroll

1. **What is activated route?**

ActivatedRoute contains the information about a route associated with a component loaded in an outlet. It can also be used to traverse the router state tree. The ActivatedRoute will be injected as a router service to access the information. In the below example, you can access route path and parameters,

@Component({...})

class MyComponent {

constructor(route: ActivatedRoute) {

const id: Observable<string> = route.params.pipe(map(p => p.id));

const url: Observable<string> = route.url.pipe(map(segments => segments.join('')));

// route.data includes both `data` and `resolve`

const user = route.data.pipe(map(d => d.user));

}

}

1. **How do you define routes?**

A router must be configured with a list of route definitions. You configures the router with routes via the RouterModule.forRoot() method, and adds the result to the AppModule's imports array.

const appRoutes: Routes = [

{ path: 'todo/:id', component: TodoDetailComponent },

{

path: 'todos',

component: TodosListComponent,

data: { title: 'Todos List' }

},

{ path: '',

redirectTo: '/todos',

pathMatch: 'full'

},

{ path: '\*\*', component: PageNotFoundComponent }

];

@NgModule({

imports: [

RouterModule.forRoot(

appRoutes,

{ enableTracing: true } // <-- debugging purposes only

)

// other imports here

],

...

})

export class AppModule { }

1. **What is the purpose of Wildcard route?**

If the URL doesn't match any predefined routes then it causes the router to throw an error and crash the app. In this case, you can use wildcard route. A wildcard route has a path consisting of two asterisks to match every URL. For example, you can define PageNotFoundComponent for wildcard route as below

{ path: '\*\*', component: PageNotFoundComponent }

1. **Do I need a Routing Module always?**

No, the Routing Module is a design choice. You can skip routing Module (for example, AppRoutingModule) when the configuration is simple and merge the routing configuration directly into the companion module (for example, AppModule). But it is recommended when the configuration is complex and includes specialized guard and resolver services.

1. **What is Angular Universal?**

Angular Universal is a server-side rendering module for Angular applications in various scenarios. This is a community driven project and available under @angular/platform-server package. Recently Angular Universal is integrated with Angular CLI.

1. **What are different types of compilation in Angular?**

Angular offers two ways to compile your application,

* 1. Just-in-Time (JIT)
  2. Ahead-of-Time (AOT)

1. **What is JIT?**

Just-in-Time (JIT) is a type of compilation that compiles your app in the browser at runtime. JIT compilation is the default when you run the ng build (build only) or ng serve (build and serve locally) CLI commands. i.e, the below commands used for JIT compilation,

ng build

ng serve

1. **What is AOT?**

Ahead-of-Time (AOT) is a type of compilation that compiles your app at build time. For AOT compilation, include the --aot option with the ng build or ng serve command as below,

ng build --aot

ng serve --aot

**Note:** The ng build command with the --prod meta-flag (ng build --prod) compiles with AOT by default.

1. **Why do we need compilation process?**

The Angular components and templates cannot be understood by the browser directly. Due to that Angular applications require a compilation process before they can run in a browser. For example, In AOT compilation, both Angular HTML and TypeScript code converted into efficient JavaScript code during the build phase before browser runs it.

1. **What are the advantages with AOT?**

Below are the list of AOT benefits,

* 1. **Faster rendering:** The browser downloads a pre-compiled version of the application. So it can render the application immediately without compiling the app.
  2. **Fewer asynchronous requests:** It inlines external HTML templates and CSS style sheets within the application javascript which eliminates separate ajax requests.
  3. **Smaller Angular framework download size:** Doesn't require downloading the Angular compiler. Hence it dramatically reduces the application payload.
  4. **Detect template errors earlier:** Detects and reports template binding errors during the build step itself
  5. **Better security:** It compiles HTML templates and components into JavaScript. So there won't be any injection attacks.

1. **What are the ways to control AOT compilation?**

You can control your app compilation in two ways

* 1. By providing template compiler options in the tsconfig.json file
  2. By configuring Angular metadata with decorators

1. **What are the restrictions of metadata?**

In Angular, You must write metadata with the following general constraints,

* 1. Write expression syntax with in the supported range of javascript features
  2. The compiler can only reference symbols which are exported
  3. Only call the functions supported by the compiler
  4. Decorated and data-bound class members must be public.

1. **What are the two phases of AOT?**

The AOT compiler works in three phases,

* 1. **Code Analysis:** The compiler records a representation of the source
  2. **Code generation:** It handles the interpretation as well as places restrictions on what it interprets.
  3. **Validation:** In this phase, the Angular template compiler uses the TypeScript compiler to validate the binding expressions in templates.

1. **Can I use arrow functions in AOT?**

No, Arrow functions or lambda functions can’t be used to assign values to the decorator properties. For example, the following snippet is invalid:

@Component({

providers: [{

provide: MyService, useFactory: () => getService()

}]

})

To fix this, it has to be changed as following exported function:

function getService(){

return new MyService();

}

@Component({

providers: [{

provide: MyService, useFactory: getService

}]

})

If you still use arrow function, it generates an error node in place of the function. When the compiler later interprets this node, it reports an error to turn the arrow function into an exported function. **Note:** From Angular5 onwards, the compiler automatically performs this rewriting while emitting the .js file.

1. **What is the purpose of metadata json files?**

The metadata.json file can be treated as a diagram of the overall structure of a decorator's metadata, represented as an abstract syntax tree(AST). During the analysis phase, the AOT collector scan the metadata recorded in the Angular decorators and outputs metadata information in .metadata.json files, one per .d.ts file.

1. **Can I use any javascript feature for expression syntax in AOT?**

No, the AOT collector understands a subset of (or limited) JavaScript features. If an expression uses unsupported syntax, the collector writes an error node to the .metadata.json file. Later point of time, the compiler reports an error if it needs that piece of metadata to generate the application code.

1. **What is folding?**

The compiler can only resolve references to exported symbols in the metadata. Where as some of the non-exported members are folded while generating the code. i.e Folding is a process in which the collector evaluate an expression during collection and record the result in the .metadata.json instead of the original expression. For example, the compiler couldn't refer selector reference because it is not exported

let selector = 'app-root';

@Component({

selector: selector

})

Will be folded into inline selector

@Component({

selector: 'app-root'

})

Remember that the compiler can’t fold everything. For example, spread operator on arrays, objects created using new keywords and function calls.

1. **What are macros?**

The AOT compiler supports macros in the form of functions or static methods that return an expression in a single return expression. For example, let us take a below macro function,

export function wrapInArray<T>(value: T): T[] {

return [value];

}

You can use it inside metadata as an expression,

@NgModule({

declarations: wrapInArray(TypicalComponent)

})

export class TypicalModule {}

The compiler treats the macro expression as it written directly

@NgModule({

declarations: [TypicalComponent]

})

export class TypicalModule {}

1. **Give an example of few metadata errors?**

Below are some of the errors encountered in metadata,

* 1. **Expression form not supported:** Some of the language features outside of the compiler's restricted expression syntax used in angular metadata can produce this error. Let's see some of these examples,
  2. 1. export class User { ... }
  3. const prop = typeof User; // typeof is not valid in metadata

2. { provide: 'token', useValue: { [prop]: 'value' } }; // bracket notation is not valid in metadata

* 1. \*\* Reference to a local (non-exported) symbol:\*\* The compiler encountered a referenced to a locally defined symbol that either wasn't exported or wasn't initialized. Let's take example of this error,
  2. // ERROR
  3. let username: string; // neither exported nor initialized
  4. @Component({
  5. selector: 'my-component',
  6. template: ... ,
  7. providers: [
  8. { provide: User, useValue: username }
  9. ]
  10. })

export class MyComponent {}

You can fix this by either exporting or initializing the value,

export let username: string; // exported

(or)

let username = 'John'; // initialized

* 1. **Function calls are not supported:** The compiler does not currently support function expressions or lambda functions. For example, you cannot set a provider's useFactory to an anonymous function or arrow function as below.
  2. providers: [
  3. { provide: MyStrategy, useFactory: function() { ... } },
  4. { provide: OtherStrategy, useFactory: () => { ... } }

]

You can fix this with exported function

export function myStrategy() { ... }

export function otherStrategy() { ... }

... // metadata

providers: [

{ provide: MyStrategy, useFactory: myStrategy },

{ provide: OtherStrategy, useFactory: otherStrategy },

* 1. **Destructured variable or constant not supported:** The compiler does not support references to variables assigned by destructuring. For example, you cannot write something like this:
  2. import { user } from './user';
  3. // destructured assignment to name and age
  4. const {name, age} = user;
  5. ... //metadata
  6. providers: [
  7. {provide: Name, useValue: name},
  8. {provide: Age, useValue: age},

]

You can fix this by non-destructured values

import { user } from './user';

... //metadata

providers: [

{provide: Name, useValue: user.name},

{provide: Age, useValue: user.age},

]

1. **What is metadata rewriting?**

Metadata rewriting is the process in which the compiler converts the expression initializing the fields such as useClass, useValue, useFactory, and data into an exported variable, which replaces the expression. Remember that the compiler does this rewriting during the emit of the .js file but not in definition files( .d.ts file).

1. **How do you provide configuration inheritance?**

Angular Compiler supports configuration inheritance through extends in the tsconfig.json on angularCompilerOptions. i.e, The configuration from the base file(for example, tsconfig.base.json) are loaded first, then overridden by those in the inheriting config file.

{

"extends": "../tsconfig.base.json",

"compilerOptions": {

"experimentalDecorators": true,

...

},

"angularCompilerOptions": {

"fullTemplateTypeCheck": true,

"preserveWhitespaces": true,

...

}

}

1. **How do you specify angular template compiler options?**

The angular template compiler options are specified as members of the **angularCompilerOptions** object in the tsconfig.json file. These options will be specified adjecent to typescript compiler options.

{

"compilerOptions": {

"experimentalDecorators": true,

...

},

"angularCompilerOptions": {

"fullTemplateTypeCheck": true,

"preserveWhitespaces": true,

...

}

}

1. **How do you enable binding expression validation?**

You can enable binding expression validation explicitly by adding the compiler option **fullTemplateTypeCheck** in the "angularCompilerOptions" of the project's tsconfig.json. It produces error messages when a type error is detected in a template binding expression. For example, consider the following component:

@Component({

selector: 'my-component',

template: '{{user.contacts.email}}'

})

class MyComponent {

user?: User;

}

This will produce the following error:

my.component.ts.MyComponent.html(1,1): : Property 'contacts' does not exist on type 'User'. Did you mean 'contact'?

1. **What is the purpose of any type cast function?**

You can disable binding expression type checking using $any() type cast function(by surrounding the expression). In the following example, the error Property contacts does not exist is suppressed by casting user to the any type.

template: '{{$any(user).contacts.email}}'

The $any() cast function also works with this to allow access to undeclared members of the component.

template: '{{$any(this).contacts.email}}'

1. **What is Non null type assertion operator?**

You can use the non-null type assertion operator to suppress the Object is possibly 'undefined' error. In the following example, the user and contact properties are always set together, implying that contact is always non-null if user is non-null. The error is suppressed in the example by using contact!.email.

@Component({

selector: 'my-component',

template: '<span \*ngIf="user"> {{user.name}} contacted through {{contact!.email}} </span>'

})

class MyComponent {

user?: User;

contact?: Contact;

setData(user: User, contact: Contact) {

this.user = user;

this.contact = contact;

}

}

1. **What is type narrowing?**

The expression used in an ngIf directive is used to narrow type unions in the Angular template compiler similar to if expression in typescript. So \*ngIf allows the typeScript compiler to infer that the data used in the binding expression will never be undefined.

@Component({

selector: 'my-component',

template: '<span \*ngIf="user"> {{user.contact.email}} </span>'

})

class MyComponent {

user?: User;

}

1. **How do you describe various dependencies in angular application?**

The dependencies section of package.json with in an angular application can be divided as follow,

* 1. **Angular packages:** Angular core and optional modules; their package names begin @angular/.
  2. **Support packages:** Third-party libraries that must be present for Angular apps to run.
  3. **Polyfill packages:** Polyfills plug gaps in a browser's JavaScript implementation.

1. **What is zone?**

A Zone is an execution context that persists across async tasks. Angular relies on zone.js to run Angular's change detection processes when native JavaScript operations raise events

1. **What is the purpose of common module?**

The commonly-needed services, pipes, and directives provided by @angular/common module. Apart from these HttpClientModule is available under @angular/common/http.

1. **What is codelyzer?**

Codelyzer provides set of tslint rules for static code analysis of Angular TypeScript projects. ou can run the static code analyzer over web apps, NativeScript, Ionic etc. Angular CLI has support for this and it can be use as below,

ng new codelyzer

ng lint

1. **What is angular animation?**

Angular's animation system is built on CSS functionality in order to animate any property that the browser considers animatable. These properties includes positions, sizes, transforms, colors, borders etc. The Angular modules for animations are **@angular/animations** and **@angular/platform-browser** and these dependencies are automatically added to your project when you create a project using Angular CLI.

1. **What are the steps to use animation module?**

You need to follow below steps to implement animation in your angular project,

* 1. **Enabling the animations module:** Import BrowserAnimationsModule to add animation capabilities into your Angular root application module(for example, src/app/app.module.ts).

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

@NgModule({

imports: [

BrowserModule,

BrowserAnimationsModule

],

declarations: [ ],

bootstrap: [ ]

})

export class AppModule { }

* 1. **Importing animation functions into component files:** Import required animation functions from @angular/animations in component files(for example, src/app/app.component.ts).

import {

trigger,

state,

style,

animate,

transition,

// ...

} from '@angular/animations';

* 1. **Adding the animation metadata property:** add a metadata property called animations: within the @Component() decorator in component files(for example, src/app/app.component.ts)

@Component({

selector: 'app-root',

templateUrl: 'app.component.html',

styleUrls: ['app.component.css'],

animations: [

// animation triggers go here

]

})

1. **What is State function?**

Angular's state() function is used to define different states to call at the end of each transition. This function takes two arguments: a unique name like open or closed and a style() function. For example, you can write a open state function

state('open', style({

height: '300px',

opacity: 0.5,

backgroundColor: 'blue'

})),

1. **What is Style function?**

The style function is used to define a set of styles to associate with a given state name. You need to use it along with state() function to set CSS style attributes. For example, in the close state, the button has a height of 100 pixels, an opacity of 0.8, and a background color of green.

state('close', style({

height: '100px',

opacity: 0.8,

backgroundColor: 'green'

})),

**Note:** The style attributes must be in camelCase

1. **What is the purpose of animate function?**

Angular Animations are a powerful way to implement sophisticated and compelling animations for your Angular single page web application.

import { Component, OnInit, Input } from '@angular/core';

import { trigger, state, style, animate, transition } from '@angular/animations';

@Component({

selector: 'app-animate',

templateUrl: `<div [@changeState]="currentState" class="myblock mx-auto"></div>`,

styleUrls: `.myblock {

background-color: green;

width: 300px;

height: 250px;

border-radius: 5px;

margin: 5rem;

}`,

animations: [

trigger('changeState', [

state('state1', style({

backgroundColor: 'green',

transform: 'scale(1)'

})),

state('state2', style({

backgroundColor: 'red',

transform: 'scale(1.5)'

})),

transition('\*=>state1', animate('300ms')),

transition('\*=>state2', animate('2000ms'))

])

]

})

export class AnimateComponent implements OnInit {

@Input() currentState;

constructor() { }

ngOnInit() {

}

}

1. **What is transition function?**

The animation transition function is used to specify the changes that occur between one state and another over a period of time. It accepts two arguments: the first argument accepts an expression that defines the direction between two transition states, and the second argument accepts an animate() function. Let's take an example state transition from open to closed with an half second transition between states.

transition('open => closed', [

animate('500ms')

]),

1. **How to inject the dynamic script in angular?**

Using DomSanitizer we can inject the dynamic Html,Style,Script,Url.

import { Component, OnInit } from '@angular/core';

import { DomSanitizer } from '@angular/platform-browser';

@Component({

selector: 'my-app',

template: `

<div [innerHtml]="htmlSnippet"></div>

`,

})

export class App {

constructor(protected sanitizer: DomSanitizer) {}

htmlSnippet: string = this.sanitizer.bypassSecurityTrustScript("<script>safeCode()</script>");

}

1. **What is a service worker and its role in Angular?**

A service worker is a script that runs in the web browser and manages caching for an application. Starting from 5.0.0 version, Angular ships with a service worker implementation. Angular service worker is designed to optimize the end user experience of using an application over a slow or unreliable network connection, while also minimizing the risks of serving outdated content.

1. **What are the design goals of service workers?**

Below are the list of design goals of Angular's service workers,

* 1. It caches an application just like installing a native application
  2. A running application continues to run with the same version of all files without any incompatible files
  3. When you refresh the application, it loads the latest fully cached version
  4. When changes are published then it immediately updates in the background
  5. Service workers saves the bandwidth by downloading the resources only when they changed.

1. **What are the differences between AngularJS and Angular with respect to dependency injection?**

Dependency injection is a common component in both AngularJS and Angular, but there are some key differences between the two frameworks in how it actually works. | AngularJS | Angular | |---- | --------- | Dependency injection tokens are always strings | Tokens can have different types. They are often classes and sometimes can be strings. | | There is exactly one injector even though it is a multi-module applications | There is a tree hierarchy of injectors, with a root injector and an additional injector for each component. |

1. **What is Angular Ivy?**

Angular Ivy is a new rendering engine for Angular. You can choose to opt in a preview version of Ivy from Angular version 8.

* 1. You can enable ivy in a new project by using the --enable-ivy flag with the ng new command

ng new ivy-demo-app --enable-ivy

* 1. You can add it to an existing project by adding enableIvy option in the angularCompilerOptions in your project's tsconfig.app.json.

{

"compilerOptions": { ... },

"angularCompilerOptions": {

"enableIvy": true

}

}

1. **What are the features included in ivy preview?**

You can expect below features with Ivy preview,

* 1. Generated code that is easier to read and debug at runtime
  2. Faster re-build time
  3. Improved payload size
  4. Improved template type checking

1. **Can I use AOT compilation with Ivy?**

Yes, it is a recommended configuration. Also, AOT compilation with Ivy is faster. So you need set the default build options(with in angular.json) for your project to always use AOT compilation.

{

"projects": {

"my-project": {

"architect": {

"build": {

"options": {

...

"aot": true,

}

}

}

}

}

}

1. **What is Angular Language Service?**

The Angular Language Service is a way to get completions, errors, hints, and navigation inside your Angular templates whether they are external in an HTML file or embedded in annotations/decorators in a string. It has the ability to autodetect that you are opening an Angular file, reads your tsconfig.json file, finds all the templates you have in your application, and then provides all the language services.

1. **How do you install angular language service in the project?**

You can install Angular Language Service in your project with the following npm command

npm install --save-dev @angular/language-service

After that add the following to the "compilerOptions" section of your project's tsconfig.json

"plugins": [

{"name": "@angular/language-service"}

]

**Note:** The completion and diagnostic services works for .ts files only. You need to use custom plugins for supporting HTML files.

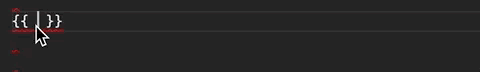
1. **Is there any editor support for Angular Language Service?**

Yes, Angular Language Service is currently available for Visual Studio Code and WebStorm IDEs. You need to install angular language service using an extension and devDependency respectively. In sublime editor, you need to install typescript which has has a language service plugin model.

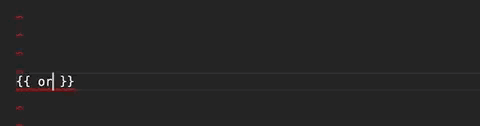
1. **Explain the features provided by Angular Language Service?**

Basically there are 3 main features provided by Angular Language Service,

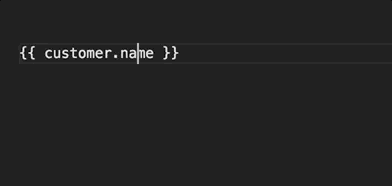
* 1. **Autocompletion:** Autocompletion can speed up your development time by providing you with contextual possibilities and hints as you type with in an interpolation and elements.



* 1. **Error checking:** It can also warn you of mistakes in your code.



* 1. **Navigation:** Navigation allows you to hover a component, directive, module and then click and press F12 to go directly to its definition.



1. **How do you add web workers in your application?**

You can add web worker anywhere in your application. For example, If the file that contains your expensive computation is src/app/app.component.ts, you can add a Web Worker using ng generate web-worker app command which will create src/app/app.worker.ts web worker file. This command will perform below actions,

* 1. Configure your project to use Web Workers
  2. Adds app.worker.ts to receive messages

addEventListener('message', ({ data }) => {

const response = `worker response to ${data}`;

postMessage(response);

});

* 1. The component app.component.ts file updated with web worker file

if (typeof Worker !== 'undefined') {

// Create a new

const worker = new Worker('./app.worker', { type: 'module' });

worker.onmessage = ({ data }) => {

console.log('page got message: $\{data\}');

};

worker.postMessage('hello');

} else {

// Web Workers are not supported in this environment.

}

**Note:** You may need to refactor your initial scaffolding web worker code for sending messages to and from.

1. **What are the limitations with web workers?**

You need to remember two important things when using Web Workers in Angular projects,

* 1. Some environments or platforms(like @angular/platform-server) used in Server-side Rendering, don't support Web Workers. In this case you need to provide a fallback mechanism to perform the computations to work in this environments.
  2. Running Angular in web worker using @angular/platform-webworker is not yet supported in Angular CLI.

1. **What is Angular CLI Builder?**

In Angular8, the CLI Builder API is stable and available to developers who want to customize the Angular CLI by adding or modifying commands. For example, you could supply a builder to perform an entirely new task, or to change which third-party tool is used by an existing command.

1. **What is a builder?**

A builder function ia a function that uses the Architect API to perform a complex process such as "build" or "test". The builder code is defined in an npm package. For example, BrowserBuilder runs a webpack build for a browser target and KarmaBuilder starts the Karma server and runs a webpack build for unit tests.

1. **How do you invoke a builder?**

The Angular CLI command ng run is used to invoke a builder with a specific target configuration. The workspace configuration file, angular.json, contains default configurations for built-in builders.

1. **How do you create app shell in Angular?**

An App shell is a way to render a portion of your application via a route at build time. This is useful to first paint of your application that appears quickly because the browser can render static HTML and CSS without the need to initialize JavaScript. You can achieve this using Angular CLI which generates an app shell for running server-side of your app.

ng generate appShell [options] (or)

ng g appShell [options]

1. **What are the case types in Angular?**

Angular uses capitalization conventions to distinguish the names of various types. Angular follows the list of the below case types.

* 1. **camelCase :** Symbols, properties, methods, pipe names, non-component directive selectors, constants uses lowercase on the first letter of the item. For example, "selectedUser"
  2. **UpperCamelCase (or PascalCase):** Class names, including classes that define components, interfaces, NgModules, directives, and pipes uses uppercase on the first letter of the item.
  3. **dash-case (or "kebab-case"):** The descriptive part of file names, component selectors uses dashes between the words. For example, "app-user-list".
  4. **UPPER\_UNDERSCORE\_CASE:** All constants uses capital letters connected with underscores. For example, "NUMBER\_OF\_USERS".

1. **What are the class decorators in Angular?**

A class decorator is a decorator that appears immediately before a class definition, which declares the class to be of the given type, and provides metadata suitable to the type The following list of decorators comes under class decorators,

* 1. @Component()
  2. @Directive()
  3. @Pipe()
  4. @Injectable()
  5. @NgModule()

1. **What are class field decorators?**

The class field decorators are the statements declared immediately before a field in a class definition that defines the type of that field. Some of the examples are: @input and @output,

@Input() myProperty;

@Output() myEvent = new EventEmitter();

1. **What is declarable in Angular?**

Declarable is a class type that you can add to the declarations list of an NgModule. The class types such as components, directives, and pipes comes can be declared in the module.

1. **What are the restrictions on declarable classes?**

Below classes shouldn't be declared,

* 1. A class that's already declared in another NgModule
  2. Ngmodule classes
  3. Service classes
  4. Helper classes

1. **What is a DI token?**

A DI token is a lookup token associated with a dependency provider in dependency injection system. The injector maintains an internal token-provider map that it references when asked for a dependency and the DI token is the key to the map. Let's take example of DI Token usage,

const BASE\_URL = new InjectionToken<string>('BaseUrl');

const injector =

Injector.create({providers: [{provide: BASE\_URL, useValue: 'http://some-domain.com'}]});

const url = injector.get(BASE\_URL);

1. **What is Angular DSL?**

A domain-specific language (DSL) is a computer language specialized to a particular application domain. Angular has its own Domain Specific Language (DSL) which allows us to write Angular specific html-like syntax on top of normal html. It has its own compiler that compiles this syntax to html that the browser can understand. This DSL is defined in NgModules such as animations, forms, and routing and navigation. Basically you will see 3 main syntax in Angular DSL.

* 1. (): Used for Output and DOM events.
  2. []: Used for Input and specific DOM element attributes.
  3. \*: Structural directives(\*ngFor or \*ngIf) will affect/change the DOM structure.

1. **what is an rxjs subject in Angular**

An RxJS Subject is a special type of Observable that allows values to be multicasted to many Observers. While plain Observables are unicast (each subscribed Observer owns an independent execution of the Observable), Subjects are multicast.

A Subject is like an Observable, but can multicast to many Observers. Subjects are like EventEmitters: they maintain a registry of many listeners.

import { Subject } from 'rxjs';

const subject = new Subject<number>();

subject.subscribe({

next: (v) => console.log(`observerA: ${v}`)

});

subject.subscribe({

next: (v) => console.log(`observerB: ${v}`)

});

subject.next(1);

subject.next(2);

1. **What is Bazel tool?**

Bazel is a powerful build tool developed and massively used by Google and it can keep track of the dependencies between different packages and build targets. In Angular8, you can build your CLI application with Bazel. **Note:** The Angular framework itself is built with Bazel.

1. **What are the advantages of Bazel tool?**

Below are the list of key advantages of Bazel tool,

* 1. It creates the possibility of building your back-ends and front-ends with the same tool
  2. The incremental build and tests
  3. It creates the possibility to have remote builds and cache on a build farm.

1. **How do you use Bazel with Angular CLI?**

The @angular/bazel package provides a builder that allows Angular CLI to use Bazel as the build tool.

* 1. **Use in an existing applciation:** Add @angular/bazel using CLI

ng add @angular/bazel

* 1. **Use in a new application:** Install the package and create the application with collection option

npm install -g @angular/bazel

ng new --collection=@angular/bazel

When you use ng build and ng serve commands, Bazel is used behind the scenes and outputs the results in dist/bin folder.

1. **How do you run Bazel directly?**

Sometimes you may want to bypass the Angular CLI builder and run Bazel directly using Bazel CLI. You can install it globally using @bazel/bazel npm package. i.e, Bazel CLI is available under @bazel/bazel package. After you can apply the below common commands,

bazel build [targets] // Compile the default output artifacts of the given targets.

bazel test [targets] // Run the tests with \*\_test targets found in the pattern.

bazel run [target]: Compile the program represented by target and then run it.

1. **What is platform in Angular?**

A platform is the context in which an Angular application runs. The most common platform for Angular applications is a web browser, but it can also be an operating system for a mobile device, or a web server. The runtime-platform is provided by the @angular/platform-\* packages and these packages allow applications that make use of @angular/coreand @angular/common to execute in different environments. i.e, Angular can be used as platform-independent framework in different environments, For example,

* 1. While running in the browser, it uses platform-browser package.
  2. When SSR(server-side rendering ) is used, it uses platform-server package for providing web server implementation.

1. **What happens if I import the same module twice?**

If multiple modules imports the same module then angular evaluates it only once (When it encounters the module first time). It follows this condition even the module appears at any level in a hierarchy of imported NgModules.

1. **How do you select an element with in a component template?**

You can use @ViewChild directive to access elements in the view directly. Let's take input element with a reference,

<input #uname>

and define view child directive and access it in ngAfterViewInit lifecycle hook

@ViewChild('uname') input;

ngAfterViewInit() {

console.log(this.input.nativeElement.value);

}

**Q1. What is Angular 4 and how it differs from Angular 1.x?**

Angular 4 is a Javascript framework built around the concept of components, and more precisely, with the Web Components standard in mind. It was rewritten from scratch by the Angular team using Typescript (although we can use it with *ES5*, *ES6*, or *Dart* as well).

Angular 4 is a big change for us compared to 1.x. Because it is a completely different framework than 1.x, and is not backward-compatible. Angular 4 is written entirely in Typescript and meets the ECMAScript 6 specification. The main differences are:

* Angular 4 is entirely component based. Controllers and $scope are no longer used. They have been replaced by components and directives.
* Angular 4 uses TypeScript. TypeScript will not be used in the browser directly. So the program code is compiled to JavaScript. This can be achieved with “Traceur”.
* The digest cycle from Angular 1.x has been replaced by another internal mechanism known as “**Change Detection**”. This feature, along with other improvements and tweaks, yields a considerable increase in performance.
* Unlike Angular 1.x where we can get most of the functionalities in angular.js file, Angular 4 follows module pattern. We need to import the functions ourself and export them when we need anywhere else.
* There are no more factory, service, provider in Angular 4. We need to use class for declaring a service.

**Q2. What is component decorators in Angular 4?**

The main objectives of decorators is to add some metadata to the class that will tell Angular 4 how to process a class. Or in another words, Decorators are functions that modify JavaScript classes. Angular has many decorators that attach metadata to classes so that it knows what those classes mean and how they should work.

If we consider Component in Angular 4, we will have following options to configure.

* **selector:** — define the name of the HTML element in which our component will live.
* **template** or **templateUrl:** — It can be inline string or link an external html file. It allows us to tie logic from our component directly to a view.
* **styles:** — the styles array for our specific component. We can also link external CSS by **styleUrls**.
* **directives:** — another component directives we want to use inside our components.
* **providers:** — This is the place we are passing the services that we need insider our components.

Immediately after this decorator or right to it, we need to export a class where our variables and functions reside that our component uses.

https://miro.medium.com/max/60/1*vPd3-cKqWDx0GwxT2sU-9g.jpeg?q=20



**Q3. What is compilation in Angular 4? And what are the types of compilation in Angular 4?**

An Angular application consists largely of components and their HTML templates. Before the browser can render the application, the components and templates must be converted to executable JavaScript by the *Angular compiler*.

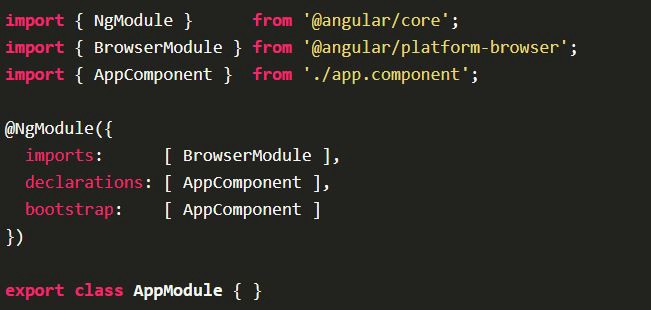
There is actually only one Angular compiler. The difference between AOT and JIT is a matter of timing and tooling. There are two types of compilation Angular 4 provides.

* **Just-in-time (JIT) compilation:** This is a standard development approach which compiles our Typescript and html files in the browser at runtime, as the application loads. It is great but has disadvantages. Views take longer to render because of the in-browser compilation step. App size increases as it contains angular compiler and other library code that won’t actually need.
* **Ahead-of-time (AOT) compilation:** With AOT, the compiler runs at the build time and the browser downloads only the pre compiled version of the application. The browser loads executable code so it can render the application immediately, without waiting to compile the app first. This compilation is better than JIT because of Fast rendering, smaller application size, security and detect template errors earlier.

**Q4. What is @NgModule?**

An NgModule class describes how the application parts fit together. Every application has at least one NgModule, the root module that we bootstrap to launch the application.

https://miro.medium.com/max/60/1*9pRJxfO4TIHUe-hUDHitfg.jpeg?q=20



Here the AppComponent is the root module of our application that Angular creates and inserts it into the index.html page.

**Q5. What are all the *metadata* properties of NgModule? And what are they used for?**

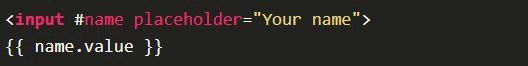
@NgModule accepts a metadata object that tells Angular how to compile and launch the application. The properties are:

* **imports** – Modules that the application needs or depends on to run like, the BrowserModule that every application needs to run in a browser.
* **declarations** – the application's components, which belongs to the NgModuleclass. We must declare every component in an NgModule class. If we use a component without declaring it, we'll see a clear error message in the browser console.
* **bootstrap** – the root component that Angular creates and inserts into the index.html host web page. The application will be launched by creating the components listed in this array.

**Q6. What is Template reference variables?**

A template reference variable (#var) is a reference to a DOM element within a template. We use hash symbol (#) to declare a reference variable in a template.

https://miro.medium.com/max/60/1*gQZoMBShop0j4NWXXKnGow.jpeg?q=20



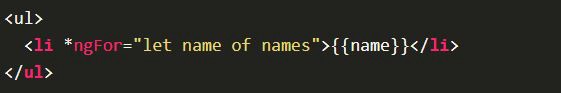
In the above code the #name declares a variable on the input element. Here the name refers to the *input* element. Now we can access any property of the inputDOM, using this reference variable. For example, we can get the value of the inputelement as name.value and the value of the placeholder property by name.placeholder anywhere in the template.

Finally, a Template reference variable refers to its attached element, component or directive. It can be accessed anywhere in the entire template. We can also use ref- instead of #. Thus we can also write the above code as ref-name.

**Q7. What are structural directives?**

Structural directives are responsible for HTML layout. They shape or reshape the DOM’s structure, typically by adding, removing, or manipulating elements. Structural directives are easy to recognize. An asterisk (\*) precedes the directive attribute name as in this example.

https://miro.medium.com/max/60/1*_fkCkQaNF1xvqGXMCTlKMw.jpeg?q=20



The ngFor directive iterates over the component's names array and renders an instance of this template for each name in that array.

Some of the other structural directives in Angular are ngIf and ngSwitch.

**Q8. What is Directive in Angular 4? How it differs from Components?**

Directives allow us to attach behavior to elements in the DOM, for example, doing something on mouse over or click. In Angular, a Directive decoraor (@Directive) is used to mark a class as an Angular directive and provides additional metadata that determines how the directive should be processed. Below are the metadata properties of a directive.

* selector - css selector that identifies this component in a template
* host - map of class property to host element bindings for events, properties and attributes
* inputs - list of class property names to data-bind as component inputs
* outputs - list of class property names that expose output events that others can subscribe to
* providers - list of providers available to this component and its children
* queries - configure queries that can be injected into the component
* exportAs - name under which the component instance is exported in a template

A Component is a directive with a template. So we should use a Component whenever we want reusable set of DOM elements with behaviors of UI. And we should use a Directive whenever we want reusable behavior to supplement the DOM.

**Q9. What are all the types of Directives?**

There are three types of directives in Angular. They are **attribute directives**, **structural directives**, and **components**.

* **Structural directives** change the DOM layout by adding and removing DOM elements. For example, \*ngIf and \*ngFor
* **Attribute directives** change the appearance or behavior of an element. . For example, \*ngStyle and \*ngClass
* **Components** are basically directives with a template.

**Q10. What are all the uses of a service?**

Services encapsulates business logic and separates them from UI concerns or the controller concerns, which governs them both.

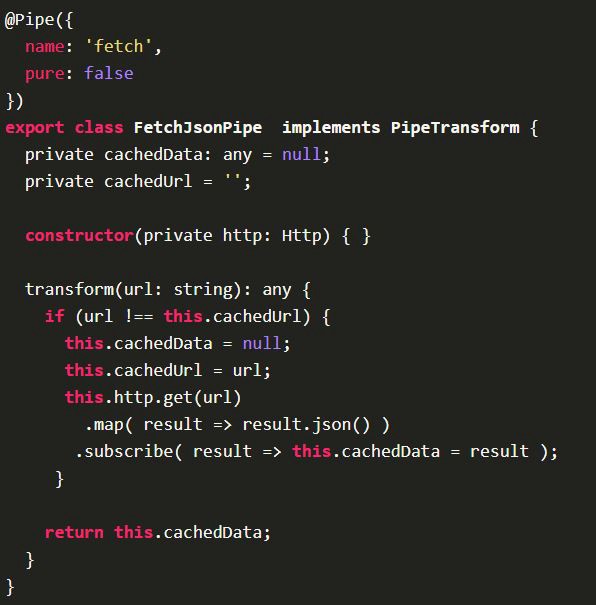
Services focus on functionality thus benefits in maintainability. The separation of UI logic from business logic is intended to reduce the coupling between the UI layer and the Model layer, leading to a cleaner design that is easier to develop, test, and maintain.

**Q11. What is Pure and Impure Pipes?**

Pure pipes are stateless that flow input date without remembering anything or causing detectable side-effects. Pipes are pure by default, hence most pipes are pure. We can make a pipe impure by setting its pure flag to false. Angular executes a pure pipe only when it detects a *pure* change to the input value. A pure change is either a change to a primitive input value or a changed object reference.

Impure pipes are those which can manage the state of the data they transform. A pipe that creates an HTTP request, stores the response and displays the output, is a impure or stateful pipe. Stateful Pipes should be used cautiously. Angular provides AsyncPipe, which is stateful. In the following code, the pipe only calls the server when the request URL changes and it caches the server response. The code uses the Angular http client to retrieve data:

https://miro.medium.com/max/60/1*-5j-XE0WK3mt17j55WyZMQ.jpeg?q=20



**Q12. What is Redux and @ngRx?**

Redux is an application state manager for JavaScript applications, and keeps with the core principles of the Flux-architecture by having a unidirectional data flow in our application. Redux applications have only one global, read-only application state. This state is calculated by “reducing” over a collection or stream of actions that update it in controlled ways.

@ngrx is a set of modules that implement the same way of managing state as well as some of the middleware and tools in the Redux ecosystem. In other way, ngrx is a collection of reactive libraries for angular, containing a redux implementation and many other useful libraries.

Using this technique, we keep our application state in Store and everything saved in the store is read only. The only way to change the state is to emit an action, an object describing what happened.

**Q13. How to prevent security threads in Angular App? What are all the ways we could secure our App?**

Some of them are:

* Avoid using/injecting dynamic HTML content to your component.
* If using external HTML which is coming from database or somewhere outside the application, sanitize it before using.
* Try not to put external urls in the application unless it is trusted. Avoid url re-direction unless it is trusted.
* Consider using AOT compilation or offline compilation.
* Try to prevent XSRF attack by restricting the api and use of the app for known or secure environment/browsers.

**Q14. How to optimize Angular app?**

* Consider lazy loading instead of fully bundled app if the app size is more.
* Make sure that any 3rd party library, which is not used, is removed from the application.
* Have all dependencies and dev-dependencies are clearly separated.
* Make sure the application doesn’t have un-necessary import statements.
* Make sure the application is bundled, uglified, and tree shaking is done.
* Consider AOT compilation.

**Q15. What is NgZone service? How Angular is notified about the changes?**

Zone.js is one of the Angular dependencies which provides a mechanism, called zones, for encapsulating and intercepting asynchronous activities in the browser (e.g. setTimeout, setInterval, promises). These zones are *execution contexts* that allow Angular to track the start and completion of asynchronous activities and perform tasks as required (e.g. change detection). Zone.js provides a global zone that can be forked and extended to further encapsulate/isolate asynchronous behaviour, which Angular does so in its NgZone service, by creating a fork and extending it with its own behaviours.

The NgZone service provides us with a number of Observables and methods for determining the state of Angular's zone and to execute code in different ways inside and outside Angular's zone.

NgZone exposes a set of Observables that allow us to determine the current status, or stability, of Angular's zone.

* onUnstable – Notifies when code has entered and is executing within the Angular zone.
* onMicrotaskEmpty - Notifies when no more microtasks are queued for execution. Angular subscribes to this internally to signal that it should run change detection.
* onStable – Notifies when the last onMicroTaskEmpty has run, implying that all tasks have completed and change detection has occurred.
* onError – Notifies when an error has occurred. Angular subscribes to this internally to send uncaught errors to its own error handler, i.e. the errors you see in your console prefixed with 'EXCEPTION:'.

We can inject the NgZone service in our component/services/etc. and can subscribe to these observables.

https://miro.medium.com/max/60/1*Qx8bfkhjvLRrsQ8cgD4IQw.jpeg?q=20



Subscribing to these can help you determine if your code is unexpectedly triggering change detection as a result of operations that do not affect application state.

**Q16. What is Traceur compiler?**

Traceur compiler is a Google project. It compiles ECMAScript Edition 6 (ES6) (including classes, generators and so on) code on the fly to regular Javascript (ECMAScript Edition 5 [ES5]) to make it compatible for the browser.

Traceur itself is written in ES6, compiled to ES5.

### 1. What’s new in Angular 5?

Certain tools are optimized in the new version of [Angular](https://www.greycampus.com/angularjs-training-instructor-led), let us see what the tools are:

* Angular 5 supports Typescript version 2.4
* Angular 5 supports RxJS 5.5 which has new features like Pipeable Operators
* A build tool to make the js bundles (files) lighter
* Ahead of Time (AOT) is updated to be on by default
* Events like ActivationStart and ActivationEnd are introduced in Router

### 2. Name the building blocks of Angular.

The Angular application is made using the following:

* Modules
* Component
* Template
* Directives
* Data Binding
* Services
* Dependency Injection
* Routing

### 3. What is Transpiling in Angular?

Transpiling is the process of converting the typescript into javascript (using Traceur, a JS compiler). Though typescript is used to write code in the Angular applications, the code is internally transpiled into javascript.

### 4. Which of the Angular life cycle component execution happens when a data-bound input value updates?

ngOnChanges is the life cycle hook that gets executed whenever a change happens to the data that was bound to an input.

### 5. Differentiate between Components and Directives in Angular 5.

Components break up the application into smaller parts; whereas, Directives add behavior to an existing DOM element.

### 6. What is the use of @Input and @Output?

When it comes to the communication of Angular Components, which are in Parent-Child Relationship; we use @Input in Child Component when we are passing data from Parent to Child Component and @Output is used in Child Component to receive an event from Child to Parent Component.

**Learn more about**[Top 5 Skills That Make You A Sure Shot Programmer](https://www.greycampus.com/blog/programming/top-five-skills-that-make-you-a-sure-shot-programmer)

### 7. What is ng-content Directive?

The HTML elements like p (paragraph) or h1 (heading) have some content between the tags. For example, <p>this is a paragraph</p> and <h1>this is a heading</h1>. Now, similar to this, what if we want to have some custom text or content between the angular tags like  <app-tax>some tax-related content</app-tax> This will not work the way it worked for HTML elements.  Now, in such cases, the <ng-content> tag directive is used.

### 8. What does a router.navigate do?

When we want to route to a component we use router.navigate.  Syntax: this.router.navigate([‘/component\_name’]);

### 9. What is ViewEncapsulation?

ViewEncapsulation decides whether the styles defined in a component can affect the entire application or not. There are three ways to do this in Angular:

Emulated: styles from other HTML spread to the component.

Native: styles from other HTML do not spread to the component.

None: styles defined in a component are visible to all components.

### 10. What are Services in Angular and what command is used to create a service?

Services help us in not repeating the code. With the creation of services, we can use the same code from different components. Here is the command to create a service in angular, ng g service User (a UserService is created when this command is used).

### 11. What is Dependency Injection in Angular 4?

When a component is dependent on another component the dependency is injected/provided during runtime.

### 12. What is Routing in Angular 5?

Routing helps a user in navigating to different pages using links.

### 13. How to handle Events in Angular 5?

Any activity (button click, mouse click, mouse hover, mouse move, etc) of a user on a frontend/web screen is termed as an event. Such events are passed from the view (.HTML) page to a typescript component (.ts).

### 14. What is RouterOutlet?

RouterOutlet is a substitution for templates rendering the components. In other words, it represents or renders the components on a template at a particular location.

### 15. Explain the usage of {{}}?

The set of brackets {{}} when used with an HTML tag, represent data from a component. For example, on a HTML page which has <h1>{{variableName}}</h1>, here the ‘variableName’ is actually typescript (component) data representing its value on the template; i.e., HTML. This entire concept is called String Interpolation.

**You may also like:**[Other Programming workshops](https://www.greycampus.com/programming)

### 16. In how many ways the Data Binding can be done?

Data Binding happens between the HTML (template) and typescript (component). Data binding can be done in 3 ways:

(i) Property Binding (ii) Event Binding (iii) Two-Way Data Binding.

### 17. What is the sequence of Angular Lifecycle Hooks?

OnChange()  -  OnInit()  -  DoCheck()  -  AfterContentInit()  -  AfterContentChecked()  -  AfterViewInit()  -  AfterViewChecked()  -  OnDestroy().

### 18. What is the purpose of using package.json in the angular project?

With the existence of package.json, it will be easy to manage the dependencies of the project. If we are using typescript in the angular project then we can mention the typescript package and version of typescript in package.json.

### 19. How is SPA (Single Page Application) technology different from the traditional web technology?

In traditional web technology, the client requests for a web page (HTML/JSP/asp) and the server sends the resource (or HTML page), and the client again requests for another page and the server responds with another resource. The problem here is a lot of time is consumed in the requesting/responding or due to a lot of reloading. Whereas, in the SPA technology, we maintain only one page (index.HTML) even though the URL keeps on changing.

### 20. What is Component in Angular Terminology?

A web page in Angular has many components involved in it. A Component is basically a block in which the data can be displayed on HTML using some logic usually written in typescript.

### 21. What are ngModel and how do we represent it?

ngModel is a directive which can be applied on a text field. This a two-way data binding. ngModel is represented by [()]

### 22. What does a Subscribe method do in Angular 4?

It is a method which is subscribed to an observable. Whenever the subscribe method is called, an independent execution of the observable happens.

### 23. Differentiate between Observables and Promises.

Observables are lazy, which means nothing happens until a subscription is made. Whereas Promises are eager; which means as soon as a promise is created, the execution takes place. Observable is a stream in which passing of zero or more events is possible and the callback is called for each event. Whereas, promise handles a single event.

### 24. What is an AsyncPipe in Angular?

When an observable or promise returns something, we use a temporary property to hold the content. Later, we bind the same content to the template. With the usage of AsyncPipe, the promise or observable can be directly used in a template and a temporary property is not required.

### 25. Explain Authentication and Authorization.

Authentication: The user login credentials are passed to an authenticate API (on the server). On the server side validation of the credentials happens and a JSON Web Token (JWT) is returned. JWT is a JSON object that has some information or attributes about the current user.  Once the JWT is given to the client, the client or the user will be identified with that JWT.

Authorization: After logging in successfully, the authenticated or genuine user does not have access to everything. The user is not authorized to access someone else’s data,  he/she is authorized to access some data.

### 26. What is AOT Compilation?

Every angular application gets compiled internally. The angular compiler takes javascript code, compiles it and produces javascript code again. Ahead-of-Time Compilation does not happen every time or for every user, as is the case with Just-In-Time (JIT) Compilation.

### 27. What is Redux?

It is a library which helps us maintain the state of the application. Redux is not required in applications that are simple with the simple data flow, it is used in Single Page Applications that have complex data flow.

### 28. What are the Pipes?

This feature is used to change the output on the template; something like changing the string into uppercase and displaying it on the template. It can also change Date format accordingly.

### 29. Differentiate between ng-Class and ng-Style.

In ng-Class, loading of CSS class is possible; whereas, in ng-Style we can set the CSS style.

### 30. Why Typescript with Angular?

Typescript is a superset of Javascript. Earlier, Javascript was the only client-side language supported by all browsers. But, the problem with Javascript is, it is not a pure Object Oriented Programming Language. The code written in JS without following patterns like Prototype Pattern becomes messy and finally leading to difficulties in maintainability and reusability. Instead of learning concepts (like patterns) to maintain code, programmers prefer to maintain the code in an OOP approach and is made available with a programming language like Typescript was thus developed by Microsoft in a way that it can work as Javascript and also offer what javascript cannot ie;

* pure OOPS as Typescript offers concepts like Generics, Interfaces and Types (a Static Typed Language) which makes it is easier to catch incorrect data types passing to variables.
* TS provides flexibility to programmers experienced in java, .net as it offers encapsulation through classes and interfaces.
* JS version ES5 offers features like Constructor Function, Dynamic Types, Prototypes. The next version of Javascript ie ES6 introduced a new feature like Class keyword but not supported by many browsers.
* TS offers Arrow Functions (=>) which is an ES6 feature not supported by many browsers directly but when used in TS, gets compiled into JS ES5 and runs in any browser.
* TS is not the only alternative to JS, we have CoffeeScript, Dart(Google).
* Finally, it is like, TS makes life easier when compared to JS.

**1) What is AngularJS?**

AngularJS is a javascript framework used for creating single web page applications.  It allows you to use HTML as your template language and enables you to extend HTML’s syntax to express your application’s components clearly

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

The key features of AngularJS are

* Scope
* Controller
* Model
* View
* Services
* Data Binding
* Directives
* Filters
* Testable

**3) Explain what is scope in AngularJS ?**

Scope refers to the application model, it acts like glue between application controller and the view.  Scopes are arranged in hierarchical structure and impersonate the DOM ( Document Object Model) structure of the application.  It can watch expressions and propagate events.

**4) Explain what is services in AngularJS ?**

In AngularJS services are the singleton objects or functions that are used for carrying out specific tasks.  It holds some business logic and these function can be called as controllers, directive, filters and so on.

**5) Explain what is Angular Expression? Explain what is key difference between angular expressions and JavaScript expressions?**

Like JavaScript,  Angular expressions are code snippets that are usually placed in binding such as {{ expression }}

The key difference between the JavaScript expressions and Angular expressions

* **Context :** In Angular, the expressions are evaluated against a scope object, while the Javascript expressions are evaluated against the global window
* **Forgiving:** In Angular expression evaluation is forgiving to null and undefined, while in Javascript undefined properties generates TypeError or ReferenceError
* **No Control Flow Statements:** Loops, conditionals or exceptions cannot be used in an angular expression
* **Filters:** To format data before displaying it you can use filters

**6) With options on page load how you can initialize a select box ?**

You can initialize a select box with options on page load by using **ng-init** directive

* <div ng-controller = " apps/dashboard/account " ng-switch
* On = "! ! accounts" ng-init = " loadData ( ) ">

**7) Explain what are directives ? Mention some of the most commonly used directives in AngularJS application ?**

A directive is something that introduces new syntax, they are like markers on DOM element which attaches a special behavior to it. In any AngularJS application, directives are the most important components.

Some of the commonly used directives are **ng-model, ng-App, ng-bind, ng-repeat , ng-show** etc.

**8) Mention what are the advantages of using AngularJS ?**

AngularJS has several advantages in web development.

* AngularJS supports MVC pattern
* Can do two ways data binding using AngularJS
* It has per-defined form validations
* It supports both client server communication
* It supports animations

**9) Explain what Angular JS routes does ?**

Angular js routes enable you to create different URLs for different content in your application.  Different URLs for different content enables user to bookmark URLs to specific content.  Each such bookmarkable URL in AngularJS is called a route

A value in Angular JS is a simple object.  It can be a number, string or JavaScript object.  Values are typically used as configuration injected into factories, services or controllers. A value should be belong to an AngularJS module.

Injecting a value into an AngularJS controller function is done by adding a parameter with the same name as the value

**10)  Explain what is data binding in AngularJS ?**

Automatic synchronization of data between the model and view components is referred as data binding in AngularJS.  There are two ways for data binding

1. **Data mining in classical template systems**
2. **Data binding in angular templates**

**11)  What makes AngularJS better ?**

* **Registering Callbacks:** There is no need to register callbacks . This makes your code simple and easy to debug.
* **Control HTML DOM programmatically:**All the application that are created using Angular never have to manipulate the DOM although it can be done if it is required
* **Transfer data to and from the UI:**AngularJS helps to eliminate almost all of the boiler plate like validating the form, displaying validation errors, returning to an internal model and so on which occurs due to flow of marshalling data
* **No initilization code:**With AngularJS you can bootstrap your app easily using services, which auto-injected into your application in Guice like dependency injection style

**12)  Explain what is string interpolation in Angular.js ?**

In Angular.js the compiler during the compilation process matches text and attributes using interpolate service to see if they contains embedded expressions.  As part of normal digest cycle these expressions are updated and registered as watches.

**13)  Mention the steps for the compilation process of HTML happens?**

Compilation of HTML process occurs in following ways

* Using the standard browser API, first the HTML is parsed into DOM
* By using the call to the $compile () method, compilation of the DOM is performed.  The method traverses the DOM and matches the directives.
* Link the template with scope by calling the linking function returned from the previous step

**14)  Explain what is directive and Mention what are the different types of Directive?**

During compilation process when specific HTML constructs are encountered a behaviour or function is triggered, this function is referred as directive.  It is executed when the compiler encounters it in the DOM.

Different types of directives are

* Element directives
* Attribute directives
* CSS class directives
* Comment directives

**15)  Explain what is linking function and type of linking function?**

Link combines the directives with a scope and produce a live view.  For registering DOM listeners as well as updating the DOM, link function is responsible. After the template is cloned it is executed.

* Pre-linking function: Pre-linking function is executed before the child elements are linked.  It is not considered as the safe way for DOM transformation.
* Post linking function: Post linking function is executed after the child elements are linked. It is safe to do DOM transformation by post-linking function

**16)  Explain what is injector?**

An injector is a service locator.  It is used to retrieve object instances as defined by provider, instantiate types, invoke methods and load modules.  There is a single injector per Angular application, it helps to look up an object instance by its name.

**17)  Explain what is the difference between link and compile in Angular.js?**

* Compile function: It is used for template DOM Manipulation and collect all of the directives.
* Link function: It is used for registering DOM listeners as well as instance DOM manipulation. It is executed once the template has been cloned.

**18)  Explain what is factory method in AngularJS?**

For creating the directive, factory method is used.  It is invoked only once, when compiler matches the directive for the first time.  By using $injector.invoke the factory method is invoked.

**19)  Mention what are the styling form that ngModel adds to CSS classes ?**

ngModel adds these CSS classes to allow styling of form as well as control

* ng- valid
* ng- invalid
* ng-pristine
* ng-dirty

**20)  Mention what are the characteristics of “Scope”?**

* To observer model mutations scopes provide APIs ($watch)
* To propagate any model changes through the system into the view from outside of the Angular realm
* A scope inherits properties from its parent scope,  while providing access to shared model properties, scopes can be nested to isolate application components
* Scope provides context against which expressions are evaluated

**21)  Explain what is DI (Dependency Injection ) and how an object or function can get a hold of its dependencies ?**

DI or Dependency Injection is a software design pattern that deals with how code gets hold of its dependencies.  In order to retrieve elements of the application which is required to be configured when module gets loaded , the operation “config” uses dependency injection.

These are the ways that object uses to hold of its dependencies

* Typically using the new operator, dependency can be created
* By referring to a global variable, dependency can be looked up
* Dependency can be passed into where it is required

**22)  Mention what are the advantages of using Angular.js framework ?**

Advantages of using Angular.js as framework are

* Supports two way data-binding
* Supports MVC pattern
* Support static template and angular template
* Can add custom directive
* Supports REST full services
* Supports form validations
* Support both client and server communication
* Support dependency injection
* Applying Animations
* Event Handlers

**23)  Explain the concept of scope hierarchy?  How many scope can an application have?**

Each angular application consist of one root scope but may have several child scopes. As child controllers and some directives create new child scopes, application can have multiple scopes. When new scopes are formed or created they are added as a children of their parent scope. Similar to DOM, they also creates a hierarchical structure.

**24)  Explain what is the difference between AngularJS and backbone.js?**

AngularJS combines the functionalities of most of the 3rd party libraries, it supports individual functionalities required to develop HTML5 Apps.  While Backbone.js do their jobs individually.

**25)  Who created Angular JS ?**

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

**Q #1) What do you understand by AngularJS?**

**Answer:**AngularJS is a JavaScript framework that is used for making rich and extensible web applications.

It runs on plain JavaScript and HTML, hence you don’t need any other dependencies to make it work. AngularJS is perfect for Single Page Applications (SPA). It is basically used for binding JavaScript objects with HTML UI elements.

**Q #2) Define the features of AngularJS.**

**Answer: Its features include:**

* The Template (View)
* The Scope (Model)
* The Controller (Controller)
* Services
* Filters
* Directives

**Q #3) Define Data Binding.**

**Answer:** Data binding is an automatic attunement of data between the view and model components.

**Q #4) Distinguish between AngularJs and JavaScript expressions.**

**Answer: There are several differences between AngularJs and JavaScript expressions:**

* We can write AngularJs expressions in HTML, but we cannot write JavaScript expressions in HTML.
* We cannot use conditional iterations, loops, and exceptions in AngularJs, but we can use all of these conditional properties in JavaScript expressions.
* Filters are supported in AngularJs whereas filters are not supported in JavaScript.

**Q #5) Write all the steps to configure a**n**Angular App(ng-app).**

**Answer: To set up an Angular App we must follow certain steps as mentioned below:**

* angular.module will be created at first.
* A controller will be assigned to the module.
* The module will be linked with the HTML template(i.e. UI or View) with an angular app(ng-app).
* The HTML template will be linked with the controller(i.e JS) with an ng-controller directive.

**Q #6) What are the Angular Modules?**

**Answer:** The angular modules collectively define an angular application where we can write the angular code.

Modules contain the different parts of an angular application. A module is created by angular.module function in angular.

**Q #7) What are the directive scopes in AngularJs?**

**Answer: Three directive scopes are available in AngularJs.**

**They are:**

* **Parent scope**– Whatever change you make in your directive that comes from the parent scope, will also reflect in the parent scope, and it is also a default scope.
* **Child scope**– It is a nested scope which inherits a property from the parent scope. Also if any properties and function on the scope are not connected with the parent scope directive, then a new child scope directive is created.
* **Isolated scope**– It is reusable and is used when we build a self-contained directive. It is only used for private and internal use which means it does not contain any properties of the parent scope.

**Q #8) How can we share the data between controllers in AngularJs?**

**Answer:** First, we have to create a service. Service is used to share the data between controllers in AngularJs in a very lucid, easy and fastest way. We use events, $parent, next sibling, and controller by using $rootScope.

**Q #9) What is the digest cycle in AngularJs?**

**Answer:** It is a part of the process of data binding in AngularJs. It compares the old and new version of the scope model value in each digest cycle.

The digest cycle is triggered automatically. We can also enhance the usability by using $apply (), if we want to trigger the digest cycle manually.

**Q #10) Explain the differences between one-way binding and two-way binding.**

**Answer:** One-way binding is used to bind the data from the model to view without updating the HTML template or view automatically.

Thus in order to update the HTML template, we need to write a custom code which will update the view every time whenever a data is binded from model to view.

Whereas, two-way binding is used to bind the data from the model to view and vice versa(i.e view to model) by automatically updating the HTML template without writing any custom code.

**Q #11) Difference between sessionStorage, cookies, and localStorage.**

**Answer: Given below are the various differences.**

**SessionStorage –**The data is stored for a particular session. The data will be lost whenever the browser tab will be closed or after some particular session. Maximum size stored can be up to 5MB.

**LocalStorage –**The data is stored with no expiration date. The data can only be cleared by JavaScript or by clearing the browser cache. Storage limit is maximum than the sessionStorage and cookie.

**Cookies –**It stores the data that has to be sent back to the server with some requests. The cookie's expiration varies on the type and duration set from either the server side or client side. Maximum size stored can be less than 4KB.

**Q #12) What is the role of $routeProvider in AngularJs?**

**Answer:** It is the $routeProvider that helps in navigating between different pages/links without separately loading the page/link whenever a user clicks on a link.

ngRoute config() method is used to configure the routeProvider.

**Q #13) What is the difference between $scope and scope?**

**Answer:** In AngularJs $scope is used to achieve dependency injection and scope is used for linking between View (i.e HTML) and Controller (i.e JS).

**Q #14) How are AngularJs prefixes $ and $$ used?**

**Answer:** $$ variable in AngularJS is used as a private variable, as it is used to prevent accidental code collision with the user code.

Whereas $ prefix can be used to denote angular core functionalities (like a variable, parameter, property or method).

**Q #15) Where can we implement the DOM manipulation in AngularJs?**

**Answer:** Manipulation of DOM is in directives and apart from this it should not exist in controllers’ services or anywhere else.

**Q #16) How can we show that a scope variable should have one-time binding only?**

**Answer:** To show one-time binding we have to use “**::**” in front of the scope.

**Q #17) What is SPA (Single page application) in AngularJs?**

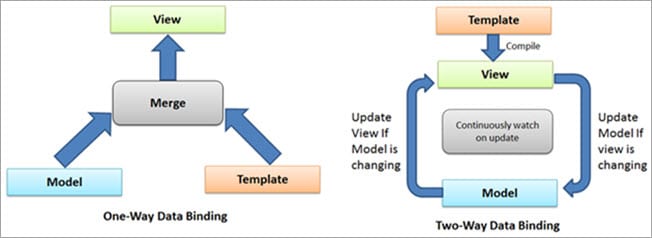
**Answer:** It is a web application which loads a single HTML page and dynamically updates the page as the user connects with the app.

By using AJAX and HTML a fluid and responsive web app can be created by SPA without invariant page reloads. Through this, we can make responsive UI with no page flicker.

**Q #18) How many types of data bindings are there in AngularJs?**

**Answer:** AngularJs supports both one way and two-way binding.

In one way binding if we change the data model, then there will be no dynamic change that you will see in view but in two way binding, there will be a dynamic change whenever a change will be made in the data model.



**Q #19) What are the binding directives in AngularJs?**

**Answer: The binding directives include:**

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

**Q #20) Explain ng-bind and ng-bind-html directives.**

**Answer:**

**ng-bind**: It is a directive which replaces the content of the HTML element with the value of the assigned variable or expression.

The content of the HTML element will change by changing the value of the variable or expression.

It is like ({{expression}}) and the syntax for this is,

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

**ng-bind-html**: It is a directive which binds the content to the HTML element(view) in a secure way. $sanitize service is used to sanitize the content to bind into an HTML element. To do this ‘angular-sanitize.js’ must be included in our application.

**Syntax to write this,**

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

**Q #21) Explain ng-bind-template and ng-non-bindable.**

**Answer:**

**ng-bind-template**: It replaces the text content of the element by interpolation of the template. It can contain multiple double curly markups.

<ANY ELEMENT ng-bind-template= " {{expression1}} {{expression2}} … {{expression n}} "> </ANY ELEMENT>

**Ng-non-bindable**: It specifies AngularJs to not compile the content of this HTML element and its child nodes.

<ANY ELEMENT ng-non-bindable > </ANY ELEMENT>

**Q #22) Explain the ng-model directive in AngularJs.**

**Answer:** This can be a leap hop with the custom HTML input form control( like input, textarea and select) to the application data. It provides form validation behavior with a two-way binding.

*<*input ng-bind="expression"*/>*

**Q #23) Define Factory method in AngularJs.**

**Answer:** It is quite similar to service, factories implement a module pattern in which we use a factory method to generate an object which is used for building models.

In a factory, a method object is returned at the end by creating a new object and adding functions as properties.

**Syntax**:

module.factory(‘factoryName', function);

**Q #24) What is ng-repeat directive in AngularJs?**

**Answer:** It renders or iterates over a collection of items and creates DOM elements. It regularly monitors the source of data to re-render a template in response to a change.

**Syntax:**

<table class="table table-bordered">

<tr ng-repeat="student stuDetails">

<td>{{stu.name}} </td>

<td> {{stu. grade}} </td>

</tr>

</table>

**Q #25) What is a controller in AngularJs?**

**Answer:** A controller is a JavaScript function which is bound to the specified scope. Angular instantiates the new controller object and injects the new scope as a dependency.



A controller can be used to set up the initial state of the scope object and to add behavior to the object.

A controller cannot be used to share code or state across controllers, but instead of that Angular service can be used.

<Any ng-Controller=” expression”>

</Any>

<div ng-app="mainApp" ng-controller="SimpleController">

</div>

**Q #26) What are filters in AngularJs?**

**Answer:** The main work of filters is to modify the data, so that it can be merged into an expression or directive by using a pipe character (it is used for applying filters in an angular symbol of a pipe which is (|) or this is the symbol).

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 as well. A filter is a module provided by AngularJS. There are nine components of a filter which are provided by AngularJS.

**Examples:** currency, date, filter, json, limitTo etc.

**Q #27) What is ng-App directive in AngularJs?**

**Answer:** It is used to define the AngularJs Application. It appoints the root element of an AngularJs application and it is 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) and the other applications need to be bootstrapped manually.

**Example:**

<div ng-app="">

<p>My first expression: {{157 + 122}} </p>

</div>

**Q #28) What is ng-switch in AngularJs?**

**Answer:** It is used to conditionally exchange the structure of DOM on a template which is based on a scope-based expression.

This directive lets you show or hide the HTML element depending on the expression.

<element ng-switch="expression">

<element ng-switch-when="value"></element>

**Q #29) What is the use of a double-click event in AngularJs?**

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

<ANY\_HTML\_ELEMENT ng-dblclick="{expression}">

...

</ANY\_HTML\_ELEMENT>

**Q #30) What are ng-include and ng-click directives in AngularJs?**

**Answer:**

**ng-include** helps to include different files on the main page. The ng-include directive includes HTML from an external file.

The included content will be included as child nodes of the specified element. The value of the ng-include attribute can also be an expression, returning a filename.

By default, the included file must be located on the same domain as the document.

<div ng-include="'myFile.htm'"></div>

ng-click can be used in scenarios like when you click on a button or when you want to do any operation. It tells AngularJS what to do when an HTML element is clicked.

**Example:**

<button ng-click="count = count + 1" ng-init="count=0">OK</button>

The above code will increase the count variable by one whenever the button is clicked.

**Q #31) What is a representational state transfer(REST) in AngularJs?**

**Answer:** REST is an API style that operates over the HTTP request.

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, which is a term used to indicate an API that follows the REST style.

AngularJS is flexible about how RESTful web services are consumed.

**Q #32) What are the AngularJs Global API?**

**Answer:** It is a combination of global JavaScript function which is used to perform tasks like comparing objects, iterating objects and converting data.

**There are some common API functions like:**

* **angular. lowercase:** It converts a string to lowercase string.
* **angular. uppercase:** It converts a string to uppercase string.
* **angular. isString:**It will return true if the current reference is a string.
* **angular. isNumber:** It will return true if the current reference is a number.

**Q #33) What is a provider method in AngularJs?**

**Answer:** A provider is an object which creates a service object by allowing to take more control.

$get() method is used in the provider which returns the service object. The service name and the factory function are the arguments that are passed into the provider method. AngularJS uses $provide to register new providers.

**Syntax:**

serviceApp.provider("logService", function ())

**Q #34) What is Event Handling?**

**Answer:** Event handling in AngularJs is very useful when you want to create advance AngularJs applications.

We need to handle DOM events like mouse clicks, moves, keyboard presses, change events and so on. AngularJs has some listener directives like ng-click, ng-dbl-click, ng-mousedown, ng-keydown, ng-keyup etc.

**Q #35) What is AngularJs DOM?**

**Answer:** AngularJs has some directives which are used to encapsulate AngularJs application data to a disabled attribute of the HTML elements.

**Example:** ng-disabled directive encapsulates the application data to the disabled attributes of HTML DOM element.

<div ng-app="" ng-init="mySwitch=true">

<p>

<button ng-disabled="mySwitch">Click Me!</button>

</p>

<p>

<input type="checkbox" ng-model="mySwitch"/>Button

</p>

<p>

{{ mySwitch }}

</p>

</div>

**Q #36) What are the attributes that can be used during the creation of a new AngularJs directives?**

**Answer:** There are several attributes which can be used during a new directive creation.

**They include:**

1. **Template:**It describes an inline template as a string.
2. **Template URL:**This attribute specifies the AngularJs HTML compiler to replace the custom directive inside a template with the HTML content located inside a separate file.
3. **Replace:**It replaces the current element if the condition is true, if false append this directive to the current element.
4. **Transclude:**It allows you to move the original children of a directive to a location inside the new template.
5. **Scope:**It creates a new scope for this directive rather than inheriting the parent scope.
6. **Controller:**It creates a controller which publishes an API for communicating across the directives.
7. **Require:**It requires another directive to be present to function the current directive efficiently.
8. **Link:** It modifies resulting DOM element instances, adds event listeners, and set up data binding.
9. **Compile:**It modifies 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.

**Q #37) Is Nested Controllers possible or not in AngularJs?**

**Answer:** Yes, it is possible as Nested Controllers are well-defined in a classified manner while using a view.

**Q 38) Is AngularJS well-suited with all browsers?**

**Answer:** Yes, it is companionable with all browsers like Safari, Chrome, Mozilla, Opera, IE etc. as well as Mobile browsers.

**Q 39) Define services in AngularJS.**

**Answer:** AngularJS services are the singleton objects or functions which are used for carrying out definite tasks.

It embraces some corporate ideas and these purposes can be called as controllers, directive, filters and so on.

**Q 40) Explain the advantages of AngularJS.**

**Answer: Advantages of AngularJS include:**

* AngularJS supports MVC form.
* Organize two ways data binding using AngularJS.
* It supports mutual client-server communication.
* It supports simulations.

**Q #41) Difference between services and factory.**

**Answer:** Factories are functions that return the object, while services are constructor functions of the object which is used by a new keyword.

**Syntax:**

**Factory** – module.factory(`factoryName`, function);

**Service** – module.service(`serviceName`, function);

**Q #42) If both factory and service are equivalent, then when should I use them?**

**Answer:** Factory provider is preferred using an object, whereas a service provider is preferred using with class.

**Q #43) Difference between AngularJS and React.JS.**

**Answer:** AngularJS is a TypeScript language based JS framework released in October 2010 by Google. It is a completely free framework and open source that is used in SPA projects (i.e. Single Page Application projects).

React.JS is a javascript library developed by Facebook in March 2013 for building UI. React components can be used on several pages but not as a SPA (i.e. Single Page Application).

**Q #44) Difference between ng-bind and ng-model directive.**

**Answer:** ng-bind directive has one way data bindings, data flows only from object to UI, not vice versa (i.e. $scope>>view) and ng-model directive has two way data bindings, data flows between UI to object and vice versa(i.e. $scope>>view and view>>$scope).

**Q #45) What is the difference between AJAX and AngularJs?**

**Answer:** AJAX stands for Asynchronous JavaScript which is used for sending and getting responses from the server without loading the page.

Whereas, AngularJS is a typescript language based JavaScript framework following the MVC pattern.

**Q #46) Define ng-if, ng-show and ng-hide.**

**Answer:** ng-if directive is used as if clause which removes the HTML element if the expression becomes false.

**Syntax**

<element ng-if=”expression”></element>

ng-show directive is used to show the HTML element if the expression becomes true. And if the expression becomes false then the HTML element will be hidden.

**Syntax**

<element ng-show=”expression”></element>

ng-hide directive is used to hide the HTML element if the expression becomes false.

**Syntax**

<element ng-hide=”expression”></element>

Both ng-show and ng-hide uses the display property method.

**Q #47) What is the difference between ngRoute and ui-router?**

**Answer:** ngRoute is a module developed by angularJS team which was a part of the core angularJS framework. Whereas ui-router was developed by a third-party community to overcome the problems of ngRoute.

ngRoute is a location or URL based routing, and ui-router is a state-based routing which allows nested views.

**Q #48) How to set, get and clear cookies in AngularJs?**

**Answer:** AngularJS has a module called ngCookies, so before injecting ngCookies angular-cookies.js should be included into the application.

**Set Cookies** – Put method is used to set cookies in a key-value format.

$cookies.put(“username”, $scope.username);

**Get Cookies –**Get method is used to get cookies.

$cookies.get(‘username’);

**Clear Cookies –**Remove method is used to remove cookies.

$cookies.remove(‘username’);

Question 1: What is AngularJS?

**Answer**  
  
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.

You can build Angular apps using any HTML editor or a developer IDE such as Visual Studio or Visual Studio 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>*

Continue learning here: [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**

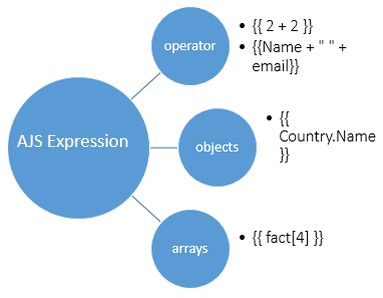
* ngBind,
* ngModel
* ngClass
* ngApp
* ngInit
* ngRepeat

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.

Learn more here: [What Are Directives In AngularJS](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>

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

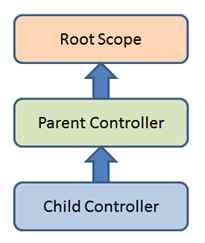
Question 4: Explain currency filter in AngularJS

**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"}}*

**Learn more here:**[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**

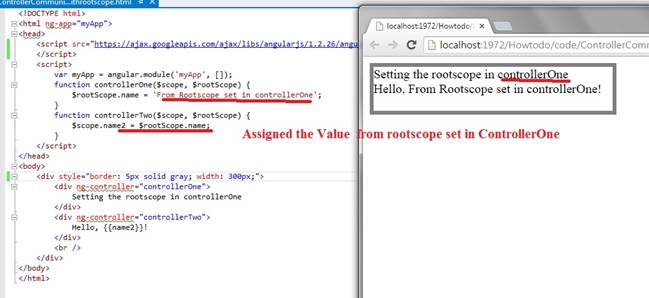
* It provides the APIs to observe model (example $watch).
* It can be nested, so that it limits access to the properties. Nested scopes are either child scope or isolated scope.
* It provides the APIs to propagate any model changes from the outside of "Angular realm" (example $apply).
* It provides context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

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

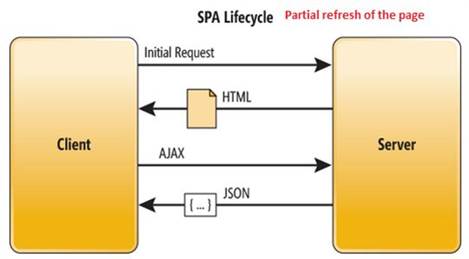
Question 6: What is “$rootScope” in AngularJS?

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

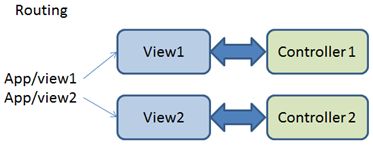
Question 7: What is SPA (Single page application) in AngularJS?

**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.

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

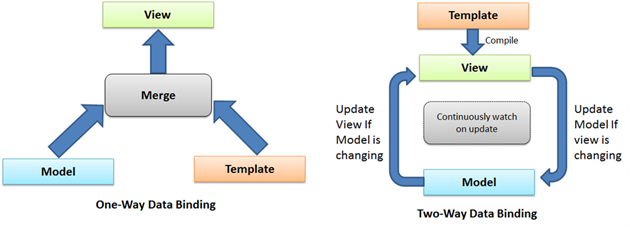
Question 8: How to implement routing in AngularJS?

**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. ]);

Learn more here: [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"/>*  
  
Learn more here: [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.  
  
Learn more here:

* [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>

**Learn more here:**[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.

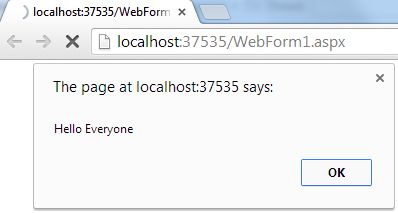
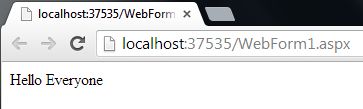
1. <html ng-app xmlns="http://www.w3.org/1999/xhtml">

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>

Learn more here: [$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**

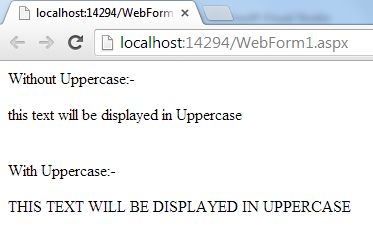
1. AppModule.controller("AppController", function($scope, $window, $log,$http)
2. {
4. });

Learn more here: [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>

  
  
  
Learn more here: [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

Learn more here: [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**

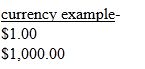
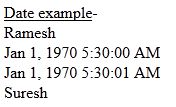
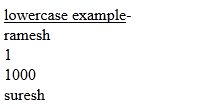
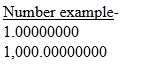
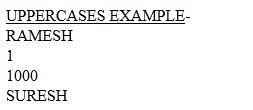
1. <Any ng-Controller=”expression”>
2. </Any>
3. <div ng-app="mainApp" ng-controller="SimpleController">
4. </div>

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

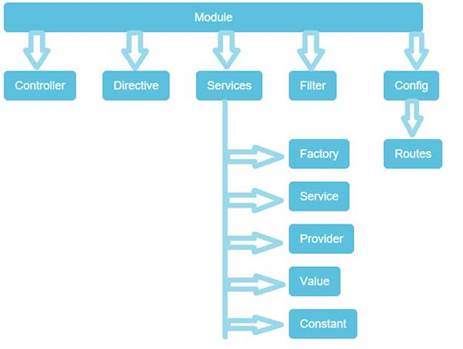
Question 18: Explain Module And Controller In AngularJS.

**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. });

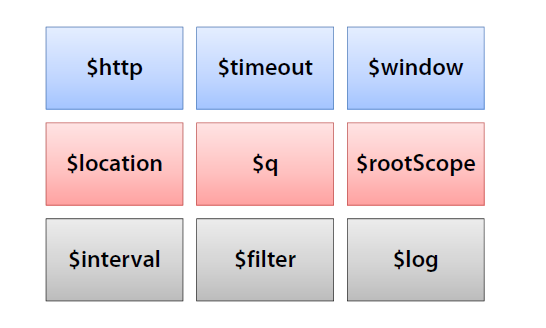
  
  
Learn more here: [Module And Controller In AngularJS](http://www.c-sharpcorner.com/article/module-and-controller-in-angularjs/)

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.

**Learn more here:**[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?

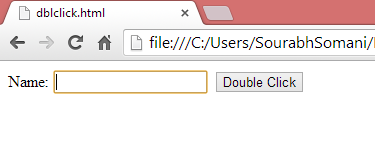
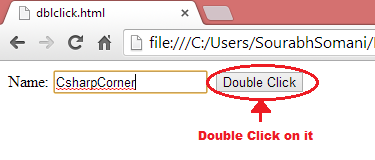
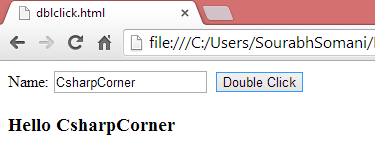
**Answer**ng-dblclick allows you to specify custom behavior on a double-click event of the mouse on the web page. We can use it (ng-dblclick) as an attribute of the HTML element like,

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**

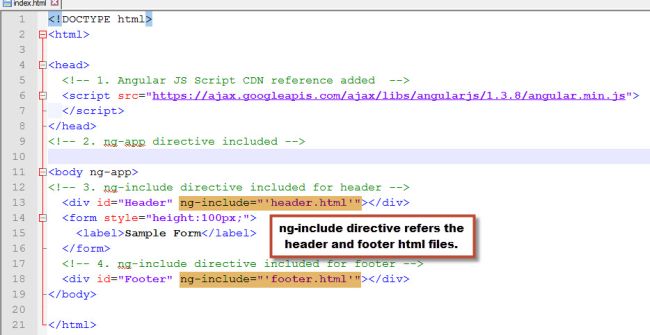
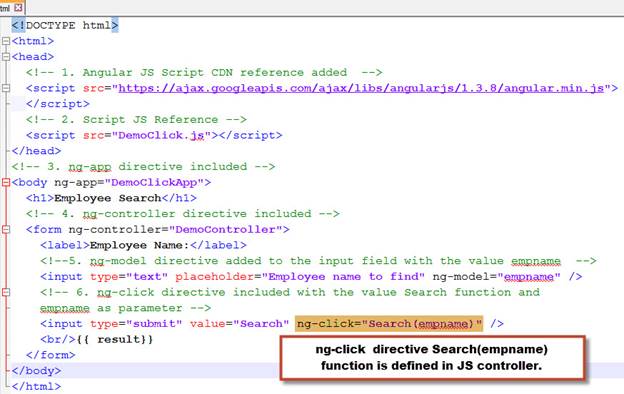
1. <!doctype html>
2. <html ng-app>
4. <head>
5. <script src="angular.min.js"></script>
6. </head>
8. <body>
9. Name:
10. <input ng-model="name" type="text" />
11. <button ng-dblclick="Msg='Hello '+name">
12. Double Click
13. </button>
14. </br>
15. <h3>
16. {{Msg}}</h3>
17. </body>
19. </html>

**Output**

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

**Learn more here:**[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>

**Learn more here:**[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!!  
  
Learn more here:

* [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>

Learn more here: [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.

**Learn more here:**[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 |

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

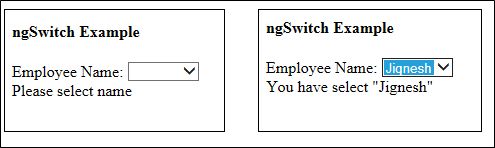
Question 26: Explain ng-Switch Directive in AngularJS.

**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**  
  
  
Learn more here: [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. });

Learn more here: [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 |

**Learn more here:**[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']);

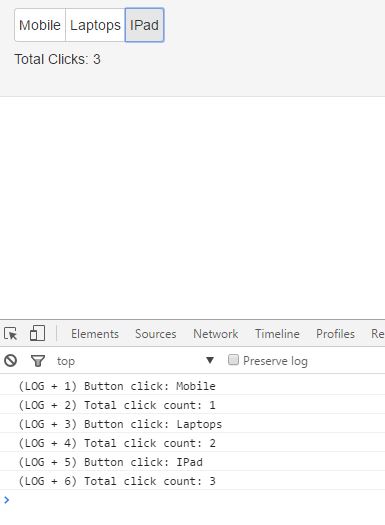
Now, I define another angualr module named testApp in which I inject the ServiceApp module. This testApp module will be used from html page for controller.  
  
**Factory.html**

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,  
  
  
  
Learn more here: [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. });

**Learn more here:**[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.  
  
Learn more here: [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.  
  
**Learn more here:**[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. }]);

**Learn more here:**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.

**Learn more here:**[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 "/>

**Learn more here:**[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.  
  
**Learn more here:**[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.

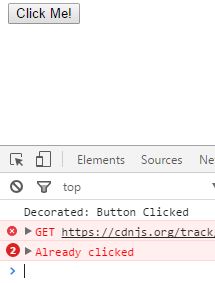
|  |  |
| --- | --- |
| **Name** | **Descriptions** |
| annotate(fn) | Gets the arguments for the specified function, including those that do not correspond to services |
| get(name) | Gets the service object for the specified service name |
| has(name) | Returns true if a service exists for the specified 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**

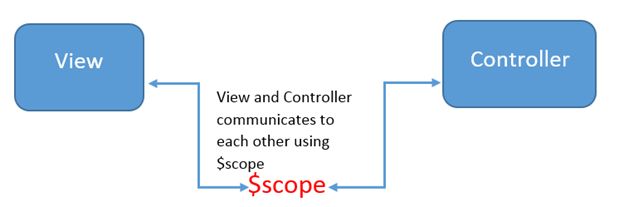
1. <!DOCTYPE html>
2. <html xmlns="http://www.w3.org/1999/xhtml" ng-app="TestApp">
4. <head>
5. <title>Angular Injection</title>
6. <script src="angular.js"></script>
7. <script src="app.js"></script>
8. <script src="Index.js"></script>
10. </head>
12. <body ng-controller="indexController">
13. <div 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 |

**Learn more here:**[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.

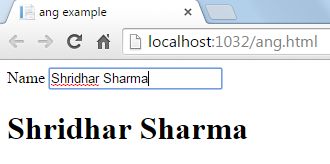
* AngularJS came with RESTful API whereas we don't have that in jQuery.
* AngularJS supports the MVC pattern whereas jQuery doesn't.
* AngularJS has the feature called Two Way Data Binding whereas we don't have that in jQuery.
* Deep Linking Routing is supported by AngularJS whereas jQuery doesn't.
* 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>



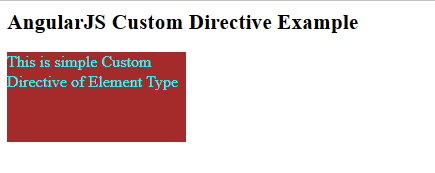
**Learn more here:**[AngularJS Vs jQuery](http://www.c-sharpcorner.com/UploadFile/092589/angularjs-vs-jquery/)

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**

**Learn more here:**[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.

**Learn more here:**[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.

Learn more here: [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**

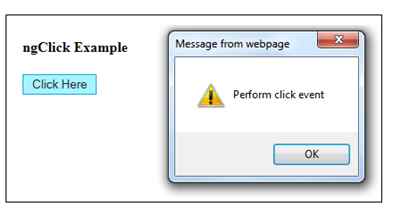
1. <ANY ELEMENT ng-click="expression">
2. ...
3. </ANY ELEMENT>

**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**

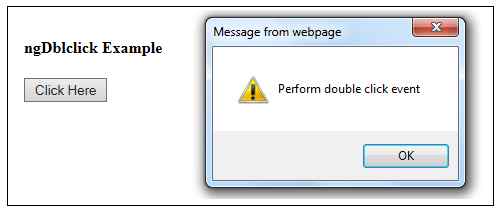
1. <ANY ELEMENT ng- dblclick ="expression">
2. ...
3. </ANY ELEMENT>

**Example**  
**HTML**

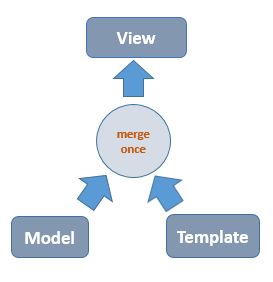
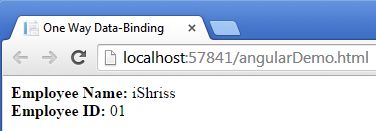
1. <h4>ngDblclick Example</h4>
2. <div ng-controller="HomeController">
3. <button ng-dblclick="dblclick()"> Click Here </button>
4. </div>

**Controller**

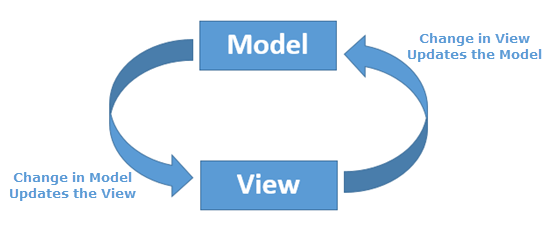
1. var app = angular.module("app", []);
2. app.controller("HomeController", function($scope)
3. {
4. $scope.dblclick = function()
5. {
6. alert('Perform double click event');
7. }
8. });

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

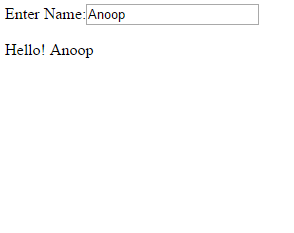
Question 45: What is One-Way Data Binding in AngularJS?

**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**  
**Learn more here:**[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.  
  
  
**Learn more here:**[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 |

**Learn more here:**[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>

**Learn more here:**[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.

[1. What is Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled1)

**Angular 2** is a completely revived component-based [**Javascript framework**](https://www.onlineinterviewquestions.com/javascript/)in which an application is a tree of loosely coupled components. It is a more advanced version of angularJS. It is more of an "all in one" framework so it also helps in creating a single website without getting trapped into different JS frameworks. An Angular 2 is a modular framework in which our code is divided into individual procedures that offer a similar kind of functionality, hence improving the testing, up gradation and maintenance of the application. It has a lot of useful features such as- server-side rendering, cross-platform, and supports more languages than any other framework. It is a new typescript framework built around the concept of components which is paving the way for a better and spacious development. We can even make hybrid applications using Angular 2 which gives us a sharp edge by providing us the flexibility to use the same standard codes for developing other applications.

[2. Why are decorators used in Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled2)

In Angular 2, **decorators** are used as an identifier of class or type of the object that is created by the TypeScript.The Angular 2 identifies the class below decorator call as the definition of the class and extends the decorator specific properties with the class definition.

[3. Explain Angular 2 hidden property ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled3)

The hidden property in Angular 2 is a special case.

* The property is more powerful and is used to bind any property of the elements.
* It is considered the closest cousin of **ngshow** and **nghide**.
* It sets the display property “display: none”.

[4. What is a template in Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled4)

The template in Angular 2 is used to define the views of the AngularJS Application.

[5. List the key components of Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled5)

**The Angular 2 comprises of the following key components:**

* Module – This is used to break the application into the logical pieces of the program code and each piece of code or module is designed to perform a single and unique task.
* Component – This is used to bring the modules together.
* Templates – This is used to define the Views of an Angular JS application.
* Metadata – This is used to add more data to an Angular JS application.
* Service – This component is used to develop the components, which can be used to share in the entire application.

[6. What is the meaning of component lifecycle in Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled6)

The component lifecycle hooks overview the life cycle sequence and the interfaces. Angular manages the life cycle of a component. Angular creates it, renders it. It can also create and render its children. It also checks when its data-bound properties change. It can even destroy it before removing it from the DOM. The life cycle hook offered by angular provides the visibility into these key life moments and the ability to act when they occur. The components go through an entire set of processes or life cycle right from its initiation to the end of the application.

There are a number of lifecycle hooks which are listed below:–

1. ngOnChanges
2. ngOnInit
3. ngDoCheck
4. ngAfterContentInit
5. ngAfterContentChecked
6. ngAfterViewInit
7. ngAfterViewChecked
8. ngOnDestroy

[7. Explain the concept of lazy loading in Angular 2](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled7)

**Lazy loading** is a module which is used to decrease the start-up time. When lazy is used, then our system application does not need to load everything at once. It only needs to load what the user expects to see when the application first loads. The modules which are lazily loaded will only be loaded when the user navigates to their routes. Lazy loading improves the performance of our system applications. It keeps the initial payload small and these smaller payloads lead to faster download speeds. It helps in lowering the resource cost, especially on mobile networks. If a user doesn’t visit a section of the application, they won’t ever download those resources. The concept of lazy loading in angular requires us to format the application in a certain way. All the assets that are to be lazy loaded should be added to its own module. Lazy loading is setup in the main routing file. Lazy loading overcomes the problem of slow loading of applications in their own way which hence improves the loading time of the application.

Lazy loading can be done only in four steps:–

1. Update your route file
2. Install angular-router-loader and add the loader to your webpack configuration file.
3. Define the lazy routes
4. Import the routes to the module.

[8. What would you have in a shared module in Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled8)

**Shared module** is used to import the services in both eager and lazy loaded module. We all know that lazy loaded modules create their own branch on the dependency injection tree. Shared module consists of the services that are registered by the angular in the root app injector. For this, we need not import it in the lazy module because lazy loaded modules already have access to the services defined at the root. Components, pipes and directives are also defined in the shared module. Other modules that import the shared module can use it in their templates. This means that the modules can be imported normally in the lazy loaded module. The shared module contains the code that will be used across the applications and featured modules. It also consists of the common template components. “Dumb components” should also be present in the shared module. It typically consists of some common angular modules too. When you are importing the shared module, you will also need to import the module with its providers, because there is no app module in the test.

[9. How to cache an observable data in Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled9)

**Caching of an observable data** is done with the help of “observable.cache”. We can use caching in order to cache the response in the memory and then, on the next subscription, instead of requesting the remote server again. This operator is used at the end of the string. Caching is important for the performance, especially on bandwidth restricted devices and slow networks. You should have a good understanding of caching while working with promises but while translating it to observable, it is a bit difficult. Therefore, when interacting with observables, we typically set up a subscription on the consumer side and react to values coming through the pipe. We can easily add caching to the observables by adding publishReplay(1) and refCount.

[10. When to use Ngoninit and constructor in Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled10)

**Constructors** are used for initializing class members and also for dependency injection. Ngonlnit is used for the initialization work. Both of these methods are called when the component is created. It is really important that we should know, when to and how to use them. These are used for providing the best structure for your component’s code. A constructor method is a pre-defined method in the constructor class which is only called when the class is instantiated. It is also used for properly initializing the fields. The constructor in Angular 2 is used to create a new instance of the class. Ngonlnit is the class we import when we implement the constructor in order to use it in a class. The method used in this case is ngOnlnit(). This method helps in initializing the directive or the component after the data-bound properties are displayed and the directive or components input is set.

[11. List the modern browsers supported by Angular 2.](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled11)

Angular supports most of the recent browsers some of which are:

* Google Chrome
* Firefox
* Edge
* IE for versions 9-11
* Safari
* iOS 7.1
* Android 4.1
* IE Mobile

[12. How to declare a component in Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled12)

**Components in Angular 2** are simply directives that are always associated with a direct template. Angular 2 components have an extremely well defined life-cycle. When working with angular components, we can make use of interfaces which allows us to implement functionality for different times in a components lifecycle. A component must belong to an NgModule in order for it to be usable by another component or application. Components can even control their runtime behaviour by implementing various Life-cycle hooks.

Declaration of component:

@component ({selector: 'great', template: 'hello {{name}}!'})

Class greet{

Name: string = 'world';

}

Components always have a template and only one component can be instantiated per an element in a template. When a component is instantiated, angular creates a change detector, which is responsible for propagating the component’s building.

[13. List some advantages of Angular 2 over Angular1.](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled13)

**Angular 2** is a re-written version of **Angular1** and not an update. The best way to compare Angular 2 and Angular 1 is by finding out the new features in Angular 2. This way we will be able to find out the advantages of Angular 2 over Angular1 precisely. So, some of the advantages of Angular 2 are:-

|  |  |
| --- | --- |
| Angular 2 | Angular1 |
| Angular 2 is a mobile-oriented framework | Whereas Angular1 was not developed with mobile base in mind. |
| Angular 2 is a versatile framework, i.e. we have more choices for languages. We can use ES5, ES6, Typescript or Dart to write an Angular 2 code | Whereas an Angular1 code can written by using only ES5, ES6 and Dart. We don’t have many choices of language in Angular1. |
| Nowadays, the controllers are replaced by components and Angular 2 is completely component based. | Whereas Angular1 was based on controllers whose scope is now over. |

Angular 2 directly uses the valid HTML DOM element properties and events which reduces the usage of many available built-in directives.

[14. Can you automate porting Angular 1 code to Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled14)

No, currently there is not any such tool available that ports the Angular 1 code to the Angular 2 code.  
In the process of porting, the Angular 1 code to Angular 2, the side by side manual conversion of Angular 1 directives to the **Angular 2 components**takes place because they are two different frameworks and hence requires different approaches to solve the same problem.

**Also, Read:** [80 Best AngularJs interview Questions](https://www.onlineinterviewquestions.com/angular-js-interview-questions/)

[15. How can you handle errors in Angular 2 Applications?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled15)

The Angular 2 Applications provide with the option of error handling.The errors in Angular 2 can be handled by including the React JS catch library and later using the catch function.

* The catch function, which is used after adding the catch library contains the link to the Error handler function.
* And in this error, handler function, the errors are sent to the error console, and also the errors are thrown back to continue the execution.
* So, whenever an error occurs it will be redirected to the error console of the web.

[16. What are Pipes in Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled16)

**Pipes in Angular 2** are used in templates in order to convert them into a content that is user-friendly and readable one within the interpolation braces that is {{release| date}}, here the symbol “|” denotes the pipe.

[17. Explain host decorator in Angular 2 ?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled17)

The **host decorators in Angular 2** bind the properties of components with UI element values.The properties inside a component class definition which are decorated with @HostBinding are accessed in a template from the assigned property that is @HostBinding()title=’Our title'( whatever the title is).

[18. How do you define transition between two states in angular?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled18)

Transitions between two states take place so that we can build simple animations between two states driven by a model attribute. Transition basically means navigating from the current state to a new state. In angular, the transition is an animation-specific function which is used in angular’s animation DSL language. Transition declares the sequence of animation steps that will be executed when the entered value is satisfied. A function is provided an argument for a transition and it will be executed each time a state change occurs. In this, if the function is true, then the animation will run else it won’t get executed.

These animation transitions are placed within the animation triggers. The transition depends upon what the animation was in the previous state and what it will become in the next state. In other words, if a transition is defined that matches the old/current state criteria then the associated animation will be triggered.

**Syntax:**

function transition (stateChangeExpr: string,steps: AnimationMetadata |

AnimationMetadata []):AnimationTransitionMetadata;

[19. Is Angular Modules and ES modules are the same?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled19)

No, Both are different.

[20. What is .angular-cli.json. Where can I find it.](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled20)

**angular-cli.json** is used to configure a project in angular2. You can find it in the root folder of your angular2 Project.

[21. Which module does is required for every Angular 2 app?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled21)

**AppModule**is required for every Angular 2 app.

[22. What is AOT compilation?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled22)

AOT stands for Ahead of Time.It is the compilation in which Angular compiles the components and templates to JavaScript and HTML while developing.

[23. What are Event emitters?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled23)

An Event emitter is a class defined in core module that can be used by components and directives to emit custom events.

[24. What is Angular @ RouteParams?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled24)

The RouteParams are used to map the given URL’s based on the route URLs and they become optional parameters for that route.

[25. What are the new features of Angular 2?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled25)

Angular 2 is a platform that encompasses a wide range of capabilities. Some new features were added in Angular 2 which includes:

* **Universal server rendering-** It is the library which is used to make building universal apps a smooth experience. It is an important feature of Angular 2.
* **A mobile toolkit-** It provides all the mobile toolkit and techniques to build high-performance mobile applications. The web applications which are developed using the mobile toolkit can be loaded on any device with or without internet connection which is a great advantage.
* **A command line interface-**it can generate components, routes, services, and pipes with the help of commands.
* **Data binding-** data binding has been improved in Angular 2. So, whatever DOM element property you need to bind, you just wrap it in square brackets. E.g.-

<img[src]='product.image' />

* **Modular-** various modules have been removed from angular’s core, which has resulted in better performance.
* **Modern-** Angular 2 has been targeted as modern browsers in which various hacks that make angular harder to develop have been removed.

[26. What is the difference between observable and promises?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled26)

The differences between observable and promises are:

1. Observable is a more powerful way of handling HTTP asynchronous requests. Whereas, A promise handles a single event when an asynchronous operation completes or fails.
2. An observable is like a stream which allows passing zero or more events where the callback is called for each event. Whereas, A promise eventually calls the success or failed callback even when you don’t need the notification or the result it provides anymore.
3. Observable works with multiple values for a particular time. Whereas, Promises works with and even returns a single value at a time.
4. Observables can be canceled. Whereas, Promises cannot be canceled.
5. Observable supports map, filter, reduce and similar operators. Whereas, Promises have more readable codes with try/catch and async/await.
6. In observable, one operator ‘retry’ can be used to retry whenever needed. Whereas, Promises cannot be retried. A promise should have access to the original function that returned the promise in order to have a retry capability.

[27. List the differences between Angular 2 components vs. directives.](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled27)

Apart from components, directives are also used in Angular 2 which allows us to attach behavior to elements in DOM. There are certain differences between the components and directives in Angular 2. They are:

1. In Angular 2, a component is a directive with a view whereas a directive is a decorator with no view. Components are the specific type of directive that allows us to utilize web component functionality throughout our application. Whereas, Directive is the mechanism by which we attach behavior to elements.
2. A component is used to break up the application into smaller components. Whereas, Directive is used to design the re-usable components.
3. Components can be used to define pipes. Whereas, We cannot define pipes using directives.
4. Components can be present per DOM element. Whereas, Directive is used to add behavior to an existing DOM element.

[28. What is ECMAScript?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled28)

ECMAScript is a standard for scripting languages. It is a subset of **Javascript**. Languages such as ActionScript, JavaScript use ECMAScript as its core. ECMA stands for European Computer Manufacturer’s Association. Coders commonly use ECMAScript for client-side scripting on the World Wide Web. It is also used for server applications and services. It includes structured, dynamic, functional, and prototype-based features. The ECMAScript was developed by Brendan Eich of Netscape. The ECMAScript is standardized by the ECMA international standards organization in the ECMA-262 and ECMA-402 specifications. It is a programming language which is designed specifically for acting on an existing entity or system. It provides the rules, details, and guidelines that a scripting language must observe to be considered ECMAScript compliant.

[29. What is Traceur Compiler?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled29)

Traceur is a compiler which takes ECMAScript and compiles it down to regular Javascript that runs in your browser. Traceur can be used in several ways like- typing or pasting the ES6 code into the read-eval-print-loop page, or by including traceur in the web page and compiling ES6 code content on the fly, or many other ways. Even traceur is written in ES6, compiled to ES5. The main goal of a traceur compiler is to inform the designs of Javascript features and allows us to write the code in a better manner. Nowadays, traceur compilers are broadly used in Angular 2 platform. It also supports transpilling and type checking via type annotations.

[30. List out the differences between ActivatedRoute and RouterState, with reference to Angular 2.](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled30)

Here are some of the differences between ActivatedRoute and RouterState with reference to Angular 2:-

1. ActivatedRoute consists of the information about a route associated with a component loaded in an outlet. Whereas, RouterState represents the state in which the writer actually is.
2. We need ActivatedRouteSnapchat to traverse all the activated routes. Whereas, during a navigation, after redirects have been applied, the router creates a RouterStateSnapshot.
3. ActivatedRouteSnapshot has old data. When route changes, ActivateRouteSnapshot has data from previous route. Whereas, the RouterState cares about application components, or, to be more specific, about their arrangements.

[31. What do you mean by a structural directive in Angular 2?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled31)

Structural directives are used to manipulate DOM in angular. Structural directives are responsible for HTML layout. By adding, removing, or manipulating LMNs in angular, they shape or reshape the structure of DOM. This structural directive is applied to a host element with the help of other directives. The directives then do whatever it is supposed to do with that host element and its descendants. Structural directives can be easily recognized. It can also delay the instantiation of a component or an element. It can also be used for cosmetic effect or manually handling the timing of the loading of components. Structural directives are bound to a template. The two most common structural directives are “ngIf” and “ngFor”. The process occurring in a structural directive is dynamic.

[32. What do you understand by a template variable? How is it used?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled32)

A template in Angular 2 is used to instantiate embedded views. A template variable can be accessed in two ways. Either by placing a directive on an element and have the template variable for this embedded view injected into the constructor of the directive using the template variable token, or you can query for the template variable from a component or a directive via the query. A template variable in Angular 2 is a reference to a DOM element or directive within a template. Template variables are used to access the values of DOM element properties. It is declared with the help of “#” and “ref-“as a prefix. For example: – #myVar and ref-myVar. Template variable names cannot be made duplicate as in this way, it might give unpredictable values. The scope of a reference variable is the entire template. It can be used anywhere inside a template. In Angular 2, a component needs to have a view and to define a view, a template variable is used. It allows us to express data and property binding, event binding and template concerns.

[33. What is the difference between constructor and ngOnlnit in Angular js?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled33)

The comprehensive comparison that taps into components initialization process is given below:-

1. ngonInit is just a method in a class which structurally is not different to any other method in a class. Whereas, a constructor is a completely different thing. It will be called when an instance of a class is created.
2. A class constructor in angular is used to inject dependencies, which is called constructor injection pattern. Whereas, when ngOnInit is called, it has finished creating a component DOM, injected all required dependencies through constructor and processed input bindings.
3. A constructor is a default method of the class that is executed when the class is instantiated. Whereas, ngOnInit is a life cycle hook called by Angular 2 to indicate that angular is done creating the component.
4. ngOnInit relies on the binding of the component. Whereas, it is not the case when a constructor is used.

[34. What is the use of ngForTrackBy directive?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled34)

For iterating over a collection in Angular 2, the ngFor directive is used which instantiates a template once per item from the collection. If a data needs to be changed at some point in the collection, then a problem occurs because angular cannot keep a track of items in the collection and has no knowledge of the items which were added or deleted. This results in the deletion of all the DOM elements that are associated with the data and are again created. If the collection is big, then it becomes more complicated because a lot of DOM manipulation occurs which are expensive. So, to solve this problem, a trackBy function is used which takes the index and the current item as arguments and returns the unique identifier for this item.

[35. How will you convert a string into a percentage?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled35)

To convert a string into a percentage format, a percent filter is used.

[36. Explain component specific hooks?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled36)

Below are few component specific hooks in Angular2.

* ngafterContentinit: It initializes the component content
* ngAfterConctentChecked: It checks the binding of the external content.
* ngafterViewinit: It creates the component view.
* ngAfterviewChecked: It checks the bindings of the component’s view.

[37. What is CLI?](https://www.onlineinterviewquestions.com/angular2-interview-questions/" \l "collapseUnfiled37)

CLI is the acronym of Command Line Interface, which can be used to create the Angular JS application.Using CLI, you can also create a unit and end-to-end tests for the Angular application.

**Example Angular Basic Interview Questions and Answers**

*Note: Important keywords are underlined in the answers. Bonus points if the candidate mentions them!*

**Question 1: Write an example of a simple HTML document with some header information and page content.**  
*Requirement: Basic HTML skills*

**Answer:** HTML documents are all different, but they follow a basic structure of head and body. Here you‘re checking the candidate has a good grasp of HTML document structure and basic tags such as DOCTYPE, html, head, title, meta, body, h1, p, etc.

For example:

*<!DOCTYPE html>*

*<html>*

*<head>*

*<title>Page Title</title>*

*<meta charset="UTF-8">*

*<meta name="description" content="Page description">*

*</head>*

*<body>*

*<h1>Interview Example Web Page</h1>*

*<p>Some content goes here</p>*

*</body>*

*</html>*

**Question 2: Briefly explain the CSS box model. Write some code snippets to describe show what you mean.**  
*Requirement: Basic CSS skills*

**Answer:** CSS is the language that describes how webpages look. Every front-end developer should have good CSS knowledge. Good candidates will be able to describe CSS concepts concisely.

The CSS box model refers to the layout and design of HTML elements. It‘s a box shape that wraps around each HTML element. A box is made up of its content, padding, border and margin.

* Content of the box
* Padding
* Border
* Margin

(the same padding on all 4 sides)

*padding: 25px;*

(padding for the top, right, bottom, left)

*padding: 25px 50px 75px 100px;*

(top/bottom padding 25 pixels, right/left padding 50 pixels)

*padding: 25px 50px;*

**Question 3: In JavaScript, how can the style of an HTML element be changed?**  
*Requirement: Basic JavaScript skills*

**Answer:**For example, to change the font size:

document.getElementById(“someElement").style.fontSize = "20";

**Question 4: Write some code for a basic class in TypeScript with a constructor and a method.**  
*Requirement: Basic TypeScript skills*

**Answer:**Here‘s a simple [class](https://www.typescriptlang.org/docs/handbook/classes.html) that is created with a greeting message which can be retrieved with the greet() function.

*class Greeter {*

*greeting: string;*

*constructor(message: string) {*

*this.greeting = message;*

*}*

*greet() {*

*return "Hello, " + this.greeting;*

*}*

}

*let greeter = new Greeter("world");*

**Question 5: What are Single Page Applications? How do they work in Angular?**  
*Requirement: Foundational Angular knowledge*

**Answer:** Single Page Applications (SPAs) are web applications that use only one HTML page. As the user interacts with the page, new content is dynamically updated on that master page. Navigation between pages happens without refreshing the whole page. Angular uses AJAX and to dynamically update HTML elements. Angular Routing can be used to make SPAs. The result is an application that feels more like a desktop app rather than a webpage.

**Question 6: What‘s the basic syntax of a Decorator in Angular?**  
*Requirement: Foundational Angular Knowledge*

**Answer:** *@()* with optional parameters.

**Question 7: What is [(ngModel)] used for?**  
*Requirement: Foundational Angular Knowledge*

**Answer:** Two-way data binding.

**Question 8: What are the basic parts of an Angular application?**  
*Requirement: Foundational Angular Knowledge*

**Answer:** Modules, Component, Data Binding, Template, Directives, Dependency Injection, Services, Routing.

**Question 9: Tell me about a time you received feedback on a task.**  
*Requirement: Following instructions and receiving feedback*

**Answer:** This is a typical open-ended question. The candidate should demonstrate they can accept, understand and act on feedback.

**Question 10: Describe how you would approach solving (some problem) on a high level?**  
*Requirement: Thinking like a programmer*

**Answer:**In this question, the problem should be directly related to the work the candidate will actually be doing. You aren‘t looking for a perfect answer or even necessarily a correct answer. Instead, listen to how they approach solving a problem, their ability to break a problem down into parts, and if they can anticipate problems.

**Question 11: What are some advantages of using Angular framework for building web applications?**  
*Requirement: Expert Angular knowledge*

**Answer:**Advantages of using the Angular framework include:

* Angular does lots of things for you under the hood. It saves time for developers by doing a lot of the work for them like writing tedious DOM manipulation tasks
* TypeScript and the Angular framework allow you to catch errors much earlier
* In many cases has faster performance than traditional web development techniques
* Can give web apps the feel of a desktop application
* It separates out the code of an application to make it easier for multiple developers to work on an app and easier to test
* More consistent code base that‘s easy to maintain
* Big developer community

**Question 12: What function is called when an object is created in TypeScript? What is it‘s basic syntax in  
TypeScript code?**  
*Requirement: TypeScript knowledge*

**Answer:**The constructor function is called. It‘s syntax is:  *Constructor(){}*

**Question 13: In Angular, how can you interact between Parent and Child components?**  
*Requirement: Expert Angular knowledge*

**Answer:** When passing data from Parent to Child component, you can use the *@Input* decorator in the Child component. When passing data from Child to Parent component, you can use the *@Output* decorator in the Child component.

**Question 14: Write an example usage of ngFor for displaying all items from an array *’Items‘* in a list with  
<li>.**  
*Requirement: Expert Angular knowledge*

**Answer:**

*<li \*ngFor=”let item of Items”>*

*{{item}}*

*</li>*

**Question 15: What is the sequence of Angular Lifecycle Hooks?**  
*Requirement: Foundational Angular knowledge*

**Answer:**OnChange()  – OnInit() –  DoCheck() – AfterContentInit()  – AfterContentChecked() – AfterViewInit()  – AfterViewChecked() – OnDestroy().

**Question 16: If you provide a service in two components‘ “providers” section of @Component decorator, how many  
instances of service shall get created?**  
*Requirement: Foundational Angular knowledge*

**Answer:**2

**Question 17: What is the main difference between constructor and ngOnInit?**  
*Requirement: Foundational Angular knowledge*

**Answer:**The constructor is a feature of the class itself, not Angular. The main difference is that Angular will launch ngOnInit after it has finished configuring the component. Meaning, it is a signal through which the @Input() and other banding properties and decorated properties are available in ngOnInit, but are not defined within the constructor by design.

**Angular Advanced Interview Questions**

Here are some more advanced and technical interview questions and answers for experienced Angular developers. Use them to pick out the right Angular developers with the skills to build your web app.

An expert Angular developer has to know the Angular framework inside and out. They will also be able to design efficient applications, write clean and robust code, work effectively with your team, and pass on their experience to junior developers. Remember to list your requirements before you choose your questions.

**Skill Requirements for Senior Angular Developers**

* Expert Angular knowledge and its different versions (2, 4, 5, 6, 7)
* Component based architecture
* Designing for specific requirements (e.g. security, scalability, optimization)
* Maintaining and upgrading applications
* Experience in frameworks/toolkits/libraries you use
* Efficient programming and clean code
* Debugging
* End-to-end testing and unit testing
* Leadership skills
* Clear communication skills
* Mentoring less experienced developers

**Example Angular Advanced Interview Questions and Answers**

*Note: Important keywords are underlined in the answers. Look out for them in interviews!*

[Or save yourself time and request a team to match your needs right away.](https://www.devteam.space/get_started/)

**Question 18: What modules should you import in Angular to use [(ngModel)] and reactive forms?**  
*Requirement: Middle Angular knowledge, Tools/libraries*

**Answer:** FormsModule and Reactiveforms Module.

**Question 19: How similar is AngularJS to Angular 2?**  
*Requirement: Middle Angular knowledge*

**Answer:** Both are front-end frameworks maintained by Google, but Angular 2 is not a simple update of AngularJS, it is a new framework written from scratch. Updating an app from AngularJS to Angular 2 would require a complete rewrite of the code.

**Question 20: What were some features introduced in the different versions of Angular (2, 4, 5 and 6)?**  
*Requirement: Expert Angular knowledge, Component-based architecture*

**Answer:**

Angular 2:

* Complete rewrite of the Angular framework
* Component-based rather than controllers/view/$scope. This allows more code to be reused, easier communication between components and easier testing
* Much faster
* Support for mobile devices
* More language choices such as TypeScript

Angular 4:

* An update to Angular 2, not a complete rewrite. Updating from Angular 2 to 4 just requires updating the core libraries
* Improvements to Ahead-of-time (AOT) generated code
* Support for new versions of TypeScript
* Animation packages are removed from the core package
* Else block

Angular 5:

* Focused on making Angular smaller and faster to use
* Http is depreciated and HttpClient API client is now recommended for all apps
* Supports TypeScript 2.3
* Introduction of a build optimizer
* Angular Universal State Transfer API
* Improvements to the Angular Compiler
* Router Lifecycle Events
* Better cross-browser standardization

Angular 6:

* Better service worker support
* Better URL serialization
* Ivy rendering engine
* ng update and ng add
* <template> element completely removed
* Angular Elements/Custom Elements
* Form validation changes
* Schematics

**Question 21: What is Transpiling in Angular?**  
*Requirement: Middle Angular knowledge, TypeScript*

**Answer:** Transpiling means converting the source code of one programming language into another. In Angular, that usually means converting TypeScript into JavaScript. You can write the code for your Angular application in TypeScript (or another language such as Dart) that is then transpiled to JavaScript for the application. This happens internally and automatically.

**Question 22: What is AOT Compilation?**  
*Requirement: Expert Angular knowledge, Optimization*

**Answer:** AOT refers to Ahead-of-time compilation. In Angular, it means that the code you write for your application is compiled at build time before the application is run in a browser.  It‘s an alternative to Just-in-time compilation, where code is compiled just before it is run in the browser. AOT compilation can lead to better application performance.

**Question 23: What are HTTP Interceptors?**  
*Requirement: Middle Angular knowledge*

**Answer:** Interceptor is just a fancy word for a function that receives requests/responses before they are processed/sent to the server. You should use interceptors if you want to pre-process many types of requests in one way. For example, you need to set the authorization header Bearer for all requests:

token.interceptor.ts

import { Injectable } from '@angular/core';

import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from '@angular/common/http';

import { Observable } from 'rxjs/Observable';

@Injectable()

export class TokenInterceptor implements HttpInterceptor {

public intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

*const token = localStorage.getItem('token') as string;*

if (token) {

*req = req.clone({*

*setHeaders: {*

*'Authorization': `Bearer ${token}`*

*}*

*});*

*}*

*return next.handle(req);*

*}*

*}*

*And register the interceptor as singleton in the module providers:*

*app.module.ts*

*import { NgModule } from '@angular/core';*

*import { BrowserModule } from '@angular/platform-browser';*

*import { HTTP\_INTERCEPTORS } from '@angular/common/http';*

*import { AppComponent } from './app.component';*

*import { TokenInterceptor } from './token.interceptor';*

*@NgModule({*

*imports: [*

*BrowserModule*

*],*

*declarations: [*

*AppComponent*

*],*

*bootstrap: [AppComponent],*

*providers: [{*

*provide: HTTP\_INTERCEPTORS,*

*useClass: TokenInterceptor,*

*multi: true // < - - - - an array of interceptors can be registered*

*}]*

*})*

*export class AppModule {}*

**Question 24: How many Change Detectors can there be in the whole application?**  
*Requirement: Expert Angular knowledge*

**Answer:** Each component has its own ChangeDetector. All Change Detectors are inherited from AbstractChangeDetector.

**Question 25: What change detection strategies do you know?**  
*Requirement: Expert Angular knowledge*

**Answer:** There are two strategies – Default and OnPush. If all components use the default strategy, Zone checks the entire tree regardless of where the change occurred. To inform Angular that we will comply with the performance improvement conditions, we need to use the onpush change detection strategy. This will tell Angular that our component depends only on the input and any object that is passed to it should be considered immutable. This is all built on the Principle of the mile automaton, where the current state depends only on the input values.

**Question 26: What is Change Detection, how does Change Detection Mechanism work?**  
*Requirement: Expert Angular knowledge*

**Answer:**Change Detection is the process of synchronizing a model with a view. In Angular, the flow of information is unidirectional, even when using the ng Model to implement two-way binding, which is syntactic sugar on top of a unidirectional flow.

Change Detection Mechanism-moves only forward and never looks back, starting from the root (root) component to the last. This is the meaning of one-way data flow. The architecture of an Angular application is very simple — the tree of components. Each component points to a child, but the child does not point to a parent. One-way flow eliminates the need for a $digest loop.

**Question 27: How do you update the view if your data model is updated outside the ‘Zone’?**  
*Requirement: Expert Angular knowledge*

**Answer:**

1. Using the ApplicationRef.prototype.tick method, which will run change detection on the entire component tree.
2. Using NgZone.prototype.run method, which will also run change detection on the entire tree. The run method under the hood itself calls tick, and the parameter takes the function you want to perform before tick.
3. Using the ChangeDetectorRef.prototype.detectChanges method, which will launch change detection on the current component and its children.

**Question 28: Why do we need lazy loading of modules and how is it implemented?**  
*Requirement: Middle Angular knowledge*

**Answer:**Lazy loading of modules is needed to break the code into pieces. When downloading the app in the browser, it doesn’t load all of the application code. During the transition to the route with lazy loading, the module has to load the code into a browser.

Exemple for using lazy loading modules:

{ path: ‘example’, loadChildren: ‘./example/example.module#ExampleModule’, component: PublicComponent },

**Question 29: What are Core and Shared modules for?**  
*Requirement: Middle Angular knowledge*

**Answer:**A *Shared module* serves as a generic module for all modules, components, directives, pipes, etc., which are not required to be in a single copy for the application but need to be imported into many different modules.

A *Core module* is a place to store services that you need to have in the form of singleton for the entire application (for example, a user authorization service with data storage about it).

**Question 30: What are some points to consider when optimizing an Angular 6 application for performance?**  
*Requirement: Application performance optimization*

**Answer:** There are many ways, some ideas include:

AOT compilation, bundling and uglifying the application, tree shaking, lazy loading, separating dependencies and devDependencies, Using OnPush and TrackBy, removing unnecessary 3rd party libraries and import statements, avoid computing values within the template,

**Question 31: What are some important practices to secure an Angular application?**  
*Requirement: Designing for security*

**Answer:** Some basic guidelines include:

* Check that all requests come from within your own web app and not external websites
* Sanitize all input data
* Use Angular template instead of DOM APIs
* Content Security Policies
* Validate all data with server-side code
* Use an offline template compiler
* Avoid including external URLs in your application
* Make JSON responses non-executable
* Keep all libraries and frameworks up-to-date

**Question 32: What‘s the difference between unit testing and end-to-end testing? What are some testing tools you  
would use for an Angular application?**  
*Requirement: End-to-end and unit testing*

**Answer:**Unit testing is a technique to test that isolated segments of code are functioning properly. End-to-end testing involves checking that entire sets of components to make sure they are working together properly and that the application is working as you would expect. End-to-end tests often simulate user interactions to test that an app is functioning as it should. Jasmine and Karma are all great testing tools.

**Open-Ended Questions**

Once you‘ve established that your developer is an expert with some Angular coding interview questions, you should ask some less technical questions. These should spark a discussion and you should tailor them to fit your own job requirements. Don‘t be afraid to ask follow up questions!

**Question 33: Describe a time you fixed a bug/error in an application. How did you approach the problem? What  
debugging tools did you use? What did you learn from this experience?**  
*Requirement: Debugging, Breaking down a problem into parts*

Debugging is one of the key skills for any software developer. However, the real skill is in breaking the problem down in a practical way rather than finding small errors in code snippets. Debugging often takes hours or even days, so you don‘t have time in an interview setting. Asking these questions will give you an idea of how your candidate approaches errors and bugs.

**Answer:** In the candidate‘s response you should look out for things like:

* A measured, scientific approach
* Breaking down the problem into parts
* Finding out how to reproduce the error
* Expressing and then testing assumptions
* Looking at stack traces
* Getting someone else to help/take a look
* Searching the internet for others that have had the same problem
* Writing tests to check if the bug returns
* Checking the rest of the code for similar errors
* Turn problems into learning experiences

**Question 34: What’s the most important thing to look for or check when reviewing another team member’s code?**  
*Requirement: Mentoring less experienced developers, Leadership skills*

**Answer:**Here you‘re checking for analysis skills, knowledge of mistakes that less experienced developers make, keeping in mind the larger project and attention to detail.

A good answer might mention code functionality, readability and style conventions, security flaws that could lead to system vulnerabilities, simplicity, regulatory requirements, or resource optimization.

**Question 35: What tools & practices do you consider necessary for Continuous Integration and Delivery of an  
Angular application?**  
*Requirement: DevOps systems design, Maintaining and upgrading applications*

100+ AngularJS Interview Questions And Answers

What Is AngularJS Technology?

AngularJS is a JavaScript-based Web Development Framework for creating dynamic web pages. It incorporates HTML as the template language and enables the developer to extend the HTML tags for representing the application components more clearly.

It was initially developed by Google, who later published it as an open-source and licensed under the Apache License version 2.0.

What Type Of Architecture Does AngularJS Support?

Angular proposes to create applications based on the MVC architecture. It helps to develop a maintainable solution that is easy to test at the client-end.

It requires the following three core components.

* Template – the view part
* Scope – the model
* Controller – the control part

AngularJS customizes the HTML attributes with directives and data binding using expressions.

What Does Template Mean In AngularJS?

The templates are unique HTML components in Angular applications. They are more like a static version of a web page with some additional properties to inject and render data at runtime.

What Does Scope Mean In AngularJS?

A scope is an object which defines the model for an Angular application. It provisions data fields to pass in templates for presenting to the user.

It can even bind functions for responding to actions which a user could perform.

What Does Controller Mean In AngularJS?

The controller takes the form of a function which accepts an empty scope object as the argument, delegates it to the fields and functions for exposing to the user.

What Does Directive Mean In AngularJS?

AngularJS directives are an extension of the HTML attributes. Following are three standard directives in Angular JS:

* **ng-app:** It marks the HTML element that Angular intends to be the root element of our application. The custom attributes use spinal-case whereas the corresponding directives follow the camelCase.
* **ng-model:** It binds values of HTML elements (such as input, select, textarea) to the application data.
* **ng-bind:** It binds application modal data to the HTML template view.

What Are The Main Features Of Angular?

Here is the list of AngularJS features that makes it the hottest tech for web dev.

* **Data-binding –** Handles synchronization of data across model, controllers, and view.
* **Scope –** Object representing the model, acts as a glue layer between controller and view.
* **Controllers –** These are JS functions bound to the scope object.
* **Services –** Substitutable objects that are wired together using dependency injection. e.g. $location service.
* **Filters –** Formats the value of an expression for displaying to the user. e.g., uppercase, lowercase.
* **Directives –** These are extended HTML attributes start with the “ng-” prefix. e.g., the ng-app directive used to initialize the angular app.
* **Templates –** HTML code including AngularJS specific elements and attributes.
* **Routing –** It’s an approach to switch views.
* **MVC pattern –** A design pattern made up of three parts.
  + **Model –** Represents data, could be static data from a JSON file or dynamic data from a database.
  + **View –** Renders data for the user.
  + **Controller –** Gives control over the model and view for collating information to the user.
* **Deep linking –** Enables the encoding of the application state in the URL and vice versa.
* **Dependency injection –** A design pattern to let the components injected into each other as dependencies.

Why Would You As A Developer Choose Angular?

Following are some of the key reasons to choose AngularJS as your web development framework:

* It follows the MVC design pattern which allows segregating an application into different components (called Model, View, and Controller) thus making it easy to maintain.
* It allows the HTML to extend by adding directives to the HTML markup. This feature helps in defining dynamic templates which can include new attributes, tags, and expressions.
* It allows the creation of user-defined directives and reusable components. These directives help the developer to concentrate on creating logic, thus enabling them to work efficiently.
* It supports two-way data binding, i.e., enables automatic synchronization of data between model and view components. Thus, any update in the model gets reflected in the view automatically. And there is no need to add any Javascript code or event listeners to notify the data changes.
* It encapsulates the behavior of your application in controllers which gets instantiated with the help of dependency injection.
* It supports built-in services to perform routine tasks for web applications. For example, it provides $http service to communicate with the REST service.
* It makes the development and testing of the application’s JavaScript code easy.
* Also, AngularJS has a mature community to help developers. It has broad support over the internet.

List Down The Popular AngularJS IDE Plugins/Extensions For Web Development?

Here is a list of IDE Plugins and Extensions which can enhance the way you code with AngularJS:

* Sublime Text
* WebStorm
* Eclipse
* Netbeans
* Visual Studio 2012/2013 Express or higher
* TextMate
* Brackets
* ATOM

What Are The Steps Involved In The Boot Process For AngularJS?

Whenever a web page loads in the browser, the following steps get executed in the background.

1. First, the HTML file containing the code gets loaded into the browser. After that, the JavaScript file mentioned in the HTML code gets loaded. It then creates a global object for angular. Now, the JavaScript which displays the controller functions gets executed.
2. In this step, AngularJS browses the complete HTML code to locate the views. If the same is available, then Angular links it to the corresponding controller function.
3. In this step, AngularJS initiates the execution of required controller functions. Next, it populates the views with data from the model identified by the controller. With this the page is ready.

Which Browsers Are Compatible With Angular?

AngularJS works fine with the latest versions of Safari, Chrome, Firefox, Opera 15+, and IE9+ (Internet Explorer). However, below is the recent compatibility chart.

**Browser**

**Compatibility**

**Google Chrome**

Latest

**Mozilla Firefox**

Latest

**Edge**

Two most recent major versions

**IE**

v11 v10 v9

**IE Mobile**

v11

**Safari**

Two most recent major versions

**iOS**

Two most recent major versions

**Android**

Nougat (v7.0) Marshmallow (v6.0) Lollipop (v5.0, v5.1) KitKat (v4.4)

What Are Polyfills?

Polyfills are compatibility scripts which enable Angular support in browsers which don’t have all modern HTML5 or above features.

How Do You Enable Polyfills?

We can get it enabled from the Angular CLI via the **“src/polyfills.ts”** file.

This file has got all the mandatory and optional polyfills required to run an Angular application.

What Are The Security Features Provided By AngularJS?

AngularJS provides built-in protection from the following security flaws.

* It prevents cross-site scripting attacks: Cross-site scripting is a technique where anyone can send a request from the client side and can get confidential information easily.
* It prevents HTML injection attacks.
* It prevents XSRF protection for server-side communication: “Auth token” mechanism can handle it. When the user logins for the first time a user id and password is sent to the server, and it will, in turn, return an auth token. Now, this token does the authentication in future transactions.

What Are The Security Risks That A Web Developer Should Manage While Developing An AngularJS App?

Following are the most critical web application development flaws that a developer should take care of:

* Injection attack
* Broken authentication and Session management
* Cross-Site Scripting (XSS)
* Insecure direct object references
* Security misconfiguration
* Sensitive data exposure
* Missing function level access control
* Cross-Site request forgery (CSRF)
* Using components posing vulnerabilities
* In-validated redirects and forwards

What Are The Most Common Directives Used In AngularJS Applications?

AngularJS extends the behavior of HTML and DOM elements with new attributes called Directives. It directs the AngularJS’s HTML compiler ($compile) to attach a unique action to that DOM element. This AngularJS component starts with the prefix “ng.”

Following is the list of AngularJS built-in directives.

* **ng-bind –** The ng-bind directive tells AngularJS to replace the content of an HTML element with the value of a given variable, or expression.  
  If there is any change in the value of the given variable or expression, then the content of the specified HTML element will also be updated accordingly. It supports one-way binding only.
* **ng-model –** This directive is used to bind the value of HTML controls (input, select, text area) to application data. It is responsible for linking the view into the model. Directives such as ‘input,’ ‘text area,’ and ‘select’ require it. It supports two-way data binding.
* **ng-class –**This directive dynamically binds one or more CSS classes to an HTML element. The value of the ng-class directive can be a string, an object, or an array.
* **ng-app –** Just like the “Main()” function of Java language, this directive marks the beginning of the application to AngularJS’s HTML compiler ($compile). If we do not use this directive first, an error gets generated.
* **ng-init –** This is used to initialize the application data so that we can use it in the block where it is declared. If an application requires local data like a single value or an array of values, this can be achieved using the ng-init directive.
* **ng-repeat –** This repeats a set of HTML statements for the defined number of times. The set of HTML statements will be repeated once per item in a collection. This collection must be an array or an object.

We can even create custom directives and use them in our AngularJS Application.

Why Are Expressions Used In AngularJS?

AngularJS binds data to HTML using Expressions. It can be written inside double braces: **{{ expression}}**or inside a directive as **ng-bind=”expression”**. AngularJS evaluates the expression and substitutes the result in place of the expression.

AngularJS expressions are much like JavaScript expressions. They can include literals, operators, and variables.

**For example –**

{{ 2 + 2 }} (numbers)

{{Name + " " + email}} (string)

{{ Country.Name }} (object)

{{ fact[4] }} (array)

What Does A Filter Do In Angular?

A Filter in Angular changes or transforms the data before passing it to the view. These Filters work in combination with AngularJS expressions or directives.

AngularJS uses pipe character **(“|”)** to add filters to the expressions or directives.

**For example:**

<p> {{ bid | currency }} </p>

The above example is an expression enclosed in the curly braces using the **currency filter**.

It is important to note that filters are case-sensitive.

Which Are The Filters Angular Supports?

AngularJS provides the following filters to transform data.

* **Currency –** It is used to format a number to a currency format.
* **Date –** It allows date-formatting to a specified format.
* **Filter –** It chooses a subset of items from an array.
* **JSON –** It formats an object to a JSON string.
* **LimitTo –** Its purpose is to create an array or string containing a specified number of elements/characters either from the beginning or the end of the source. It depends on the value and sign (positive or negative) of the limit.
* **Lowercase –** This filter converts a string to lower case.
* **Number –** It formats a number as text.
* **OrderBy –** It enables to sort an array. By default, sorting of strings happens alphabetically. And the sorting of numbers happens numerically. It also supports a comparator function where we can define what will be counted as a match or not.
* **Uppercase –** This filter converts a string to upper case.

What Are Angular Prefixes $ And $$?

To prevent accidental name collisions within the code, AngularJS prefixes the names of public objects with **$** and the private ones with **$$**.

However, we should not use these literals (**$** or **$$)** for any other purposes or reasons.

What Are Different Ways To Invoke A Directive?

There are four different ways to invoke a directive in an angular application. They are as follows.

**1) As an attribute:**

<span my-directive></span>

**2) As a class:**

<span class="my-directive: expression;"></span>

**3) As an element:**

<my-directive></my-directive>

**4) As a comment:**

<!-- directive: my-directive expression -->

What Is A Singleton Pattern? How Does Angular Use It?

Generally, the singleton pattern is a design approach which allows us to limit the instantiation of a class to have only one object.

In AngularJs, we can use the dependency injection and the services to enable the singleton pattern.

Technically, if we call the **“new Class()”** two times without following the singleton pattern, the outcome will be two objects of the same class.

Whenever the singleton pattern is enabled, then the class will create the object first time and return its reference in the next call.

What Are The Essential Characteristics Of The Angular Scope Object?

A scope is an application object which behaves as the owner of the apps variables and functions. It has access to both the View and controller. Thus it works as a medium of communication between both of them. This object contains both data and functions. We can use it to access model data of the controller.

Following are the essential characteristics of the scope object.

* It provides observers to watch for all the model changes.
* Provides the ability to propagate model changes through the application as well as outside the system to other associated components.
* Scopes allow nesting in such a way that they can isolate functionality and model properties.
* Provides an execution environment in which expressions get evaluated.

What Is “$RootScope” In AngularJS?

Every AngularJS application has a **“$rootScope**” that is the top-most scope created on the DOM element.

An app can have only one **$rootScope** which shares among all its components. It contains the ng-app directive. Every other scope is its child scope. It can watch expressions and propagate events.

Using the root scope, we can set the value in one controller and read it from the other controller.

What Is Scope Hierarchy In Angular? How Many Scopes Can An Application Have?

Every AngularJS application consists of one root scope but may have several child scopes.

As child controllers and directives create new child scopes, they get attached to the application. These new scopes get added as children of their parent scope.

Similar to DOM, they also create a hierarchical structure.

What Are Single Page Applications (SPA) In AngularJS?

Single-Page Applications (SPAs) are web applications that fit on a single HTML page. It dynamically updates the web page as the user performs actions on the app.

SPAs use AJAX and HTML to create quick and responsive web apps. A single page load extracts all the web app code (JS, HTML, CSS).

Thus the user navigates to different parts of the application quickly as it happens without refreshing the whole page.

What Are The Benefits Does SPA Provide?

The main characteristics of Single-Page Applications are as follows.

* Its UI is fast and responsive. Also, the Back/Forward buttons present in the UI work properly.
* IT contains more JavaScript code than actual HTML as compared to other applications.
* Dynamic data loading occurs from the server-side. The API uses restful web service with JSON format.
* It allows to pre-load and cache all the app pages. Thus fewer data download requests are made towards the server.
* Applications written in AngularJS are cross-browser compliant. It automatically handles the JavaScript code suitable for each browser.
* Even if the user has lost the internet connection, then also the SPA can work as all the pages load in the starting itself.

What Is The Difference Between $Scope And Scope?

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

Factory methods like controller, filter, service, etc. receive its arguments via dependency injection (DI). In DI, the order of passing the arguments does not matter.

For example, a controller may have the following definition.

(let’s set $scope as the first parameter in this case)

myApp.controller('MyController', ['$scope', function($scope, $http) {

//rest of the code goes here }

OR ( if the $scope is the second parameter)

myApp.controller('MyController', ['$scope', function($http, $scope) {

//rest of the code goes here }

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

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

app.directive("myDirective", function() {

return {

scope: {};

link: function(scope, element, attrs) {

// code goes here.

}

};

});

In the case of non-dependency injected arguments, we can also give them a name as seen in the below example:

app.directive("myDirective", function() {

return {

scope: {};

link: function(foo, bar, biz) {

// code goes here.

}

};

});

To summarize, in DI case, we pass the <scope object> as <$scope> whereas, in non-DI cases, the <scope object> is returned either as a scope or with any name.

How Does The Compilation Process Happen In Angular JS?

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

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

The compilation occurs in two phases.

* **Compile –** The service first traverses the entire DOM, retrieves all the directives, and generates a linking function.
* **Link –** It supplements the directives with a scope and generates a live view. Changing either the “Scope” or “View” will have a reciprocal effect.

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

How Is AngularJS Compilation Different From Other JavaScript Frameworks?

The popular Javascript frameworks like the Backbone and jQuery parse the HTML template as a stream and returns the result as a string.

They dump the resulting string into the DOM where we can retrieve it using the **innerHTML()** method.

On the contrary, AngularJS processes the template differently. It directly works on HTML DOM rather than strings and manipulates it as required. It uses two-way data binding between the model and view to sync the data.

What Is The Use Of Ng-View In Angular?

The ng-view tag creates a placeholder where an HTML or ng-template view can be placed based on the configuration.

Give An Example Of Ng-View In Angular?

Let’s take an example of the ng-view.

<div ng-app = "testApp">

<div ng-view>

<!-- Target Html Template here -->

</div>

</div>

In other words, ng-view is the directive that works as a container for angularJS to switch between views.

What Is The Purpose Of Ng-Template In Angular?

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

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

Give An Example Of Ng-Template In Angular?

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

**Example-**

<div ng-app = "mainApp">

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

<h2> Add Employee </h2>

{{message}}

</scrip t>

</div>

**$routeProvider part.**

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

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

$routeProvider.

when('/addEmployee', {

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

}).

otherwise({

redirectTo: '/addEmployee'

});

}]);

What Is $RouteProvider In Angular?

The $routeProvider is a primary AngularJS service which sets the configuration of URLs, map them to the corresponding HTML page or ng-template, and attach a controller with the same.

Give An Example Of The $RouteProvider In Angular?

Let’s see the following example:

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

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

$routeProvider.

when('/addEmployee', {

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

}).

otherwise({

redirectTo: '/addEmployee'

});

}]);

Following are some important points to consider for the above example.

* The routeProvider acts as a function under the config of the mainApp module using a key as ‘$routeProvider.’
* $routeProvider.when defines a URL “/addEmployee” which is then mapped to “addEmployee.htm”. The “addEmployee.htm” should be present on the same path as the main HTML page.
* The “otherwise” clause sets the default view.
* The “controller” clause sets the corresponding controller for the view.

What Does Data Binding Mean In AngularJS?

Data binding is the connection bridge between view and business logic (view model) of the application. In AngularJS, it does the automatic synchronization between the model and view.

When the model changes, the view reflects it automatically and vice versa. AngularJS supports the following two types of bindings:

* One-way and,
* Two-way

What Are The Data Binding Directives Does Angular Support?

AngularJS provides the following data binding directives:

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

What Is Ng-Bind Directive In Angular?

It updates the text content of the specified HTML element with the value of the given expression.

The text content gets updated when there is any change in the expression value. It is very similar to the double curly markup ( {{expr }}) but is less verbose.

It has the following syntax.

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

What Is Ng-Bind-Html Directive In Angular?

It evaluates the expression and inserts the HTML content into the element in a secure way. To use this functionality, it has to use $sanitize service. For this, it is mandatory that $sanitize is available.

It has the following Syntax.

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

What Is Ng-Bind-Template Directive In Angular?

It replaces the element text content with the interpolation of the template. It can contain multiple double curly markups.

It has the following syntax.

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

What Is Ng-Non-Bindable Directive In Angular?

This directive informs AngularJS, not to compile or bind the contents of the current DOM element. It is useful in the case when the user wants to display the expression only and do not want to execute it.

It has the following syntax.

<ANY ELEMENT ng-non-bindable > </ANY ELEMENT>

What Is The Ng-Model Directive In Angular?

This directive is capable of binding with the input, select, text area or any custom form control. It provides two-way data binding. It also provides validation behavior. It also retains the state of the HTML elements (like valid/invalid, touched/untouched and so on).

It has the following syntax.

<input ng-model="name"/>

What Directives Are Used To Show And Hide HTML Elements In AngularJS?

The directives used to show and hide HTML elements in the AngularJS are <ng-show> and <ng-hide>. They do this based on the result of an expression.

Its syntax is as follows.

<element ng-show="expression"></element>

When the expression for <ng-show> evaluates to true, then HTML element(s) are shown on the page, otherwise the HTML element is hidden. Similarly, <ng-hide> directive hides the HTML element if the expression evaluates to true.

Let’s take the following example.

<div ng-controller="MyCtrl">

<div ng-show="data.isShow">ng-show Visible</div>

<div ng-hide="data.isHide">ng-hide Invisible</div>

</div>

<script>

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

app.controller("MyCtrl", function ($scope) {

$scope.data = {};

$scope.data.isShow = true;

$scope.data.isHide = true;

});

</script>

What Is Ng-If Directive In AngularJS?

This directive can add/remove HTML elements from the DOM based on the input expression.

If the expression evaluates to true, it adds a copy of HTML elements to the DOM. If the expression evaluates to false, this directive removes the HTML element from the DOM.

<div ng-controller="MyCtrl">

<div ng-if="data.isVisible">ng-if Visible</div>

</div>

<script>

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

app.controller("MyCtrl", function ($scope) {

$scope.data = {};

$scope.data.isVisible = true;

});

</script>

What Is Ng-Switch Directive In AngularJS?

This directive can add/remove HTML elements from the DOM conditionally based on scope expression.

Child elements with the **“ng-switch-when”** directive will be displayed if it gets a match, else the element and its children get removed. It also allows defining a default section, by using the **“ng-switch-default”**directive. It falls back to the default when no other section matches.

Let’s see the following example that displays the syntax for **“ng-switch.**”

<div ng-controller="MyCtrl">

<div ng-switch on="data.case">

<div ng-switch-when="1">Shown when case is 1</div>

<div ng-switch-when="2">Shown when case is 2</div>

<div ng-switch-default>Shown when case is anything else than 1 and 2</div>

</div>

</div>

<script>

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

app.controller("MyCtrl", function ($scope) {

$scope.data = {};

$scope.data.case = true;

});

</script>

What Does The Ng-Repeat Directive Do In AngularJS?

This directive is used to iterate over a collection of items and generate HTML from it.

<div ng-controller="MyCtrl">

<ul>

<li ng-repeat="name in names">

{{ name }}

</li>

</ul>

</div>

<script>

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

app.controller("MyCtrl", function ($scope) {

$scope.names = [ 'Mahesh', 'Raj', 'Diksha' ];

});

</script>

What Are Different Variables Used With The Ng-Repeat Directive?

The **“ng-repeat”** directive has a set of unique variables that are useful while iterating the collection.

These variables are as follows.

* $index
* $first
* $middle
* $last

The “$index” contains the index of the element used for the traversal. The variables $first, $middle and $last returns a boolean value depending on whether the current item is the first, middle or last element in the collection.

Can You Demonstrate The Use Of Ng-Repeat Variables?

Below is the Angular code to show the usage of ng-repeat variables.

<html>

<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<head>

<script>

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

app.controller("ctrl", function ($scope)

{

$scope.employees = [

{

name: 'A',

gender: 'alphabet'

},

{

name: 'B',

gender: 'number'

},

{

name: 'C',

gender: 'alphanumeric'

},

{

name: 'D',

gender: 'special character'

}];

});

</script>

</head>

<body ng-app="app">

<div ng-controller="ctrl">

<ul>

<li ng-repeat="employee in employees">

<div> {{employee.name}} is a {{employee.gender}}. <span ng-if="$first">

<strong>(first element found)</strong>

</span> <span ng-if="$middle">

<strong>(middle element found)</strong>

</span> <span ng-if="$last">

<strong>(last element found)</strong>

</span>

</div>

</li>

</ul>

</div>

</body>

</html>

The output is as follows.

A is a alphabet. (first element found)

B is a number. (middle element found)

C is a alphanumeric. (middle element found)

D is a special character. (last element found)

What Do You Know About The Factory Method In AngularJS?

A factory is a simple function which allows you to add some logic before creating the object.

In the end, it returns the packed object. The factory method has the following signature.

**Syntax**

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

How Do You Create A Service In Angular Using The Factory Method?

See the below Angular code to create a service.

<script>

//creating module

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

//define a factory using factory() function

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

var serviceObj = {};

serviceObj.function1 = function () {

//TO DO:

};

serviceObj.function2 = function () {

//TO DO:

};

return serviceObj;

});

</script>

Where Can You Use The Factory Method In Angular?

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.

How Does String Interpolation Take Place In AngularJS?

During the compilation process, AngularJS compiler matches text and attributes using interpolate service to see if it contains embedded expressions.

During normal, digest life cycle, these expressions are updated and registered as watches.

Define AngularJS Application Lifecycle?

Understanding the life cycle of an AngularJS application makes it easier to learn about the way to design and implement the code.

The Apps life cycle consists of the following three phases-

* Bootstrap,
* Compilation, and
* Run-time

Describe The Different Stages Of The AngularJS Application Lifecycle?

These three phases of the life cycle occur each time a web page of an AngularJS application gets loaded into the browser. Let’s learn about each of the three stages in detail:

* **The Bootstrap Phase –** In this phase, the browser downloads the AngularJS javascript library. After this, AngularJS initializes its necessary components and the modules to which the ng-app directive points. Now that the module has loaded, required dependencies are injected into it and become available to the code within that module.
* **The Compilation Phase –** The second phase of the AngularJS life cycle is the HTML compilation stage. Initially, when a web page loads in the browser, a static form of the DOM gets loaded. During the compilation phase, this static DOM gets replaced with a dynamic DOM which represents the app view. There are two main steps – first, is traversing the static DOM and collecting all the directives. These directives map to the appropriate JavaScript functionality which lies either in the AngularJS built-in library or custom directive code. After adding the scope, it generates a live view.
* **The Runtime Data Binding Phase –** This is the final phase of the AngularJS application. It remains until the user reloads or navigates to a different web page. At this point, any changes in the scope get reflected in the view, and vice-versa. It makes Scope as the single source of data for the View.

The above points indicate that AngularJS behaves differently from traditional methods of binding data. The conventional approaches combine a template with data, received from the engine and then manipulate the DOM each time there is any change in the data.

However, AngularJS compiles the DOM only once and then links the compiled template as necessary, making it much more efficient than the traditional methods.

Define AngularJS Scope Lifecycle?

After the angular app gets loaded into the browser, scope data passes through different stages called its life cycle. Learning about this cycle helps us to understand the interaction between scope and other AngularJS components.

What Are The Different Phases Of The AngularJS Scope Lifecycle?

The scope data traverses through the following phases.

* **Creation –** This phase initializes the scope. During the bootstrap process, the $injector creates the root scope of the application. And during template linking, some directives create new child scopes. A digest loop also gets created in this phase that interacts with the browser event loop. This loop is responsible for updating DOM elements with the changes made to the model as well as executing any registered watcher functions.
* **Watcher registration –** This phase registers watchers (by using the $watch() function) for the scope created in the above point. These watches propagate the model changes to the DOM elements, automatically.
* **Model mutation –** This phase occurs when there is any change in the scope data. When we do any modification in the angular app code, the scope function <$apply()> updates the model and then calls the <$digest()> function to update the DOM elements and the registered watches. However, when we change the scope of the angular code like within controllers or services, angular internally calls <$apply()> function for us. But, when we do the changes to the scope outside the Angular code, we have to call the <$apply()> function explicitly, to force the model and DOM to be updated correctly.
* **Mutation observation –** This phase occurs, when the digest loop execute the $digest() function at the end of $apply() call. When the $digest() function executes, it evaluates all watches for model changes. If there is a change in the value, $digest() calls the $watch listener and updates the DOM elements.
* **Scope destruction –** This phase occurs when the child scopes that are no longer needed, are removed from the browser’s memory by using the $destroy() function. It is the responsibility of the child scope creator to destroy them via scope.$destroy() API. It stops propagation of $digest calls into the child scopes and enables the browsers’ garbage collector to reclaim the unused memory.

What Is An Auto Bootstrap Process In AngularJS?

AngularJS initializes automatically upon the **“DOMContentLoaded**” event or when the browser downloads the angular.js script and at the same time **document.readyState** is set to ‘complete.’ At this point, AngularJS looks for the ng-app directive which is the root of the Angular app compilation process.

After locating the ng-app directive, AngularJS does the following tasks.

* Load the module associated with the directive.
* Create the application injector.
* Compile the DOM starting from the ng-app root element.

We term this process as **Auto-bootstrapping**.

Give An Example Of The Auto Bootstrap Process In Angular?

Following is the sample code that helps to understand it more clearly:

**Example-**

<html>

<body ng-app="myApp">

<div ng-controller="Ctrl">Hello {{msg}}!</div>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.5/angular.min.js"></script>

<script>

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

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

$scope.msg = 'Welcome';

});

</script>

</body>

</html>

What Is The Manual Bootstrap Process In AngularJS?

Sometimes we may need to manually initialize the Angular app to have more control over the initialization process. We can do that by using **angular.bootstrap()** function within **angular.element(document).ready()**function. AngularJS fires this function when the DOM is ready for manipulation.

The **angular.bootstrap()** function takes two parameters, the document, and module name injector.

Give An Example Of The Manual Bootstrap Process In Angular?

Following is the sample code that helps to understand the concept more clearly.

<html>

<body>

<div ng-controller="Ctrl">Hello {{msg}}!</div>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.5/angular.min.js"></script>

<script>

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

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

$scope.msg = 'Welcome';

});

//manual bootstrap process

angular.element(document).ready(function () { angular.bootstrap(document, ['myApp']); });

</script>

</body>

</html>

How Does Automatic Bootstrap Of Multiple Modules Happen In Angular?

AngularJS gets initialized for the first module by default. If there are many modules, then we combine them into one, and the angular app automatically initializes it. Other modules act as dependencies for this newly created module.

Let’s take an example, suppose we have two modules: module1 and model2. To initialize the app automatically, based on these two modules following code is used:

<html>

<head>

<title>Multiple modules bootstrap</title>

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

<script>

//module1

var app1 = angular.module("module1", []);

app1.controller("Controller1", function ($scope) {

$scope.name = "Welcome";

});

//module2

var app2 = angular.module("module2", []);

app2.controller("Controller2", function ($scope) {

$scope.name = "World";

});

//module3 dependent on module1 & module2

angular.module("app", ["module1", "module2"]);

</script>

</head>

<body>

<!--angularjs autobootstap process-->

<div ng-app="app">

<h1>Multiple modules bootstrap</h1>

<div ng-controller="Controller2">

</div>

<div ng-controller="Controller1">

</div>

</div>

</body>

</html>

How Does Manual Bootstrap Of Multiple Modules Happen In Angular?

We can manually bootstrap the app by using angular.bootstrap() function for multiple modules.

Let’s modify the example in the previous question for a manual bootstrap process.

<html>

<head>

<title>Multiple modules bootstrap</title>

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

<script>

//module1

var app1 = angular.module("module1", []);

app1.controller("Controller1", function ($scope) {

$scope.name = "Welcome";

});

//module2

var app2 = angular.module("module2", []);

app2.controller("Controller2", function ($scope) {

$scope.name = "World";

});

//manual bootstrap process

angular.element(document).ready(function () {

var div1 = document.getElementById('div1');

var div2 = document.getElementById('div2');

//bootstrap div1 for module1 and module2

angular.bootstrap(div1, ['module1', 'module2']);

//bootstrap div2 only for module1

angular.bootstrap(div2, ['module1']);

});

What Is Compile-Time Linking In Angular?

It collects an HTML string or DOM into a template and produces a template function.

It can then be used to link the scope and the template together.

AngularJS uses the compile function to change the original DOM before creating its instance and before the creation of scope.

Let’s see the Link function in detail.

It has the duty of linking the model to the available templates. AngularJS does the data binding to the compiled templates using Link.

Following is the Link syntax.

link: function LinkFn(scope, element, attr, ctrl){}

Where each of the four parameters is as follows-

* **Scope –** It is the scope of the directive.
* **Element –** It is the DOM element to go for processing.
* **Attr-** It is the collection of attributes of the DOM element.
* **Ctrl –**It is the array of controllers required by the directive.

AngularJS allows setting the link property to an object also. The advantage of having an object is that we can split the link function into two separate methods called, pre-link and post-link.

When Does The Pre And Post Linking Happen In AngularJS?

Execution of Post-Link function starts after the linking of child elements. It is safer to do DOM transformation during the link time. The post-link function is suitable to execute the logic.

It gets executed before the child elements are linked. It is not safe to do DOM transformation as the compiler linking function will fail to locate the correct fields.

It is good to use the pre-link function to implement the logic that runs when AngularJS has already compiled the child elements. Also, before any of the child element’s post-link functions have been called.

Give An Example Of The Compile, Pre-Link, And Post-Link Functions?

Let’s see an example that talks about Compile, Pre-Link, and Post-Link functions.

<html>

<head>

<title>Compile vs Link</title>

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

<script type="text/javascript">

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

function createDirective(name){

return function(){

return {

restrict: 'E',

compile: function(tElem, tAttrs){

console.log(name + ': compile');

return {

pre: function(scope, iElem, iAttrs){

console.log(name + ': pre link');

},

post: function(scope, iElem, iAttrs){

console.log(name + ': post link');

}

}

}

}

}

}

app.directive('levelOne', createDirective('levelOne'));

app.directive('levelTwo', createDirective('levelTwo'));

app.directive('levelThree', createDirective('levelThree'));

</script>

</head>

<body ng-app="app">

<level-one>

<level-two>

<level-three>

Hello {{name}}

</level-three>

</level-two>

</level-one>

</body>

</html>

Output:

Hello

What Are The Rules Does A Controller Enforce In AngularJS?

A Controller is a set of JavaScript functions which is bound to a specified scope, the ng-controller directive.

Angular creates a new instance of the Controller object to inject the new scope as a dependency. The role of the Controller is to expose data to our view via $scope and add functions to it, which contains business logic to enhance view behavior.

**Controller Rules**

* A Controller helps in setting up the initial state of the scope object and define its behavior.
* The Controller should not be used to manipulate the DOM as it contains only business logic. Instead, for managing the DOM, we should use data binding and directives.
* Do not use Controllers to format input. Instead, using angular form controls is recommended for that.
* Controllers should not be used to share code or states. Instead, use angular services for it.

What Are The Steps To Create A Controller In AngularJS?

* It needs ng-controller directive.
* Next step is to add Controller code to a module.
* Name your Controller based on functionality. Its name should follow the camel case format (i.e., SampleController).
* Set up the initial state of the scope object.

**Example-**

Declaring a Controller using ng-Controller directive-

<div ng-app="mainApp" ng-controller="SampleController">

</div>

Following code displays the definition of SampleController.

<script>

function SampleController($scope) {

$scope.sample = {

firstSample: "INITIAL",

lastSample: "Initial",

fullName: function() {

var sampleObject;

sampleObject = $scope.sample;

return sampleObject.firstSample + " " + sampleObject.lastSample;

}

};

}

</script>

What Does Service Mean In Angular?

**Answer.**

Services are functions that are bound to perform specific tasks in an application.

* It gives us a method that helps in maintaining the angular app data for its lifetime.
* It gives us methods that facilitate to transfer data across the controllers in a consistent way.
* It is a singleton object, and its instance is created only once per application.
* It is used to organize and share, data and function across the application.

Two main execution characteristics of angular services are that they are Singleton and lazy instantiated.

What Does Lazy Instantiation Mean In Angular?

It means that AngularJS instantiates a service only when a component of an application needs it. The dependency injection method gets used here which makes the Angular codes, robust and less error-prone.

What Are Singletons In Angular?

Each application component dependent on the service, work with the single instance of the service created by the AngularJS.

Let us take an example of a straightforward service that calculates the square of a given number:

var CalculationService = angular.module('CalculationService', [])

.service('Calculation', function () {

this.square = function (a) { return a\*a};

});

What Are The Built-In Services Provided By Angular?

AngularJS provides many built-in services. Each of them is responsible for a specific task. They always have prefixed with the $ symbol.

Some of the commonly used services in any AngularJS application are as follows:

* **$http –** used to make an Ajax call to get the server data.
* **$window –** Provides a reference to a DOM object.
* **$Location –** Provides reference to the browser location.
* **$timeout –** Provides a reference to the window.set timeout function.
* **$Log –** used for logging.
* **$sanitize –** Used to avoid script injections and display raw HTML in the 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.
* **$q –** Provides a promise object.
* **$cookies –** Use this service for reading, writing and deleting the browser cookies.
* **$parse –** This service intends to convert an AngularJS expression into a function.
* **$cacheFactory –** This service evaluates the specified expression when the user changes the input.

What Are Different Ways To Create A Service In AngularJS?

There are five different ways to create services in AngularJS.

* Value
* Factory
* Service
* Provider
* Constant

How To Create A Service Using Angular Value?

It is the simplest service type supported by AngularJS that we can create and use. It is similar to a key-value pair or like a variable having a value. It can store only a single value. Let’s take an example and create a service that displays the username:

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

app.value("username","Madhav");

**Code to use “Value”:**

We can use this service anywhere by using dependency injection. Following example injects the service in a controller:

app.controller("MainController",function($scope, username){

$scope.username=username;

});

In the above example, we have created a Value service “username” and used it in MainController.

How To Create A Service Using Angular Factory?

Value service is simple to write, but they may lack many essential features. So, the next service type we will look at is “Factory” service. After its creation, we can even inject other services into it. Unlike Value service, we cannot add any dependency on it.

Let’s take an example to create the Factory service.

app.factory("username",function(){

var name="John";

return {

name:name

}

});

The above code shows that the Factory service takes “function” as an argument. We can inject any number of dependencies or methods in this “function” as required by this service. This function must return some object. In our example, it returns an object with the property name. Now, let us look, as to how we can use this service:

**Code to use “Factory”:**

The function returns an object from service which has a property name so we can access it and use it anywhere. Let’s see how we can use it in the controller:

app.controller("MainController",function($scope, username){

$scope.username=username.name;

});

We are assigning the username from factory service to our scope username.

How To Create A Service From The Angular Service?

It works the same as the “Factory” service. But, instead of a function, it receives a Javascript class or a constructor function as an argument. Let’s take an example. Suppose we have a function:

function MyExample(num){

this.variable="value";

}

Now, we want to convert the function into a service. Let’s take a look at how we can do this with “Factory” method:

app.factory("MyExampleService",["num" ,function(num){

return new MyExample(num);

}]);

Thus in this way, we will create its new instance and return it. Also, we have injected <num> as a dependency on Factory service. Now, let’s see how we can do this using Service type:

app.service("MyExampleService",["num", MyExample]);

Thus, we have called the service method on the module and provided its name, dependency, and the name of the function in an array.

How To Create A Service Using Angular Provider?

It is the parent of all the service types supported by AngularJS, except the “Constant” that we will discuss in the next section. It is the core of all the service types. Thus we can say that other services work on top of it. It allows us to create a configurable service that must implement the <$get> method.

We use this service to expose the API that is responsible for doing the application-wide configuration. The configuration should complete before starting the application.

Let’s take an example.

**Example-**

app.provider('authentication', function() {

var username = "John";

return {

set: function(newUserName) {

username = newUserName;

},

$get: function() {

function getUserName() {

return username;

}

return {

getUserName: getUserName

};

}

};

});

This example initializes a provider with its name as “authentication.” It also implements a <$get> function, which returns a method “getUsername” which in turn returns the private variable called username. It also has a setter, using it we can set the username on application startup as follows:

app.config(["authenticationProvider", function(authenticationProvider) {

authenticationProvider.set("Mihir");

}]);

How To Create A Service Using Angular Constant?

As the name suggests, this service helps us to declare constants in our application. We can then use them wherever needed, just by adding it as a dependency. There are many places, where we use constants like some base URLs, application name, etc.

We define them once and use them anywhere as per our need. Thus, this technique allows us to write the definition in one place. If there is any change in the value later, we have to do the modifications at one location only.

Here is an example of how we can create constants:

app.constant('applicationName', 'Service Tutorials');

What Is The Use Of $Watch() In Angular?

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

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

Following is the example of using the **$watch()** function in AngularJS applications.

**Example-**

<html>

<head>

<title>AngularJS Watch</title>

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

<script>

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

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

($scope) {

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

$scope.counter = 0;

//watching change in name value

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

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

});

});

</script>

</head>

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

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

<br /><br />

Counter: {{counter}}

</body>

</html>

What Is The Use Of $Digest() In Angular?

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

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

Following is the example of using the **$digest()** function in AngularJS applications:

**Example-**

<html>

<head>

<title>AngularJS Digest Example</title>

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

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

</head>

<body ng-app="app">

<div ng-controller="Ctrl">

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

<br />

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

</div>

<script>

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

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

$scope.obj = { value: 1 };

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

console.log("digest clicked!");

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

//update value

$scope.$digest();

});

});

</script>

</body>

</html>

What Is The Use Of $Apply() In Angular?

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

Following is the example of using the **$apply()** function in AngularJS applications.

**Example-**

<html>

<head>

<title>AngularJS Apply Example</title>

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

<script>

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

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

($scope) {

$scope.datetime = new Date();

$scope.updateTime = function () {

$scope.datetime = new Date();

}

//outside angular context

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

//update the value

$scope.$apply(function () {

console.log("update time clicked");

$scope.datetime = new Date();

console.log($scope.datetime);

});

});

});

</script>

</head>

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

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

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

<br />

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

</body>

</html>

What Is The Main Difference Between $Apply() And Digest()?

Following is the key difference between **$apply()** and **$digest()**.

* The $**apply()** method performs an update to the model properties forcibly.
* The **$digest()** method evaluates the watchers for the current scope. However, the **$apply()** method does the same for the root scope.

Which One Handles Exception Automatically Between $Digest And $Apply?

When an error occurs in one of the watchers, **$digest()** cannot handle them via **$exceptionHandler**service. In that case, you have to manage the exception yourself. However, **$apply()** uses try catch block internally to handle errors. But, if an error occurs in one of the watchers, then it transfers the errors to $exceptionHandler service.

Code for **$apply()** function.

$apply(expr) {

try {

return $eval(expr);

} catch (e) {

$exceptionHandler(e);

} finally {

$root.$digest();

}

}

What Is The Use Of $Watchgroup() In Angular?

This function $watchgroup() first came in **Angular 1.3**. It works in the same way as **$watch()** function except that the first parameter is an array of expressions.

$watchGroup(watchExpression, listener)

The listener is also an array containing the new and old values of the variables. The listener gets called whenever any expression contained in the **watchExpressions** array changes.

$scope.teamScore = 0;

$scope.time = 0;

$scope.$watchGroup(['teamScore', 'time'], function(newVal, oldVal) {

if(newVal[0] > 20){

$scope.matchStatus = 'win';

}

else if (newVal[1] > 60){

$scope.matchStatus = 'times up';

});

What Is The Use Of $WatchCollection() In Angular?

The use of this function is to watch the properties of an object. It gets fired when there is any change in their values.

It takes an object as the first parameter and monitors the properties of the object.

$watchCollection(obj, listener)

The listener gets called whenever there is any change in the obj.

$scope.names = ['shailendra', 'deepak', 'mohit', 'kapil'];

$scope.dataCount = 4;

$scope.$watchCollection('names', function (newVal, oldVal) {

$scope.dataCount = newVal.length;

});

How To Perform Mandatory Input Field Validation In Angular?

AngularJS allows form validation on the client-side in a simplistic way. First of all, it monitors the state of the form and its input fields. Secondly, it observes any change in the values and notifies the same to the user.

By using “Required Field” validation we can prevent, form submission with a null value. It’s mandatory for the user to fill the form fields.

The syntax for required field validation is as follows.

<input type="text" required />

Example Code.

<form name="myForm">

<input name="myInput" ng-model="myInput" required>

</form>

<p>The input's valid state is:</p>

<h1>{{myForm.myInput.$valid}}</h1>

How To Perform Minimum & Maximum Field Length Validations In Angular?

To prevent the user from providing less or excess number of characters in the input field, we use Minimum & Maximum length validation. The AngularJS directive used for Minimum & Maximum length validations are <ng-minlength> and <ng-maxlength>. Both of these attributes take integer values. The <ng-minlength> attribute is used to set the number of characters a user is limited to, whereas the <ng-maxlength> attribute sets the maximum number of characters that a user is allowed to enter.

This type of validation requires the following syntax.

<input type="text" ng-minlength=5 />

<input type="text" ng-maxlength=10 />

Example code:

<label>User Message:</label>

<textarea type="text" name="userMessage" ng-model="message"

ng-minlength="100" ng-maxlength="1000" required>

</textarea>

<div ng-messages="exampleForm.userMessage.$error">

<div ng-message="required">This field is required</div>

<div ng-message="minlength">Message must be over 100 characters</div>

<div ng-message="maxlength">Message must not exceed 1000 characters</div>

</div>

How To Perform Pattern Validation In Angular?

AngularJS provides a <ng-pattern> directive to ensure that input fields match the regular expressions in the attributes.

To handle pattern validation, we can use the following syntax.

<input type="text" ng-pattern="[a-zA-Z]" />

To activate the error message in <ng-pattern>, we pass the value of pattern into ng-message.

Example code.

<label>Phone Number:</label>

<input type="email" name="userPhoneNumber" ng-model="phoneNumber"

ng-pattern="/^[\+]?[(]?[0-9]{3}[)]?[-\s\.]?[0-9]{3}[-\s\.]?[0-9]{4,6}$/"

required/>

<div ng-messages="exampleForm.userPhoneNumber.$error">

<div ng-message="required">This field is required</div>

<div ng-message="pattern">Must be a valid 10 digit phone number</div>

</div>

How To Perform Email Validation In Angular?

To validate an email id, AngularJS provides the ng-model directive. Using the following syntax, we can verify the email from an input field.

<input type="email" name="email" ng-model="user.email" />

Example code.

<label>Email Address:</label>

<input type="email" name="userEmail" ng-model="email" required />

<div ng-messages="exampleForm.userEmail.$error">

<div ng-message="required">This field is required</div>

<div ng-message="email">Your email address is invalid</div>

</div>

How To Perform Number Validation In Angular?

To validate input against Number we can use ng-model directive from AngularJS.

Its syntax is as follows.

<input type="number" name="personage" ng-model="user.age" />

How To Perform URL Validation In Angular?

To validate an input field for URL, we can use the following signature.

<input type="url" name="weblink" ng-model="user.facebook\_url" />

Example Code.

<div ng-app="urlInputExample">

<form name="myForm" ng-controller="UrlController">

<label for="exampleInput">Enter Email</label>

<input type="url" name="input" ng-model="example.url" required />

<p style="font-family:Arial;color:red;background:steelblue;padding:3px;width:350px;"

ng-if='!myForm.input.$valid'>Enter Valid URL</p>

</form>

</div>

How Do You Exchange Data Among Different Modules Of Your Angular JS Application?

There are a no. of ways in Angular to share data among modules. A few of them are as follows.

* The most common method is to create an Angular service to hold the data and dispatch it to the modules.
* Angular has a matured event system which provides $broadcast(), $emit() and $on() methods to raise events and pass data among the controllers.
* We can also use $parent, $nextSibling, and $ controllerAs to directly access the controllers.
* Variables defined at the root scope level ($rootScope) are available to the controller scope via prototypical inheritance. But they behave like globals and hard to maintain.

How Would You Use An Angular Service To Pass Data Between Controllers? Explain With Examples?

Using services is the best practice in Angular to share data between controllers. Here is a step by step example to demonstrate data transfer.

We can prepare the data service provider in the following manner.

**Example-**

app.service('dataService', function() {

var dataSet = [];

var addData = function(newData) {

dataSet.push(newData);

};

var getData = function(){

return dataSet;

};

return {

addData: addData,

getData: getData

};

});

Now, we’ll inject the service dependency into the controllers.

Say, we have two controllers – pushController and popController.

The first one will add data by using the data service provider’s addData method. And the latter will fetch this data using the service provider’s getData method.

**Example-**

app.controller('pushController', function($scope, dataService) {

$scope.callToAddToProductList = function(currObj){

dataService.addData(currObj);

};

});

app.controller('popController', function($scope, dataService) {

$scope.dataSet = dataService.getData();

});

How Will You Send And Receive Data Using The Angular Event System?

We can call the $broadcast method using the $rootScope object and send any data we want.

$scope.sendData = function() {

$rootScope.$broadcast('send-data-event', data);

}

To receive data, we can use the $scope object inside a controller.

$scope.$on('send-data-event', function(event, data) {

// process the data.

});

How Do You Switch To Different Views From A Controller Function?

With the help of **<ui-sref>** directive or using the **$state.go()** function, we can switch between different views from the controllers.

Give A Working Example Of Switching Views In Angular?

Here is a working example to bring you more clarity on switching views in Angular.

**AngularJS Code**

var myapp = angular.module('myapp', ['ui.router']);

//Setup a basic AngularJS app.

myapp.config(['$stateProvider', function ($stateProvider) {

$stateProvider.state('python', {

url: '/python',

template: '<div style="background: blue">{{pythonCtrl.title}}</div>',

controller: 'TestFirstCtrl',

controllerAs: 'pythonCtrl'

});

$stateProvider.state('java', {

url: '/java',

template: '<div style="background: green">{{javaCtrl.title}}</div>',

controller: 'TestSecondCtrl',

controllerAs: 'javaCtrl'

});

}]);

//These two controllers will control the same view with different routes.

myapp.controller('TestFirstCtrl', ['$scope', function($scope) {

this.title = 'Learn Python';

}]);

myapp.controller('TestSecondCtrl', ['$scope', function($scope) {

this.title = 'Learn Java';

}]);

**HTML Page**

It is the HTML code to test the Angular functionality given above.

<html>

<head>

<script src="//ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

<script src="//angular-ui.github.io/ui-router/release/angular-ui-router.js"></script>

</head>

<body ng-app="myapp">

<input type="button" ui-sref="python">Go to Learn Python</button>

<input type="button" ui-sref="java">Go to Learn Java</button>

<div ui-view></div>

</body>

</html>

What Would You Do To Limit A Scope Variable To Have One-Time Binding?

We can prefix the “**::”** operator to the scope variable.

It’ll make sure the candidate is aware of the available variable bindings in AngularJS.

What Is The Difference Between One-Way Binding And Two-Way Binding?

The main difference between one-way binding and two-way binding is as follows.

* In one-way binding, the scope variable in the HTML gets initialized with the first value its model specifies.
* In two-way binding, the scope variable will change its value whenever the model gets a different value.

Which Angular Directive Would You Use To Hide An Element From The DOM Without Modifying Its Style?

It is the conditional **ngIf Directive** which we can apply to an element. Whenever the condition becomes false, the **ngIf Directive** removes it from the DOM.

What Are Some Main Differences Between Angular V1.0 And Angular V2.0?

Angular v2.0 is not a mere upgrade of Angular v1.0. Instead, it has got a new implementation altogether.

**Angular v1.0**

**Angular v2.0**

No mobile support

Mobile-oriented

The core concept was $scope

No $scope

Use of controllers

Components replaced controllers

ng-repeat

\*ngFor

No provision to define local variables

Provides hash(#) prefix for local variables

ng-model

[(ngModel)]

ES5, ES6, and Dart only

ES5, ES6, TypeScript or Dart

Use of lowercase (ng-class)

Use of camelCase (ngClass)

Hope the above table could be useful.

How Is Angular V6.0 Different From Its Past Versions?

Angular v6.0 has got some of the modern features, and it is much easier to use and has a quick turnaround time. Check out from the below points.

* Improved support of decorators and error messages
* More validators for array functions
* Native element compatibility
* Closure compiler to produce small bundles
* Brought-in Typescript 2.7 + support
* HttpInterceprtor support to inject into HttpClient
* Projected components too can have tests
* Generic types for ElementRef
* Provision for canonical view queries
* Directive def to have type and hooks
* Source navigation improvements

Q1) What is Means by AngularJS 2/4?

Answer: Angular is a TypeScript-based open-source front-end web form framework led by the Angular Team at Google and by an identity of individuals and corporations. Angular is a thoroughgoing rewrite from the same team that built AngularJS.

Q2) Are there breaking changes between angular 2 & angular 4?

Answer: Well, there is no path-breaking change that means Angular 4 is not an absolute rewrite of Angular 2. Google has adapted SEMVER  approach for it’s Angular Framework. Due to misalignment of the router can exception. As @angular/router obtained using v3.3 already, so they switch to Angular 4.

Q3) What are the new features of Angular 4?

Answer:

* Revamped \*ngIf and \*ng For
* Router ParamMap
* TypeScript Compatibility
* Animations Package
* Dynamic Components
* Angular Universal
* Smaller and Quick
* View Engine – AOT Compilation
* Flat ES Modules (Flat ESM / FESM)
* Source Maps for Templates

Q4) What are Components in Angular 2/4?

Answer: A component is a reduced version of a directive. It receptacle do dom use and “replace” is quit too. Components are “restrict: E” and they are configured utilizing an object. In an AngularJS bond, a modularization is a group by use instead of type.

Q5) What is mean Routing and how does it work in Angular 2/4?

Answer: The basic building blocks of an Angular purpose are Ng Modules to use the router help in your app, you convey the Router that this system knows what those involve and how they should work.

Q6) Why is AngularJS used?

Answer: AngularJS is a structural frame for dynamic web apps. With AngularJS, designers can use HTML as the template language also it allows for the extension of HTML’s syntax to suggest this application’s components effortlessly. Angular performs much like the specific code you would differently have to write quite redundant.

Q7) What are the advantages of AngularJS?

Answer: The main Angular advantages above its closest rival, KnockoutJS, are No need to practice observable objects; Angular reports the page DOM and builds the bindings based on those Angular-specific element attributes. That demands smaller writing, the code is cleaner, easier to learn and less error-prone.

Q8) What is ECMAScript?

Answer: ECMAScript is a standard script language, developed with the help from Netscape and Microsoft and mainly taken from Netscape’s JavaScript, the widely-used scripting information that is done in Web pages to affect how they view or work for the user.

Q9) What are pipes in Angular 2/4?

Answer: Pipes are a valuable leader in Angular. They are a simple way to change values in a Lean template. There are any built-in channels, but you can further build your personal pipes. A pipe takes in use or contents and then delivers a value.

Q10) What Are Event Emitters in Angular 2/4?

Answer: Angular 2/4 will never guarantee us that Event Emitter will continue doing an Observable. Then this means refactoring our policy if it turns. This unique API us must request is its emit() method. Us should nevermore support manually to an Event Emitter.

[*Get Angular 4 Online Training*](https://www.gangboard.com/app-programming--scripting/angular-4-training?utm_source=GB_blog)

Q11) How to enable lazy loading in Angular 2/4?

Answer: Configure the routes. Set up an applying. If you don’t already have an app, you can regard the steps below to create one with the CLI. Create a center module with a routing link. Next, you’ll need some following module to route to. Add a part on the new module link. Add another characteristic module link. Set up the link.

Q12) How to share global data across components?

Answer: We often use `window` instance to grow also set global variables. In the browser program, `glass` is a single global setting thing. In the opposite view, Node.js. Use `GlobalRef` in elements. All done! Let’s use data: any; tagged Data Science. State oversize: creating the entire data science project.

Q13) What is the use of Interceptors?

Answer: We often use `window` instance to grow also set global variables. In the browser program, `glass` is a single global setting thing. In the opposite view, Node.js. Use `GlobalRef` in elements. All done! Let’s use data: any; tagged Data Science. State oversize: creating the entire data science project.

Q14) Architecture of Angular Apps

Answer: Architecture overview link. Angular is a platform and framework for developing client relationships in HTML and TypeScript. Angular is signed in TypeScript. It performs focus and elective functionality as a set of TypeScript libraries that you send into your apps

Q15) How Does directory of all external modules  function and files are stored in Angular 4?

Answer: I’ve been working on an Angular use for a couple of months now.I’m mainly concerned about looking at the src/app folder, where all Under actuality each of those files is when broken feathers into their .html and .scss counterparts. The first task is to choose how to split up many features into ‘Modules’.

Q16) What is angular data binding?

Answer: Data-binding in AngularJS apps is the automated synchronization of data between each model and view elements. The way that AngularJS implements databinding lets you treat this model as the single-source-of-truth in your contact. The movie is a projection of the image at all times.

Q17) What is Dependency Injection (DI) in Angular 4?

Answer: Dependency Injection in Angular. Dependency Injection (DI) is a focus concept of Angular 2 and allows a group to receives dependences from another class. Most of the time in Angular, dependency injection is done by including a service class into a component or module class.

Q18) What is dependency injection in angular?

Answer: Dependency Injection (DI) is a software object pattern that contracts with how elements make hold of their dependencies. The AngularJS injector subsystem is in charge of building components, choosing their dependencies, including providing them to other components as requested.

Q19) What is dependency injection in Web API?

Answer: The container automatically figures out the dependency relations. Common IoC cases also enable you to control things like objective lifetime and scope. “IoC” transfers for “inversion of power”, which is a common design anywhere a framework requests into request code.

Q20) Difference between ng-Class and ng-Style

Answer: ng-style is used to include javascript thing into style quality, not CSS class. The following directive will be turned to style=”color: blue” ng-style=”{color: ‘blue’}” And ng-class directive changes your thing into class quality. Following will be explained to class=”deleted” when the isDeleted variable is true. ng-class=”{‘deleted’: isDeleted}”

[*Get Angular 4 100% Practical Training*](https://www.gangboard.com/app-programming--scripting/angular-4-training?utm_source=GB_blog)

Q21) What are the Pipes angular 4?

Answer: Every application starts out including everything appears like a simple task: get data, change them, and show them to users. Getting data could be because easy as generating each local variable or as complex as streaming data over a WebSocket.

Q22) What is Redux?

Answer: Redux is an open-source JavaScript library for managing use state. It appears most commonly used with libraries such as React or Angular for developing user interfaces.

Q23) Explain Authentication and Authorization

Answer: Difference between Authentication and Authorization. Authentication means proving your own identity, while authorization involves granting way to the system. In simple terms, authentication is the process of proving who you are, while support is the method of checking what you should access to.

Q24) What is an AsyncPipe in Angular?

Answer: The async pipe subscribes to an Observable or Promise and displays the modern state it has released. When a new value is given, the async pipe marks the component to continue compared to changes. When the element gets removed, the async pipe unsubscribes automatically to bypass possible memory leaks.

Q25)Difference between Observables and Promises

Answer: An Observable is like a Stream (in many languages) and allows to give nothing or and events wherever the callback does need for each event. Often Observable is preferred over Promise because it gives the features of Insurance and more. With Observable, it doesn’t mean if you desire to handle 0, 1, or recurring issues. You can use the same API in each case.

Q26) What is difference between Angularjs observable and promise 2?

Answer: An exciting new innovation used with Angular is the Observable. This isn’t an Angular specific characteristic, but rather a proposed standard to running async data that order is added during the release. Observables open up a constant stream of data in which increased values of data package be released over time.

Q27) What is promise and observable in angular?

Answer: To get some relief, you consent to give it to them when it’s published. The function moved to new Promise is called the executor.do a job and then call resolve or discard to change this. Next, let’s see more useful examples of how promises can encourage us to write asynchronous code.

Q28) What are the Subscribe method of Angular 4?

Answer: The Observable isn’t an Angular particular highlight, but a new model for managing async data that will be involved in the ES7 release. Angular uses observables extensively in the development system and the HTTP service. Getting your cover nearby observables can be quite a thing, therefore I’m here to explain it the easy way.

Q29) What are ng- Model and how do we represent it?

Answer: The ng-model directive is done to connect the member variable called “pDescription” to the “text-area” control. Here we are adding the member variable to the field object called “p-Description” and setting a string value to the variable.

Q30) What is angular JSON?

Answer: An Angular Application Environment is JSON form data that tells that build system which records to change at you apply ng build and ng serve. Let’s say you have a final end REST API deployed toward a server that gives services to your Angular form.

[*Get Angular 4 Training with Real time Live Projects*](https://www.gangboard.com/app-programming--scripting/angular-4-training?utm_source=GB_blog)

Q31) What is two way data binding AngularJS?

Answer: Data Binding. Data-binding into AngularJS apps is this automated synchronization of data within the model and view components. The means that AngularJS performs data-binding lets you manage the model as the single-source-oftruth in your application. The view is a prediction of the model at all times.

Q32) What is ViewEncapsulation

Answer: To know ViewEncapsulation in Angular, first, we should assume the Shadow DOM. You can receive in detail of the Shadow DOM here. Simply put, the Shadow DOM makes Encapsulation to HTML Elements. Using the Shadow DOM, markup, styles, and functions are scoped to the part and do not clash with other nodes of the DOM.

Q33) What is the use of @Input and @Output?

Answer: @Input is a decorator to mark an input field and @Output is a decorator to identify an output property. @Input is used to determine an input property to achieve element property binding.

Q34) What is Transpiling in Angular

Answer: TypeScript is a basic language for Angular application development. It is a superset of JavaScript with design-time help for type safety and tooling. Browsers can’t perform TypeScript directly. Typescript must be “transpired” into JavaScript utilizing the tsc compiler, which needs some configuration.

Q35) Does angular 5 require TypeScript?

Answer: Angular is a common framework built uniquely in TypeScript, and as a result, using TypeScript with Angular gives a seamless experience. The Angular documentation not only maintains TypeScript as a first-class difficulty but uses it as its primary language.

Q36) How Angular 4 is different from Angular 2?

Answer: Angular is the blanket designation managed to refer to Angular 2, Angular 4 and all other reports that come after AngularJS. Both Angular 2 and 4 are open-source, TypeScript-based front-end web application platforms programs.

Angular 4 is one of the latest version of Angular. Although Angular 2 was a complete rewrite value of AngularJS, there are no main differences between Angular 2 and Angular 4. Angular 4 is only an extra and is backward compatible with Angular 2

Q37) What are Animation Functions in AngularJS?

Answer: This function can be called through the config form of an app. It uses a filtered role as the only evidence, which will next be applied to “filter” animations. Only when the filter function reflects true, will the animation be done.

Q38) What do you means understand by services and with reference to angularjs?

Answer: AngularJS services are substitutable objects that are wired commonly using dependency injection (DI). You can use services to produce and distribute code across your app. AngularJS just instantiates a service when a request element depends on that.

Q39) What is factory service in AngularJS?

Answer: In AngularJS, services are reusable singleton objects that are managed to produce more division code across your app. They can visit related in controllers, filters, directives. AngularJS gives you three ways: service, factory, and provider to perform a service.

[*Enroll Now!*](https://www.gangboard.com/app-programming--scripting/angular-4-training?utm_source=GB_blog)

Q40) what is Pure Functions?

Pure Functions do not change the provided value and give the same output always, when the same value is given multiple times.

Q41) what is impure functions?

Impure Functions change the provided value and give different output, when the same value is given multiple times.

Q42) what is store in Angular ?

Store is one single entity where you can combine the state of whole web application, and acts as a database for the application.

Q43) what are Actions in Angular?

Actions are simple objects that are dispatched to reducers when a user interacts with the application.

Q44) what is the use of @Input?

@Input() is used to import data from another component.

Q45) Which type of directional flow used in Angular 4?

one directional flow

Q46) what is @output ?

@Output and EventEmitter are used to emit event to the other component.

Q47) what is Lazyloading ?

It allows to load component asynchronously when a specific route is activated.

Q48) what is widget ?

External module get components, directives and pipes from widget module.

Q49) What is Router Module?

Routing configuration for another module is provided by router module.

Q50) What are the building blocks of Angular?

Modules,Component,Template,Directives,Data Binding,Services,Dependency Injection and Routing.

Q51) What is Transpiling

We use Transpiling to convert JS ( Javascript ) to TS( Typescript) using Tracuer .

Q52) What is Router Outlet ?

Router Outlet is the place holder for rendering the components .It’s same like ng-view in Angular Js

Q53) What is Routing ?!

Routing is used to navigate to different pages on clicking of corresponding link .

Q54) What is AOT?

The compilation of Angular application gets compiled internally . In case of Ahead of time the compilation does not happen every time.

Q55) What is Async Pipe?

To have promise / observable directly in the template , async pipe used . When we are using async pipe, there is no need of temporary property.

Q56) What is the use of Router.navigate ()?

We are using  router.navigate() to navigate to different component ( different pages ).

Q57) What is  string interpolation ?

It is used to bind the value to HTML where the variable is defined inside component .

Q58) What is the use of subscribe method ?

Independent execution of the observable happens,  while subscribe method is called and it’s always subsribed to observable.

[*Get Angular 4 100% Practical Training*](https://www.gangboard.com/app-programming--scripting/angular-4-training?utm_source=GB_blog)

Q59) What are the methods in promises?

Defer, Notify and resolve.

Q60) What is NPM?

Npm( Node package Manager) act as a online repository and package Manager for JS.

Q61) Whar is the difference between Angular JS and Angular 4 app?

Angular 4 is complete rewrite of Angular JS.Angular JS is based on controller based application and Angular 4 is component based application with Typescript .

Q62) What are the features of ECMA Script ?!

Generic , Namespace , Enumeration and Interface .

Q63) What are pipes in Angular?

Pipes are used to format the data within the template . Currency , percentage , date are some of the custom pipes.

Q64) what is @view child ?!

It’s used to communicate data to child component from parent class using class name.

Q65) Is it possible to load modules twice?

Yes . We can load modules twice and the latest imported module will be used .

Q66) Can we reimport classes and modules ?!

Yes

Q67) what is bootstrapping ?

Bootstrapping is starting of the angular application . It loads the root module and other components defined inside it .

Q68) When will be ngonit() will be called in angular app?!

It will be called only once  after first ngonchanges()

Q69) What kind of data that can we used inside async pipe ?!

Asynchronous and Stateful

Q70) What is impure pipe?!

The pipe that will execute during every component change  detection.

Q71) How does load children works?

Dynamically lazy loaded modules are loaded by router for it’s particular routes

Q72) what is the use of ngoninit()?

To fetch the initial component data

Q73) What is Router Gaurd?!

To protect component is activated through router.

Q74) What is the use of NgZone Service?

To run the Asynchronous process outside of $watch

Q1: What is difference between "declarations", "providers" and "import" in NgModule?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

* imports makes the exported declarations of other modules available in the current module
* declarations are to make directives (including components and pipes) from the current module available to other directives in the current module. Selectors of directives, components or pipes are only matched against the HTML if they are declared or imported.
* providers are to make services and values known to DI. They are added to the root scope and they are injected to other services or directives that have them as dependency.

A special case for providers are lazy loaded modules that get their own child injector. providers of a lazy loaded module are only provided to this lazy loaded module by default (not the whole application as it is with other modules).

🔗 **Source:** medium.com  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q2: What is AOT?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

The Angular Ahead-of-Time compiler pre-compiles application components and their templates during the build process. Apps compiled with AOT launch faster for several reasons.

* Application components execute immediately, without client-side compilation.
* Templates are embedded as code within their components so there is no client-side request for template files.
* You don't download the Angular compiler, which is pretty big on its own.
* The compiler discards unused Angular directives that a tree-shaking tool can then exclude.

🔗 **Source:** stackoverflow.com  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q3: Explain the difference between "Constructor" and "ngOnInit"

Topic: **Angular**  
Difficulty: ⭐⭐⭐

* The Constructor is a default method of the class that is executed when the class is instantiated and ensures proper initialization of fields in the class and its subclasses.
* ngOnInit is a life cycle hook called by Angular to indicate that Angular is done creating the component. We have to import OnInit in order to use like this (actually implementing OnInit is not mandatory but considered good practice).

Mostly we use ngOnInit for all the initialization/declaration and avoid stuff to work in the constructor. The constructor should only be used to initialize class members but shouldn't do actual "work".

🔗 **Source:** stackoverflow.com  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q4: What's new in Angular 6 and why shall we upgrade to it?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

* **Angular Elements** - Angular Elements is a project that lets you wrap your Angular components as Web Components and embed them in a non-Angular application.
* **New Rendering Engine: Ivy** - increases in speed and decreases in application size.
* **Tree-shakeable providers** - a new, recommended, way to register a provider, directly inside the @Injectable() decorator, using the new providedInattribute
* **RxJS 6** - Angular 6 now uses RxJS 6 internally, and requires you to update your application also. RxJS released a library called rxjs-compat, that allows you to bump RxJS to version 6.0 even if you, or one of the libraries you’re using, is still using one of the “old” syntaxes.
* **ElementRef<T>** - in Angular 5.0 or older, is that the said ElementRef had its nativeElement property typed as any. In Angular 6.0, you can now type ElementRef more strictly.
* **Animations** - The polyfill web-animations-js is not necessary anymore for animations in Angular 6.0, except if you are using the AnimationBuilder.
* **i18n** - possibility to have “runtime i18n”, without having to build the application once per locale.

**Part 1 – Angular 6 Interview Questions (Basic)**

This first part covers basic Angular 6 Interview Questions and Answers

**Q1. Provide clear difference with some proper example of code snippet between “declarations”, “providers”, and “imports” in ng module for angular 6?**

**Answer:**  
Please find below an explanation of declaration, providers, and imports:

* **Declarations**: This is one of the key features of Angular for available varieties components or pipes of a single directive for the current module to other directives of the current module. If someone willing to use some same declare component in the current module from other directives then declaration should need to be done properly.
* **Imports:**Helping of availability of other module components in a current module by importing the same.
* **Providers:** It is helping DI for identifying and understanding of using services and values.

**Q2. Explain in details with a proper example on genuine differences identified specifically between “constructor” and “ngoninit” for angular JS 6 version?**

**Answer:**  
This is the basic Angular 6 Interview Question asked in an interview. Please find details explanation below on constructor and ngonint for angular JS:

 Popular Course in this category

[**Angular JS Certification Training**6 Online Courses | 25+ Hours | Verifiable Certificate of Completion | Lifetime Access   
4.5 (1,289 ratings)](https://www.educba.com/software-development/courses/angular-js-certification-training/?btnz=edu-blg-inline-banner1)

[Course Price   
₹4999 ~~₹12999~~   
**View Course**](https://www.educba.com/software-development/courses/angular-js-certification-training/?btnz=edu-blg-inline-banner1)

Related Courses

[Programming Languages Course](https://www.educba.com/software-development/courses/programming-languages-course/?btnz=edu-blg-inline-banner1)[C Programming Course](https://www.educba.com/software-development/courses/c-programming-course/?btnz=edu-blg-inline-banner1)[Selenium Training Certification](https://www.educba.com/software-development/courses/selenium-training-certification/?btnz=edu-blg-inline-banner1)

* **Constructor**: Constructor is one of the default declarations for any specific class or object, it can be called every time when any class instantiated, also ensuring initialization properly of their subclasses and different instance variable fields.
* **Ngonint**: It is one of the first initialize method using by Angular, mention in the first component of an [angular life cycle](https://www.educba.com/angular-commands/). It mainly indicated that angular has been completed of creating entire require components properly. It is not mandatory to use but best practice to use.

**Q3. Application developed in Angular 5 is properly working and could not find any issue yet. Then why we plan to upgrade our application to Angular 6, what are new features introduced with this new version of Angular JS. Explain in details?**

**Answer:**  
Angular 6 has come with varieties upgradation [compare to Angular 5](https://www.educba.com/angular-5-interview-questions/), please find details explanation below on the same:

* **Elements of Angular:**One of the key features introduced in Angular 6, where some of the elements of Angular can be a wrap and represent as one of the web components. The main utility is the same component can be used easily for a non-angular project as the component can be considered as a normal web component.
* **New Engine for rendering called Ivy:**Helping of application performance especially in increasing speed and decreasing size of the application.
* **New providers called tree shakable:**One of the new ways for registering the provider, add inside the @Injectable() using some new attribute name is providedIn.
* **RxJS6:**One of the updated library used by Angular 6.
* **ElementRef:**ElementRef can define separately in case of Angular 6, not required to use native element property always.
* **Animation:**Angular 6 version introduces one new feature called Animation Builder.
* **I18n:**In Angular 6 we can able to use i18n at runtime as well, not require to build always.

Let us move to the next Angular 6 Interview Questions

**Q4. Explain in details about the reason of using the renderer method in Angular JS, whereas we can easily able to handle the same by using some native element methods. Was there any facility of using the same? Please explain?**

**Answer:**  
Angular can assume one platform and browser is using for rendering the details fetching data from the platform. Now if angular is going to use some of the native element of Angular DOM’s, then those elements only can be used for the application of same DOM environment. In angular 6, one new class has come call Render2, which helps of preparing that native element like and web component element so that it can be used anywhere avoiding DOM dependency.

**Q5. Explain in details about the compiler using by the Angular, called AOT (Ahead of time)?**

**Answer:**  
AOT stands for ahead of time. It is using by the angular for precompiling all the angular components and available templates at the time of build process. AOT always launched in angular based application than others.

**Part 2 – Angular 6 Interview Questions (Advanced)**

Let us now have a look at the advanced Angular 6 Interview Questions.

**Q6. One of the very popular keys using in Angular that is Zone, explain the same in details?**

**Answer:**  
ngZone is nothing but one of the wrappers of the JS file called Zone.js. This is one of the key libraries which is using for creating some context on using varieties asynchronous functions to make them trackable properly. Angular always dependent on zones for detect changing.

**Q7. Angular JS developer is sometimes planning to use Lazy loading modules frequently. Explain in details why it is required to use and how it impact directly?**

**Answer:**  
Somehow developer needs some feature module load lazily, then they can easily use one of the key property called loadChildren in the route [configuration of Angular JS](https://www.educba.com/uses-of-angular-js/). It is normally using by the developer when application size is increasing day by day. So utility of using the same:

* An expected module will be loaded based on the demand of the application.
* And application start will always be faster than usual.

Let us move to the next Angular 6 Interview Questions

**Q8. Explain in details about the lifecycle designed for directive and components in Angular JS especially for the newly introduced version 6.0?**

**Answer:**  
Below lifecycle normally followed by the components and directive of Angular JS:

* Constructor
* ngOnChanges
* nhOnInit
* ngDoCheck
* ngOnDestroy
* ngAfterContentInit (only for components)
* ngAfterContentChecked (only for components)
* ngAfterViewInit (only for components)
* ngAfterViewChecked (only for components)

**Q9. Is it possible to include one embedded view from a defined templateRef which already prepared? If yes, please explain the same?**

**Answer:**  
This is the most asked Angular 6 Interview Question in an interview. The embedded view can be created by using the createdEmbeddedView method and can easily attach the same to the available DOM by using the specific utility of TemplateRef.

@Component({  
selector: 'app-root',  
template: `  
<ng-template #template let-name='example'><div>{{…}}</ng-template>  
})  
export class ApplicationComponent implements AfterViewChecked {  
@ViewChild('template1', { read: TemplateRef }) \_template: TemplateRef<…>;  
constructor() { }  
ngAfterViewChecked() {  
this.vc.createEmbeddedView(this.\_template1, {example: '….'});  
}  
}

**Q10. Explain in details if someone willing to identify exact route change of angular then how they can do this?**

**Answer:**  
In Angular 6 there has one option called Rx event which needs to subscribe to a specific instance of Router. Things can be done by below approaches:

Class SomeClass{

Constructor(private route : Router){

route.subscribe((val) => ……)

}

}

What is AngularJS?

AngularJS is a framework to build large scale and high performance web application while keeping them as easy-to-maintain. Following are the features of AngularJS framework.

* AngularJS is a powerful JavaScript based development framework to create RICH Internet Application (RIA).
* AngularJS provides developers options to write client side application (using JavaScript) in a clean MVC (Model View Controller) way.
* Application written in AngularJS is cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser.
* AngularJS is open source, completely free, and used by thousands of developers around the world. It is licensed under the Apache License version 2.0.

What is data binding in AngularJS?

Data binding is the automatic synchronization of data between model and view components. ng-model directive is used in data binding.

What is scope in AngularJS?

Scopes are objects that refer to the model. They act as glue between controller and view.

What are the controllers in AngularJS?

Controllers are JavaScript functions that are bound to a particular scope. They are the prime actors in AngularJS framework and carry functions to operate on data and decide which view is to be updated to show the updated model based data.

What are the services in AngularJS?

AngularJS come with several built-in services. For example $https: service is used to make XMLHttpRequests (Ajax calls). Services are singleton objects which are instantiated only once in app.

What are the filters in AngularJS?

Filters select a subset of items from an array and return a new array. Filters are used to show filtered items from a list of items based on defined criteria.

Explain directives in AngularJS.

Directives are markers on DOM elements (such as elements, attributes, css, and more). These can be used to create custom HTML tags that serve as new, custom widgets. AngularJS has built-in directives (ng-bind, ng-model, etc) to perform most of the task that developers have to do.

Explain templates in AngularJS.

Templates are the rendered view with information from the controller and model. These can be a single file (like index.html) or multiple views in one page using "partials".

What is routing in AngularJS?

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

What is deep linking in AngularJS?

Deep linking allows you to encode the state of application in the URL so that it can be bookmarked. The application can then be restored from the URL to the same state.

What are the advantages of AngularJS?

Following are the advantages of AngularJS.

* AngularJS provides capability to create Single Page Application in a very clean and maintainable way.
* AngularJS provides data binding capability to HTML thus giving user a rich and responsive experience.
* AngularJS code is unit testable.
* AngularJS uses dependency injection and make use of separation of concerns.
* AngularJS provides reusable components.
* With AngularJS, developer writes less code and gets more functionality.
* In AngularJS, views are pure html pages, and controllers written in JavaScript do the business processing.
* AngularJS applications can run on all major browsers and smart phones including Android and iOS based phones/tablets.

What are the disadvantages of AngularJS?

Following are the disadvantages of AngularJS.

* **Not Secure** − Being JavaScript only framework, application written in AngularJS are not safe. Server side authentication and authorization is must to keep an application secure.
* **Not degradable** − If your application user disables JavaScript then user will just see the basic page and nothing more.

Which are the core directives of AngularJS?

Following are the three core directives of AngularJS.

* **ng-app** − This directive defines and links an AngularJS application to HTML.
* **ng-model** − This directive binds the values of AngularJS application data to HTML input controls.
* **ng-bind** − This directive binds the AngularJS Application data to HTML tags.

Explain AngularJS boot process.

When the page is loaded in the browser, following things happen:

* HTML document is loaded into the browser, and evaluated by the browser. AngularJS JavaScript file is loaded; the angular *global* object is created. Next, JavaScript which registers controller functions is executed.
* Next AngularJS scans through the HTML to look for AngularJS apps and views. Once view is located, it connects that view to the corresponding controller function.
* Next, AngularJS executes the controller functions. It then renders the views with data from the model populated by the controller. The page gets ready.

What is MVC?

**M**odel **V**iew **C**ontroller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

* **Model** − It is the lowest level of the pattern responsible for maintaining data.
* **View** − It is responsible for displaying all or a portion of the data to the user.
* **Controller** − It is a software Code that controls the interactions between the Model and View.

Explain ng-app directive.

ng-app directive defines and links an AngularJS application to HTML. It also indicate the start of the application.

Explain ng-model directive.

ng-model directive binds the values of AngularJS application data to HTML input controls. It creates a model variable which can be used with the html page and within the container control( for example, div) having ng-app directive.

Explain ng-bind directive.

ng-bind directive binds the AngularJS Application data to HTML tags. ng-bind updates the model created by ng-model directive to be displayed in the html tag whenever user input something in the control or updates the html control's data when model data is updated by controller.

Explain ng-controller directive.

ng-controller directive tells AngularJS what controller to use with this view. AngularJS application mainly relies on controllers to control the flow of data in the application. A controller is a JavaScript object containing attributes/properties and functions. Each controller accepts $scope as a parameter which refers to the application/module that controller is to control.

How AngularJS integrates with HTML?

AngularJS being a pure javaScript based library integrates easily with HTML.

**Step 1** − Include angularjs javascript libray in the html page

<head>

<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

</head>

**Step 2** − Point to AngularJS app

Next we tell what part of the HTML contains the AngularJS app. This done by adding the *ng-app* attribute to the root HTML element of the AngularJS app. You can either add it to *html* element or *body* element as shown below:

<body ng-app = "myapp">

</body>

Explain ng-init directive.

ng-init directive initializes an AngularJS Application data. It is used to put values to the variables to be used in the application.

Explain ng-repeat directive.

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

What are AngularJS expressions?

Expressions are used to bind application data to html. Expressions are written inside double braces like {{ expression}}. Expressions behave in same way as ng-bind directives. AngularJS application expressions are pure JavaScript expressions and outputs the data where they are used.

Explain uppercase filter.

Uppercase filter converts a text to upper case text.

In below example, we've added uppercase filter to an expression using pipe character. Here we've added uppercase filter to print student name in all capital letters.

Enter first name:<input type = "text" ng-model = "student.firstName">

Enter last name: <input type = "text" ng-model = "student.lastName">

Name in Upper Case: {{student.fullName() | uppercase}}

Explain lowercase filter.

Lowercase filter converts a text to lower case text.

In below example, we've added lowercase filter to an expression using pipe character. Here we've added lowercase filter to print student name in all lowercase letters.

Enter first name:<input type = "text" ng-model = "student.firstName">

Enter last name: <input type = "text" ng-model = "student.lastName">

Name in Upper Case: {{student.fullName() | lowercase}}

Explain currency filter.

Currency filter formats text in a currency format.

In below example, we've added currency filter to an expression returning number using pipe character. Here we've added currency filter to print fees using currency format.

Enter fees: <input type = "text" ng-model = "student.fees">

fees: {{student.fees | currency}}

Explain filter filter.

filter filter is used to filter the array to a subset of it based on provided criteria.

In below example, to display only required subjects, we've used subjectName as filter.

Enter subject: <input type = "text" ng-model = "subjectName">

Subject:

<ul>

<li ng-repeat = "subject in student.subjects | filter: subjectName">

{{ subject.name + ', marks:' + subject.marks }}

</li>

</ul>

Explain orderby filter.

orderby filter orders the array based on provided criteria.

In below example, to order subjects by marks, we've used orderBy marks.

Subject:

<ul>

<li ng-repeat = "subject in student.subjects | orderBy:'marks'">

{{ subject.name + ', marks:' + subject.marks }}

</li>

</ul>

Explain ng-disabled directive.

ng-disabled directive disables a given control.

In below example, we've added ng-disabled attribute to a HTML button and pass it a model. Then we've attached the model to an checkbox and can see the variation.

<input type = "checkbox" ng-model = "enableDisableButton">Disable Button

<button ng-disabled = "enableDisableButton">Click Me!</button>

Explain ng-show directive.

ng-show directive shows a given control.

In below example, we've added ng-show attribute to a HTML button and pass it a model. Then we've attached the model to a checkbox and can see the variation.

<input type = "checkbox" ng-model = "showHide1">Show Button

<button ng-show = "showHide1">Click Me!</button>

Explain ng-hide directive.

ng-hide directive hides a given control.

In below example, we've added ng-hide attribute to a HTML button and pass it a model. Then we've attached the model to a checkbox and can see the variation.

<input type = "checkbox" ng-model = "showHide2">Hide Button

<button ng-hide = "showHide2">Click Me!</button>

Explain ng-click directive.

ng-click directive represents a AngularJS click event.

In below example, we've added ng-click attribute to a HTML button and added an expression to updated a model. Then we can see the variation.

<p>Total click: {{ clickCounter }}</p></td>

<button ng-click = "clickCounter = clickCounter + 1">Click Me!</button>

l

How angular.module works?

angular.module is used to create AngularJS modules along with its dependent modules. Consider the following example:

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

Here we've declared an application **mainApp** module using angular.module function. We've passed an empty array to it. This array generally contains dependent modules declared earlier.

How to validate data in AngularJS?

AngularJS enriches form filling and validation. We can use $dirty and $invalid flags to do the validations in seamless way. Use novalidate with a form declaration to disable any browser specific validation.

Following can be used to track error.

* **$dirty** − states that value has been changed.
* **$invalid** − states that value entered is invalid.
* **$error** − states the exact error.

Explain ng-include directive.

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

<div ng-app = "" ng-controller = "studentController">

<div ng-include = "'main.htm'"></div>

<div ng-include = "'subjects.htm'"></div>

</div>

How to make an ajax call using Angular JS?

AngularJS provides $https: control which works as a service to make ajax call to read data from the server. The server makes a database call to get the desired records. AngularJS needs data in JSON format. Once the data is ready, $https: can be used to get the data from server in the following manner:

function studentController($scope,$https:) {

var url = "data.txt";

$https:.get(url).success( function(response) {

$scope.students = response;

});

}

What is use of $routeProvider in AngularJS?

$routeProvider is the key service which set the configuration of urls, maps them with the corresponding html page or ng-template, and attaches a controller with the same.

What is $rootScope?

Scope is a special JavaScript object which plays the role of joining controller with the views. Scope contains the model data. In controllers, model data is accessed via $scope object. $rootScope is the parent of all of the scope variables.

What is scope hierarchy in AngularJS?

Scopes are controllers specific. If we define nested controllers then child controller will inherit the scope of its parent controller.

<script>

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

mainApp.controller("shapeController", function($scope) {

$scope.message = "In shape controller";

$scope.type = "Shape";

});

mainApp.controller("circleController", function($scope) {

$scope.message = "In circle controller";

});

</script>

Following are the important points to be considered in above example.

* We've set values to models in shapeController.
* We've overridden message in child controller circleController. When "message" is used within module of controller circleController, the overridden message will be used.

What is a service?

Services are JavaScript functions and are responsible to do specific tasks only. Each service is responsible for a specific task for example, $https: is used to make ajax call to get the server data. $route is used to define the routing information and so on. Inbuilt services are always prefixed with $ symbol.

What is service method?

Using service method, we define a service and then assign method to it. We've also injected an already available service to it.

mainApp.service('CalcService', function(MathService) {

this.square = function(a) {

return MathService.multiply(a,a);

}

});

What is factory method?

Using factory method, we first define a factory and then assign method to it.

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

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b

}

return factory;

});

What are the differences between service and factory methods?

factory method is used to define a factory which can later be used to create services as and when required whereas service method is used to create a service whose purpose is to do some defined task.

Which components can be injected as a dependency in AngularJS?

AngularJS provides a supreme Dependency Injection mechanism. It provides following core components which can be injected into each other as dependencies.

* value
* factory
* service
* provider
* constant

What is provider?

provider is used by AngularJS internally to create services, factory etc. during config phase(phase during which AngularJS bootstraps itself). Below mention script can be used to create MathService that we've created earlier. Provider is a special factory method with a method get() which is used to return the value/service/factory.

//define a module

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

...

//create a service using provider which defines a method square to return square of a number.

mainApp.config(function($provide) {

$provide.provider('MathService', function() {

this.$get = function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b;

}

return factory;

};

});

});

What is constant?

constants are used to pass values at config phase considering the fact that value cannot be used to be passed during config phase.

mainApp.constant("configParam", "constant value");

Is AngularJS extensible?

Yes! In AngularJS we can create custom directive to extend AngularJS existing functionalities.

Custom directives are used in AngularJS to extend the functionality of HTML. Custom directives are defined using "directive" function. A custom directive simply replaces the element for which it is activated. AngularJS application during bootstrap finds the matching elements and do one time activity using its compile() method of the custom directive then process the element using link() method of the custom directive based on the scope of the directive.

On which types of component can we create a custom directive?

AngularJS provides support to create custom directives for following type of elements.

* **Element directives** − Directive activates when a matching element is encountered.
* **Attribute** − Directive activates when a matching attribute is encountered.
* **CSS** − Directive activates when a matching css style is encountered.
* **Comment** − Directive activates when a matching comment is encountered.

What is internationalization?

Internationalization is a way to show locale specific information on a website. For example, display content of a website in English language in United States and in Danish in France.

How to implement internationalization in AngularJS?

AngularJS supports inbuilt internationalization for three types of filters currency, date and numbers. We only need to incorporate corresponding js according to locale of the country. By default it handles the locale of the browser. For example, to use Danish locale, use following script

<script src = "https://code.angularjs.org/1.2.5/i18n/angular-locale\_da-dk.js"></script>

1. **Question 1. What Are Components In Angular?**

**Answer :**

**The Concepts of Angular Components  -**

Components are the most basic building block of a UI in Angular applications and it controls views (HTML/CSS). They also communicate with other components and services to bring functionality to your applications.

Technically components are basically TypeScript classes that interact with the HTML files of the components, which get displayed on the browsers.

The component is the core functionality of Angular applications but you need to know to pass the data into the components to configure them.

1. **Question 2. What's New In Angular 6? What Are Improvements In Angular 6?**

**Answer :**

The Angular Team are working on lots of bug fixes, new features and added/update/remove/ re-introduce/ and many more things.

**Let’s start to explore all changes of Angular 6 step by step:**

**Added ng update -** This CLI commands will update your angular project dependencies to their latest versions. The ng update is normal package manager tools to identify and update other dependencies.

[HTML 5 Interview Questions](https://www.wisdomjobs.com/e-university/html-5-interview-questions.html)

1. **Question 3. What Are The Ngmodule Metadata Properties?**

**Answer :**

The NgModule decorator identifies AppModule as a NgModule class.

The NgModule takes a metadata object that tells Angular how to compile and launch the application.

**The NgModule importance metadata properties are as follows –**

* + providers
  + declarations
  + imports
  + exports
  + entryComponents
  + bootstrap
  + schemas
  + id

1. **Question 4. What Types Of Ngmodules?**

**Answer :**

**There are four types of NgModules –**

* + Features Module
  + Routing Module
  + Service Module
  + Widget Module
  + Shared Module

[HTML 5 Tutorial](https://www.wisdomjobs.com/e-university/html-5-tutorial-207.html)

1. **Question 5. What Is A Cookie?**

**Answer :**

A cookie is a small piece of data sent from a website and stored on the user's machine by the user's web browsers while the user is browsing.

[Java Script Interview Questions](https://www.wisdomjobs.com/e-university/java-script-interview-questions.html)

1. **Question 6. What Is Pure Pipe?**

**Answer :**

Angular executes a pure pipe only when it detects a pure change to the input value. A pure change can be primitive or non-primitive.

Primitive data are only single values, they have not special capabilities and the non-primitive data types are used to store the group of values.

@Pipe({

**name**: 'currency'

})

1. **Question 7. What Is Impure Pipe?**

**Answer :**

Angular executes an impure pipe during every component change detection cycle. An impure pipe is called often, as often as every keystroke or mouse-move.

If you want to make a pipe impure that time you will allow the setting pure flag to false.

@Pipe({

**name:** 'currency',

  pure:false

})

[Java Script Tutorial](https://www.wisdomjobs.com/e-university/java-script-tutorial-209.html) [PHP Interview Questions](https://www.wisdomjobs.com/e-university/php-interview-questions.html)

1. **Question 8. What Is Parameterizing Pipe?**

**Answer :**

A pipe can accept any number of optional parameters to achieve output. The parameter value can be any valid template expressions. To add optional parameters follow the pipe name with a colon (:). Its looks like- currency: 'INR'

**In the following example –**

<h2>The birthday is - {{ birthday | date:"MM/dd/yy" }} </h2>

<!-- Output - The birthday is - 10/03/1984 -->

1. **Question 9. What Is Chaining Pipe?**

**Answer :**

The chaining Pipe is used to perform the multiple operations within the single expression. This chaining operation will be chained using the pipe (I).

In the following example, to display the birthday in the upper case- will need to use the inbuilt date-pipe and upper-case-pipe.

**In the following example –**

{{ birthday | date | uppercase}}

[Angular JS Interview Questions](https://www.wisdomjobs.com/e-university/angular-js-interview-questions.html)

1. **Question 10. Why You Use Browsermodule, Commonmodule, Formsmodule, Routermodule, And Httpclientmodule?**

**Answer :**

**BrowserModule –** The browser module is imported from @angular/platform-browser and it is used when you want to run your application in a browser.

**CommonModule –** The common module is imported from @angular/common and it is used when you want to use directives - NgIf, NgFor and so on.

**FormsModule –** The forms module is imported from @angular/forms and it is used when you build template driven forms.

**RouterModule –** The router module is imported from @angular/router and is used for routing RouterLink, forRoot, and forChild.

**HttpClientModule –**The HttpClientModule is imported from @angular/common/http and it used to initiate HTTP request and responses in angular apps. The HttpClient is more modern and easy to use the alternative of HTTP.

**1. What is the syntax of ForEach loop? which loop would you use to parse a JSON and why?**

Below is the syntax of for each loop:

angular.ForEach(students,function(value,key)

{

//some code

}

To parse JSON, we can use any loop, but I would use **for each** loop because it will minimize my code by eliminating the need to store the length of a JSON in a variable.

**2. Explain MVC in reference to angular?**

AngularJs is an MVC based framework, where Model for a controller contains data, the controller for a view contains the logic to manipulate that data, and the view is the HTML that displays the data.

A $scope can be considered as a model, whereas the functions written in angular controller modifies the $scope and HTML display the value of the scope variable.

**3. What is two-way binding?**

Two-way binding means that when data in the view is changed the underlying model gets updated automatically and when a model from the controller is changed the view gets updated.

**4. Can there be two ng-app for a single angular application?**

No, there can't be more than one ng-apps for a single AngularJS application.  
The ng-app directive conveys AngularJS application that it is the root element. In your HTML document, you can have a single ng-app directive only. In case of more than one ng-app directives, the first appearance will be used.

**5. What is $scope?**

$scope is a model for a controller and helps the controller in interacting with the view.

(This is a super short answer to this question, but it is complete in every sense. Try not to put any additional angular terms).

**6. Name a few inbuilt angular filters?**

Currency, lowercase, uppercase, number, date are few inbuilt angular filters.

{{nameOfStudent|uppercase}}

**Intermediate AngularJS Interview Questions**

Let us take a look at some mid-level AngularJS interview questions

**7. What are custom filters? Write down a syntax of the same?**

With AngularJS we can create our own filters. This can be done by associating the filter to our module. These types of filters are called custom filters.

Below is the code to count the number of elements in the string by using filter:

angular.module('myCountFilterApp', [])

.filter('count',function()

{

return(function(input)

{

var out=[];

out=input.split(',');

return out.length;

})

});

In the above example, if the string is **"21, 34, 45"** then output after applying filter will be **3**.

Here is some more information on [custom filters](https://docs.angularjs.org/guide/filter).

**8. What is the difference between ng-if and ng-show?**

Ng-if doesn’t render the portion of DOM element on which it is associated if the specified condition is not met whereas ng-show renders the DOM element but set its CSS property of display to none if the specified condition is not met.

**9. What is the purpose of the $watch?**

The purpose of $watch is to keep track of the old and new value of the watched expression. Below is the code of using $watch.

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

console.log("I've changed : ", newValue);

});

**10. What is the purpose of $rootScope?**

$rootScope helps in communication between different controllers of an application. AngularJS can have only one rootScope for an app.

**11. Write down the syntax for sending HTTP request?**

$http({

method: "POST",

url: "URL",

data: JSON.stringify(value),

type: 'POST',

contentType: 'application/json; charset=utf-8'

}).then(function (response)

{

// success action

});

**12. Where should one use form action instead of $http for accessing a method on a server?**

Form action should be used at a place where the server-side method takes the control to some other view in other word leads to redirection whereas HTTP request should be used where the server method returns some data.

**13. What is the purpose of find index in AngularJS and what does it return if no value is found?**

Find index returns the position of an element in an object. If the requested element is not found then -1 is returned.

var index = $scope.items.findIndex(record => record.date =='2018-12-12');

In the above code, index of the object is returned where item.date=2018-12-12.

**14. What is ng-init used for?**

Ng-init is used in a scenario where we want some action to be done before the initialization of a portion of the DOM element.

**15. Can I set an angular variable from PHP session variable without sending an HTTP request?**

Yes, we can do that by injecting PHP in the required place.

$scope.name='<?= $session['name'] ?>';

This will work only if you are using PHP to render the HTML and the above javascript is writter in <script> tag inside the php file.

**16. What is the significance of pipe operator in angularJs and What would be the result of following expression**

{{ Somevalue|lowercase|uppercase}}

Pipe operator in AngularJS represents **filters** that are used on the expression. The preference order is from left to right. So, the result of the above expression would be **SOMEVALUE**.

**17. Explain the following code:**

<div ng-repeat="hotel in hotels|filter:setFinalFilter|orderBy : 'minPrice'">{{hotel.name}}</div>

Here, **setFinalFilter** is a custom filter used on the hotels object. The result would display the name of filtered hotels in ascending order of their minPrice.

**18. What is service in AngularJS used for?**

Services in AngularJS are objects which are used to communicate within entire applications.

app.service('sharedData', function () {

//methods to get and set variable

});

**19. What is dependency injection and what are the benefits of it?**

Dependency injection is a powerful design pattern that allows separating the concerns of different components in an application and provides a way to inject the dependent component into the client component.

Consider the below code:

myApp.controller('myController', function ($scope, $http, $location)

{

//logic

});

Here, a controller is declared with its dependencies.

.**$http, $location** are all services which are injected into the controller as a dependent entity. All of them have some independent specific task associated with it. MyController does not need to create their instance, but it can directly use them.

**20. Write down the syntax of creating a new date object?**

The syntax for creating new data object:

$scope.newDate=new Date();

**21. Can parent controller access the methods of child controller or vice versa?**

No, the parent controller cannot access the methods of child controller, but the child controller can access the methods of the parent controller.

**22. Evaluate the feasibility of the following code:**

<select ng-options="student.name for student in students"></select>

This code will give a syntax error. We cannot use ng-options without ng-model. Using the array or object obtained by evaluating the ngOptions expression, the ngOptions attribute dynamically generates a list of < option > elements for the < select > element. On selecting an item in the < select > menu, the array element or object property will be bound to the model identified by the ngModel directive. Hence, ng-model is must with ng-options.

**23. Can I use ternary operator in angular expression?**

Yes, ternary operators or no flow operators can be used in an angular expression.

{{name==undefined:'no name specified'?name}}

**24. When is $location used? Explain with some scenario.**

$location is an **angular service** which keeps track of the URL of the application and makes it available to a controller. If $location is changed from the controller, the impact is reflected in the address bar, and vice versa is also true.

**Advanced AngularJS Interview Questions**

**25. Explain the purpose of track by in ng-repeat?**

In AngularJS, directives like ng-repeat keep track of all the elements to minimize the DOM creation. It does that by storing the object instance when a new element is added to the collection. Angular does not re-render the entire element set; it just renders the new element.

When ng-repeats is used with the object having some unique id, the tracking should be done by that identifier, instead of the object instance. Consider the below code.

item in items track by item.id

Here, tracking is done based on item id.

**26. How is scope in directive different from scope in controller?**

Both $scope and scope are instances of scope object. The difference lies in the name that is used for them. In order to explain the difference between $scope and scope, we need to know about directives with an isolated scope.

Let us try to understand this with the following code:

.directive('testDirective', function() {

return {

scope: {},

link: function(myScopeVar, elem,attr) {

console.log(scope)

}

}

})

})

In the above code, a [directive with an isolated scope is declared](https://docs.angularjs.org/guide/directive). The link used in above code is a regular Javascript function with signature scope, element and attribute. The name of scope object is not important because whatever name you give to this element, it will be linked to the directive’s scope object. That is why, using myScopeVar will not give any error. The $scope, on other hand, cannot be used with any other name.

For further clarification, let us see the following code:

app.controller(‘myController’,function(newScope)

{

})

The above code will give an error. Here are more details on [scope v/s $scope](https://thinkster.io/scope-vs-scope).

**27. Explain this code and possible values of restrict attribute?**

.directive('myCustomer', function() {

return {

restrict: 'E',

scope: {

customerInfo: '=info'

},

templateUrl: 'my-customer.html'

};

});

Here, a custom directive of name myCustomer is declared. The directive is restricted to element name only. The directive has its own isolated scope which has a property customerInfo and takes its value from info attribute of the myCustomer element.

The **template/view** of the directive is my-customer.html. The possible declaration of this directive would be .

The possible values of restrict can be:  
'A' - only matches attribute name  
'E' - only matches element name  
'C' - only matches the class name  
'M' - only matches the comment

**28. What is the difference between compile and link?**

Compile can be considered as a service which traverses the HTML, find for all the directives and returns a link function.

Link, on the other hand, combines the model with a view. Any change in model reflects the change in view and any change in view reflects in the model.

Here is more detail on [compile and link](https://www.naukri.com/docs.angularjs.org/guide/compiler).

**29. Explain strict conceptual escaping?**

AngularJS treats all values as untrusted/unsecured by default in HTML or sensitive URL bindings. When binding untrusted values, AngularJS will automatically run security checks on them (sanitizations, whitelists, depending on context), or throw an error when it cannot guarantee the security of the result. This behaviour depends strongly on contexts: HTML can be sanitized, but template URLs cannot.

To illustrate this, consider the ng-bind-html directive. It renders its value directly as HTML. When given an untrusted input, AngularJS will attempt to sanitize it before rendering if a sanitizer is available. To bypass sanitization and render the input as-is, you will need to mark it as trusted.

Here is more detail on [SCE](https://docs.angularjs.org/api/ng/service/$sce).

**30. How logs are maintained in AngularJS?**

The follow-up question can be how to Blackbox AngularJS source in the browser and which all browser supports Blackbox?

Logs are maintained with the help of $log service. The main purpose of this service is to help in debugging and troubleshooting. This is done with the help of four methods.  
1. log()-writes a log message in the console  
2. info()-writes an information message  
3. warn()-write a warning message  
4. error()-writes an error message  
5. debug()-writes a debug message

$log.error(‘this will displayed as an error in console’)

20 Frequently Asked AngularJS Interview Questions and Answers

[By Srihari Sasikumar](https://www.simplilearn.com/authors/srihari-sasikumar?source=frs_detailsPage)Last updated on Jul 18, 201985707



[AngularJS](https://angularjs.org/), the self-proclaimed “superheroic JavaScript MVW framework,” has been around since 2009, and it remains one of the most popular web development frameworks the world over. According to [Indeed.com](https://www.indeed.com/jobtrends/q-Angularjs.html), the demand for developers has been on the rise in recent months, and this trend is expected to continue.

AngularJS can be a demanding technology, prompting a salary scope of anything from $60/hour for a freelance developer to $70,000/year for a Front End Developer/Engineer in an organization. Since this certification is a requirement for a large percentage of web-developer job postings, you can assume that there is never a lack of qualified candidates.

In every interview, you will have to contend with fierce competition. To perform well and get your dream job, you have to prepare in advance. To help you, we’ve outlined 20 frequently asked AngularJS interview questions and the kind of answers your interviewer is likely looking for.

Here Are 20 Frequently Asked AngularJS Interview Questions:

1. In 30 seconds, define AngularJS.

AngularJS is an open-source JavaScript framework designed for creating dynamic single web page applications with fewer lines of code.

1. Mention some advantages of using the AngularJS framework.

Some of the key advantages of using AngularJS framework include:

* + It provides an excellent “desktop-like” experience to the end-user.
  + By freeing the developer from having to register callbacks manually or write repetitive low-level DOM manipulation tasks, it saves months of development.
  + By separating DOM manipulation from app logic, it makes code modular and easy to test.
  + It supports two-way data binding.

1. What is the main thing that you would need to change if you were migrating from AngularJS 1.4 to AngularJS 1.5?

To adapt to the new AngularJS 1.5 components, you would need to change .directive to .component.

1. Is AngularJS compatible with all browsers?

Yes.

1. What are the key features of AngularJS?
   * Scope
   * Controller
   * Model
   * View
   * Services
   * Data Binding
   * Directives
   * Filters
   * Testable
2. Define scope in AngularJS.

The scope is a unique JavaScript object that plays the role of joining the controller (JavaScript) with the views (HTML). The controller sets properties on the scope, and the view binds to them.

1. Explain the concept of scope hierarchy.

Each AngularJS application has only one root scope. Child controllers, however, create scope for each view. When the new scopes are created, they are added to their parent root scope as child scopes. This creates a hierarchical structure when they are attached.

|  |
| --- |
| Are you a front end developer and looking to enhance your skills? Check out the [Angular Training Course](https://www.simplilearn.com/angular-certification-training). |

1. What is data binding in AngularJS and What is the difference between one-way and two-way binding?

Data binding is the automatic attunement of data between the view and model components. AngularJS uses two-way data binding. In one-way binding, the scope variable in the HTML is set to the first value that its model is assigned to.  
  
In two-way binding, the scope variable changes its value every time its model binds to a different value.

1. What are directives?

Directives are unique markers on a DOM element that tell the HTML compiler to attach a specified behavior to the DOM element. Directives start with ng-prefix. Some of the built-in directives include ngClass, ngApp, ngRepeat, ngModel, ngBind and ngInit.

1. List the different types of directives.

There are four types of directives:

* + Comment directives
  + CSS class directives
  + Attribute directives
  + Element directives

1. In what ways can you use a directive?

You can use a directive as an element, attribute, or class name. To define the way your directive will be used, you need to set the restrict option in the directive declaration. The restrict option can be set to:  
  
‘A’ - Only matches attribute name  
‘C’ - Only matches class name  
‘E’ – Only matches element name  
  
You can combine these restrictions to produce needed results.

1. Explain what a digest cycle is in AngularJS?

During every digest cycle, all new scope model values are compared against the previous values. This is called dirty checking. If change is detected, watches set on the new model are fired, and another digest cycle executes. This goes on until all models are stable.  
  
The digest cycle is triggered automatically, but it can be called manually using “.$apply().”

1. What are the common ways of communication between modules of your application, using core AngularJS functionality?

The common ways of communication include:

* + Using events
  + Using services
  + By assigning models on $rootScope
  + Directly between controllers using ControllerAs and other forms of inheritance
  + Directly between controllers using $parent, $$childHead, $$nextSibling

1. Explain what the link function is and how it differs from compile.

The link function combines the directives with a scope to produce a live view.  
The link function is responsible for instance DOM manipulation and for registering DOM listeners.  
The compile function is responsible for template DOM manipulation as well as the collection of all the directives.

1. Explain e2e testing of AngularJS applications.

End-to-end (e2e) testing involves testing an application from start to finish to determine whether all the components are working correctly. It catches issues within an application related to integration and flow.

1. What is dependency injection?

Dependency injection is the process where the dependent objects are injected rather than being created by the consumer.

1. What are the benefits of dependency injection?

Dependency injection has two significant benefits: Testing and decoupling.

1. What is a Single Page Application (SPA)?

SPA is the concept whereby pages are loaded from the server not by doing postbacks, instead of by creating a single shell page or master page and loading the web pages into the master page.

1. How can a SPA be implemented in AngularJS?

SPA can be implemented with Angular by using Angular routes.

1. How can digest cycle time be decreased?

Digest cycle time can be decreased by decreasing the number of watchers. To do this, you can:

* + Use web worker
  + Work in batches
  + Cache DOM
  + Remove unnecessary watchers
  + Use one-time Angular binding

Angular 7 Interview Questions



A list of top frequently asked **Angular 7 Interview Questions** and answers are given below.

1) What is Angular 7? How is it different from AngularJS?

Angular7 is the latest and recent version of Angular. AngularJS was the first version of Angular which is also known as Angular 1.0.

Angular7 is the complete rewrite of the Angular1.0. It supports two-way data binding, and some other features like ng update, ng add, Angular Elements, Angular Material + CDK Components, Angular Material Starter Components, CLI Workspaces, Library Support, Tree Shakable Providers, Animations Performance Improvements, and RxJS v6 etc.

2) What is Angular framework?

Angular is a TypeScript-based open-source web framework and a platform. It is used to build web/ mobile and desktop applications.

**Main features of this framework are:** Declarative templates, dependency injection, end to end tooling etc. These features make web development easy in Angular.

3) What is the difference between AngularJS and Angular?

Angular is a complete rewrite of AngularJS. It is a component-based framework in which an application is a tree of individual components.

**Difference between AngularJS and Angular:**

|  |  |
| --- | --- |
| **AngularJS** | **Angular** |
| AngularJS is based on MVC architecture. | Angular is based on Service/Controller. |
| It uses JavaScript to build the application. | It uses TypeScript to build the application. |
| It follows controller concept. | It follows Component based UI approach. |
| It is not a mobile-friendly framework. | It is a mobile friendly framework. |
| It is very difficult to create a SEO friendly application in AngularJS. | By using Angular, you can easily create a SEO friendly application. |

4) What is the difference between structural directive and attribute directive in Angular 7?

**Structural directives** are used to alter the DOM layout by removing and adding DOM elements. These directives are far better in changing the structure of the view. Examples of Structural directives are NgFor and Nglf.

**Attribute Directives** are used as characteristics of elements. For example, a directive such as built-in NgStyle in the template Syntax guide is an attribute directive.

5) What is the difference among "declarations", "providers" and "import" in NgModule?

**Difference among declarations", "providers" and "import" in NgModule:**

* **declarations** are used to make directives (including components and pipes) from the current module available to other directives in the current module. Selectors of directives, components or pipes are only matched against the HTML if they are declared or imported.
* **providers** are used to make services and values known to DI. They are added to the root scope and they are injected to other services or directives that have them as dependency. A special case for providers is lazy loaded modules that get their own child injector. Providers of a lazy loaded module are only provided to this lazy loaded module by default (not the whole application as it is with other modules).
* **import** makes the exported declarations of other modules available in the current module.

6) What are the key components of Angular?

Key components of Angular:

**Components:** Components are the basic building blocks of angular application and used to control HTML views.

**Modules:** Modules are the set of angular basic building blocks like component, directives, services etc. An application is divided into logical pieces and each piece of code is called as "module" and used to perform a single task.

**Templates:** Templates are used to represent the views of an Angular application.

**Services:** It is used to create components which can be shared across the entire application.

**Metadata:** This can be used to add more data to an Angular class.

7) Explain the Architecture overview of Angular.

Angular is the most popular web development framework for developing mobile and web applications. It uses cross platform mobile development called IONIC that's why it is not limited to web apps only.

**There are 7 main building blocks of an Angular application:**

* Component
* Templates
* Metadata
* Data Binding
* Directives
* Services
* Dependency Injection

The basic building blocks of an Angular application are NgModules, which provide a compilation context for components. Angular app is defined by a set of NgModules and it always has at least a root module that enables bootstrapping, and many more feature modules.

* Components define Template views
* Components use services

The NgModules make developers to organize an application into connected blocks of functionality.

The NgModule properties for the minimum "AppModule" generated by the CLI are as follows:

* **Declarations:** Use to declare the application components.
* **Imports:** Every application must import BrowserModule to run the app in a browser.
* **Providers:** There are none to start.
* **Bootstrap:** This is a root AppComponent that Angular creates and inserts into the index.html host web page.

8) How would you update Angular 6 to Angular 7?

You can update Angular 6 to Angular 7 by using the following command:

1. ng update @angular/cli @angular/core

9) What is the UrlSegment Interface in Angular 7?

In Angular 7, the UrlSegment interface represents a single URL segment, constructor, properties and methods like this:

1. **class** UrlSegment {
2. constructor(path: string, parameters: {...})
3. path: string
4. parameters: {...}
5. toString(): string
6. }

The UrlSegment is a part of a URL between the two slashes and it contains a path and matrix parameters associated with the segment.

10) What is Do Bootstrap (ng Do Bootstrap) In Angular 7?

The ng Do Bootstrap is a new life-cycle hook added in Angular 7 and Do Bootstrap is an interface.

**Example**

1. //ng Do Bootstrap - Life-Cycle Hook Interface
2. classApp Module **implements** Do Bootstrap {
3. ng Do Bootstrap(appRef: ApplicationRef) {
4. appRef.bootstrap(AppComponent);
5. }
6. }

11) What are directives in Angular7?

In Angular7, directives are used to add behavior to an existing DOM element or an existing component instance.

**For Example**

1. **import** { Directive, ElementRef, Input } from '@angular/core';
2. @Directive({ selector: '[myHighlight]' })
3. export **class** HighlightDirective {
4. constructor(el: ElementRef) {
5. el.nativeElement.style.backgroundColor = 'green';
6. }
7. }

Now this directive extends HTML element behavior with a green background as below:

Highlight me!

12) What are components in Angular7?

Components are the basic building blocks of an Angular app formed like a tree structure. Components are subset of directives but unlike directives, components always have a template and only one component can be instantiated per an element in a template.

**For example:**

1. **import** { Component } from '@angular/core';
2. @Component ({
3. selector: 'my-app',
4. template: ` <div>
5. <h1>{{title}}</h1>
6. <div>Learn Angular6 with examples</div>
7. </div> `,
8. })
9. export **class** AppComponent {
10. title: string = 'Welcome to Angular world';
11. }

13) What is the difference between component and directive?

A component (@component) is a directive with a template. Some major difference between components and directives are:

|  |  |
| --- | --- |
| **Component** | **Directive** |
| If you register a component, you have to use @Component meta-data annotation | If you register a directive, you have to use @Directive meta-data annotation |
| Components are used to break up the application into smaller components. | Directives are used to design re-usable components. |
| Components are used to create UI widgets. | Directives are used to add behavior to an existing DOM element. |
| Only one component can be present per DOM element. | Many directives can be used per DOM element. |
| @View decorator or templateurl/template are mandatory | Directives don't use View. |

14) What is a template in Angular7?

A template is a HTML view where you display your data by binding controls to Angular component's properties. You can store your component's template in one of two places. You can define it inline using the template property, or you can define the template in a separate HTML file and link to it in the component metadata using the @Component decorator's templateUrl property.

**For example:**

**Using inline template with template syntax**

1. **import** { Component } from '@angular/core';
2. @Component ({
3. selector: 'my-app',
4. template: '
5. <div>
6. <h1>{{title}}</h1>
7. <div>Learn Angular</div>
8. </div>
9. '
10. })
11. export **class** AppComponent {
12. title: string = 'Hello World';
13. }

**Using separate template file such as app.component.html**

1. **import** { Component } from '@angular/core';
2. @Component ({
3. selector: 'my-app',
4. templateUrl: 'app/app.component.html'
5. })
6. export **class** AppComponent {
7. title: string = 'Hello World';
8. }

15) What is a module?

Modules are the logical boundaries in the application. An application is divided into separate modules to separate the functionalities of the application.

**For example:** app.module.ts root module declared with @NgModule decorator

1. **import** { NgModule }      from '@angular/core';
2. **import** { BrowserModule } from '@angular/platform-browser';
3. **import** { AppComponent }  from './app.component';
4. @NgModule ({
5. imports:      [ BrowserModule ],
6. declarations: [ AppComponent ],
7. bootstrap:    [ AppComponent ]
8. })
9. export **class** AppModule { }

**Here, the NgModule decorator has three options:**

* The imports option is used to import other dependent modules. The BrowserModule is required by default for any web based angular application.
* The declarations option is used to define components in the respective module.
* The bootstrap option tells Angular which Component to bootstrap in the application.

16) What are the new features added in Angular7?

Following is a list of new features added in Angular7:

* Angular7 displays an elegant look in the new update.
* It provides virtual scrolling by using the scrolling package .
* It facilitates you to use drag and drop property by using the @angular/cdk/drag-drop module.
* In Angular7, libraries make changes to itself automatically with the updated version of the Material design.
* Angular7 provides better error handling for @Output if property is not initialized.
* Angular7 provides added support for Node v10.

**Some more Angular7 features are:**

**Angular Console:** It is a downloadable console to start and run Angular projects on your local machine.

**@angular/fire:** It is a latest update on npm, and has its first stable release for Angular7.

**NativeScript:** It facilitates you to make a single project that builds for both web and installed mobile.

**StackBlitz:** StackBlitz 2.0 is now released and includes the Angular Language Service and more features like tabbed editing.

**Angular 7 Interview Questions and Answers**

**1. What is new in the Angular 7?**

Angular Elements will be enabled to support content projection. This is done with the help of web standards for custom elements.

* **Angular Material Gets Minor Updates**

Angular Material will get better display. It will give in an elegant look in the new update. Moreover, it will also add a new homepage for the material, material.io. In this, we will get tooling, design guidance, development components and stay up-to-date with the latest news.

If we are using an Angular Material v7 then we can observe a visual difference as library can make changes to itself. This is done with the updated version of the Material design.

* **Better Accessibility for Selects**

In the updated version, it will include a lot of new features to enhance accessibility for selects. It will add a new feature of the native select inside mat-form-field. It is so far better and outperformed than the mat- select.

Both the select and mat-select are available so we can choose what we want to do.

* **Virtual Scrolling**

The Component Dev Kit (CDK) is available in the market with the great virtual scrolling capabilities that the user can apply by importing the `ScrollingModule`!

<cdk-virtual-scroll-viewport itemSize=”20″>

<div \*cdkVirtualFor=”let dog of dogsArray”> {{dog}}</div>

</cdk-virtual-scroll-viewport>

* **Drag & Drop**

The CDK in the new update also now recommend to Drag & Drop. This will possess these great hallmarks.

* **Automated render as a user moves items**

It is a new feature available only in Angular 7.

* **Helper methods for reordering/transferring items in lists**

For reordering or transferring items in lists, we can use a helper method. They are moveItemInArray and transferArrayItem.

* **Enhancing Application Performance**

We will get enhanced application performance in Angular 7.

* **A safeguard has come into play for the users of Angular 7**

It gives a portent to new application builders when they are crossing the budget with their bundle size. The warning occurs on 2 MB whereas an error occurs over 5 MB. But we don’t need to frighten. We can change the limits simply in our angular.json file. The thing we must do is add in a bit about the warnings and error sizes with budget details.

“budgets”: [

{

“type”: “initial”,

“maximumWarning”: “2mb”,

“maximumError”: “5mb”

}

]

**2. What do you mean by Angular Framework?**

In simple words we can say that angular is a TypeScript-based open-source front-end platform. It is very easy to build applications with in web, mobile and desktop. The major features of this framework are declarative templates, dependency injection, end to end tooling etc., which are used to ease the development.

**3. What do you mean by TypeScript?**

TypeScript is a typed superset of JavaScript. It is created by Microsoft. It will add optional types, classes, async or await, and many other features. It will compile to plain JavaScript. Angular built entirely in TypeScript and used as a primary language.

**4. How can you install TypeScript globally?**

We can install TypeScript globally as follows

npm install -g typescript

Let us see a simple example of TypeScript usage,

function greeter(city: string)

{

return “Welcome, ” + city;

}

let user = “Chennai+”;

document.body.innerHTML = greeter(user);

**5. How will you update angular versions to latest?**

Following are the simple steps to update the angular version. They are,

* First, we have to update the Angular version globally. This is possible by inserting the latest version via the terminal: sudo npm install -g @angular/cli@latest.
* The next step is to upgrade the version locally in our project. We need to assure the altering for the new version are displayed in the package.json file ng update @angular/cli.
* When it iscompleted, upgrade all our dependencies and dev dependencies in package.json.
* To build Angular applications, Angular-devkit was introduced in Angular 6. It will need the dependency on the CLI projects.
* With all of this, we will require upgrading the version for Typescriptnpm install typescript@2.9.2 –save-dev.
* Then, we need to relocate the configuration of angular-cli.json to angular.json ng update @angular/cli and ng update @angular/core.
* Use this command: ng update @angular/material in case of Angular material is used.
* The next step is the removal of deprecated RxJS 6 features npm install -g rxjs-tslint rxjs-5-to-6-migrate -p src/tsconfig.app.json.
* When it is completed, uninstall rxjs-compat as it is not required for Angular npm uninstall –save rxjs-compat.
* Also change import {Observable} from ‘rxjs/Observable’; to import {Observable} from ‘rxjs’;
* Finally, we can start our Angular 7 application by using ng serve.

**6. Can you state the difference between AngularJS and Angular?**

|  |  |  |
| --- | --- | --- |
| **S:NO** | **AngularJS** | **Angular** |
| 1 | This will be based on MVC architecture. | This will be based on Service/Controller. |
| 2 | This will use JavaScript to build the application. | It Introduced the typescript to write the application. |
| 3 | It is Based on controller concept. | It is a component-based UI approach. |
| 4 | It is not a mobile friendly framework. | It is developed considering mobile platform. |
| 5 | Difficulty in SEO friendly application development. | Ease to create SEO friendly applications. |

**7. Can you give an example of a simple HTML document with some header information and page content?**

HTML documents are different. They will follow a basic structure of head and body. Here we are providing a simple document structure and basic tags such as DOCTYPE, html, head, title, meta, body, h1, p, etc.

**For example:**

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

<meta charset=”UTF-8″>

<meta name=”description” content=”Page description”>

</head>

<body>

<h1>Hope Tutors</h1>

<p>Angular 7 Course</p>

</body>

</html>

**8. What is your understanding about the CSS box model? Write some code to describe?**

In simple words we can say that CSS is a language which will describe how the webpages look. The CSS box model will refer to the layout and design of HTML elements. It is in box shape which will wrap around each HTML element. A box is made up of the following elements. They are,

* Content of the box
* Padding
* Border
* Margin

**Condition 1:** The same padding on all 4 sides.

padding: 25px;

**Condition 2:** padding for the top, right, bottom, left.

padding: 25px 50px 75px 100px;

**Condition 3:** Top/bottom padding 25 pixels, right/left padding 50 pixels.

padding: 25px 50px;

**9. How will you change the style of an HTML element using JavaScript?**

Let us see with an example to change the font size.

document.getElementById(“someElement”).style.fontSize = “18”;

**10. Can you write some of the code for a basic class in TypeScript with a constructor and a method?**

Here we are going to see the simple class that is created with a greeting message which can be retrieved with the greet () function.

class Greeter

{

greeting: string;

constructor(message: string)

{

this.greeting = message;

}

greet()

{

return “Hello, ” + this.greeting;

}

}

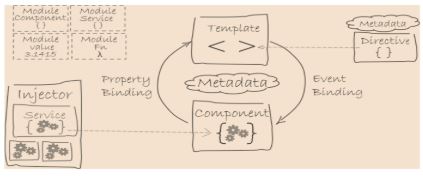
let greeter = new Greeter(“world”);

**11. What do you mean by Single Page Applications? How do they work in Angular?**

In short term Single Page Applications is SPAs. These are web applications. It will use only one HTML page. As the user interacts with the page, new content is dynamically updated on that master page. Navigation between pages happens without refreshing the whole page. Angular uses AJAX and to dynamically update HTML elements. Angular Routing can be used to make SPAs. The result is an application that feels more like a desktop app rather than a webpage.

**12. Can you draw the pictorial diagram of Angular architecture?**

The following diagram shows the architectural model of angular



**13. Can you list the key components of Angular?**

Following are the key components of angular. They are,

* **Component:** These are the basic building blocks of angular application. It is used to control HTML views.
* **Modules:** An angular module is set of angular basic building blocks. They are like component, directives, services etc. An application is divided as logical pieces. Each piece of code is called as “module”. It will perform a single task.
* **Templates:** This will represent the views of an Angular application.
* **Services:** It is used to create components. This will be shared across the entire application.
* **Metadata:** This will be used to add more data to an Angular class.

**14. What do you mean by directives?**

Directives will add the behaviour to an existing DOM element or an existing component instance.

import { Directive, ElementRef, Input } from ‘@angular/core’;

@Directive({ selector: ‘[myHighlight]’ })

export class HighlightDirective {

constructor(el: ElementRef) {

el.nativeElement.style.backgroundColor = ‘yellow’;

}

}

Now this directive extends HTML element behavior with a yellow background as below

<p myHighlight>Highlight me!</p>

**15. What do you mean by components?**

Components are the most basic UI building block of an Angular app. It will form a tree of Angular components. These components are subset of directives. Unlike these directives, components always have a template and only one component can be instantiated per an element in a template. Let’s see a simple example of Angular component

import { Component } from ‘@angular/core’;

@Component ({

selector: ‘my-app’,

template: ` <div>

<h1>{{title}}</h1>

<div>Learn Angular6 with examples</div>

</div> `,

})

export class AppComponent {

title: string = ‘Welcome to Angular world’;

}

**16. List out the differences between Component and Directive?**

|  |  |  |
| --- | --- | --- |
| **S:NO** | **Components** | **Directives** |
| 1 | To register a component, we should use @Component meta-data annotation. | To register directives, we should use @Directive meta-data annotation. |
| 2 | Components are used to create UI widgets. | Directives are used to add behavior to an existing DOM element. |
| 3 | Components are used to break up the application into smaller components | Directives are used to design re-usable components. |
| 4 | Only one component can be present per DOM element . | Many directives can be used per DOM element. |
| 5 | @View decorator or templateurl/template are mandatory. | Directive doesn’t use View. |

**17. What do you mean by template?**

In simple term we can say that a template is a HTML view. Using this we can able to display the data by binding controls to properties of an Angular component. You can store your component’s template in one of two places. You can define it inline using the template property, or you can define the template in a separate HTML file and link to it in the component metadata using the @Component decorator’s templateUrl property. Using inline template with template syntax,

import { Component } from ‘@angular/core’;

@Component ({

selector: ‘my-app’,

template: ‘

<div>

<h1>{{title}}</h1>

<div>Learn Angular</div>

</div>

‘

})

export class AppComponent {

title: string = ‘Hello World’;

}

Using separate template file such as app.component.html

import { Component } from ‘@angular/core’;

@Component ({

selector: ‘my-app’,

templateUrl: ‘app/app.component.html’

})

export class AppComponent {

title: string = ‘Hello World’;

}

**18. What do you mean by a module?**

Modules are logical boundaries in your application. The application will be divided into separate modules to separate the functionality of your application. Let us see with an example of app.module.ts root module declared with @NgModule decorator as below,

import { NgModule } from ‘@angular/core’;

import { BrowserModule } from ‘@angular/platform-browser’;

import { AppComponent } from ‘./app.component’;

@NgModule ({

imports: [ BrowserModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

The NgModule decorator has three options. They are,

* The imports option is used to import other dependent modules. The Browser Module is required by default for any web based angular application.
* The declarations option is used to define components in the respective module.
* The bootstrap option tells Angular which Component to bootstrap in the application.

**19. How will you handle the events in the angular 7?**

There are various methods to handle events in Angular 7. These are:

* Binding to user input events: You can use the Angular event binding to answer to DOM event. User input will trigger so many DOM events. It is a very effective method to get the input from the user.

For example,

<button (click)=”onClickMe()”>Click me!</button>

* Get user input from the event object: DOM will carry a cargo of the information that possibly is valuable for the components. Here is an example to show you the keyup event of an input box to obtain the user’s input after every stroke.

src/app/keyup.components.ts (template v.1)

content\_copy

template: `

<input (keyup)=”onKey($event)”>

<p>{{values}} </p>

* 3Key event filtering: Every keystroke is heard by the (keyup) event handler. The enter keys matter the most as it provides the sign of the user that he has done with the typing. The most efficient method of eliminating the noise is to look after every $event. keyCode and the action is taken only when the enter key is pressed.

**20. What are Core Dependencies of angular 7?**

There are two type of core dependencies. They are RxJS and TypeScript.

* RxJS 6.3:

RxJS version 6.3 is used by Angular 7. It has no changes in the version from Angular 6.

* TypeScript 3.1

TypeScript version 3.1 is used by Angular 7. It is the upgrade from the version2.9 of Angular 6.

**21. Explain about Bazel?**

Bazel is one of the best test tools. It is just like the Make, Maven and Gradle. It is an open-source build. Bazel utilizes readable and high-level build language. It will handle the project in a great number of languages and builds the product for many platforms. Moreover, it will support multiple users and large codebases over several repositories.

**22. How to create a decorator in Angular?**

There are two ways to register decorators in Angular. These are as follows.

* $provide.decorator.
* module.decorator.

**23. What do you mean by IVR Render? Is it supported by angular 7?**

Angular will be releasing a new kind of rendering pipeline and view engine.

The purpose of angular view engine is to translate the templates as well as components that we have written into the regular HTML and JavaScript. It is very easy for the browser to read it comfortably. Ivy is believed to be splendid for the Angular Renderer.

Yes, it is supported by Angular 7.

**24. How will you generate the class in angular 7 using CLI?**

To create a class we should use the below code:

ng generate class <name> [options]

ng g class <name> [options]

Where name refers the name of a class.

Options refer to the project name, spec value in Boolean or type of a file.

**25. Can you state the difference between Structural and attribute directives in angular?**

Structural directives are used to alter the DOM layout. This is done by removing and adding DOM elements. It is far better in changing the structure of the view. Best examples of Structural directives are NgFor and Nglf.

Attribute Directives These are being used as characteristics of elements. For example, a directive such as built-in NgStyle in the template Syntax guide is an attribute directive.

**26. What are the available lifecycle hooks?**

Angular application goes through an entire set of processes or has a lifecycle right from its initiation to the end of the application. The representation of lifecycle in pictorial representation as follows.

The description of each lifecycle method is as below,

* ngOnChanges: When the value of a data bound property changes, then this method is called.
* ngOnInit: This is called whenever the initialization of the directive/component after Angular first displays the data-bound properties happens.
* ngDoCheck: This is for the detection and to act on changes that Angular can’t or won’t detect on its own.
* ngAfterContentInit: This is called in response after Angular projects external content into the component’s view.
* ngAfterContentChecked: This is called in response after Angular checks the content projected into the component.
* ngAfterViewInit: This is called in response after Angular initializes the component’s views and child views.
* ngAfterViewChecked: This is called in response after Angular checks the component’s views and child views.
* ngOnDestroy: This is the cleanup phase just before Angular destroys the directive/component.

**27. What do you mean by data binding?**

Data binding is a core concept in Angular. It will allow to define communication between a component and the DOM. It is very easy to define interactive applications without worrying about pushing and pulling data. There are four forms of data binding. This will be divided as 3 categories which differ in the way the data is flowing.

* From the Component to the DOM: Interpolation: {{ value }}: Adds the value of a property from the component

<li>Name: {{ user.name }}</li>

<li>Address: {{ user.address }}</li>

Property binding: [property]=”value”: The value is passed from the component to the specified property or simple HTML attribute

<input type=”email” [value]=”user.email”>

* From the DOM to the Component: Event binding: (event)=”function”: When a specific DOM event happens (eg.: click, change, keyup), call the specified method in the component

<button (click)=”logout()”></button>

* Two-way binding: Two-way data binding: [(ngModel)]=”value”: Two-way data binding allows to have the data flow both ways. For example, in the below code snippet, both the email DOM input and component email property are in sync

<input type=”email” [(ngModel)]=”user.email”>

**28. What do you mean by metadata?**

Metadata is used to decorate a class. It can configure the expected behavior of the class. The metadata is represented by decorators.

* Class decorators, e.g. @Component and @NgModule

import { NgModule, Component } from ‘@angular/core’;

@Component({

selector: ‘my-component’,

template: ‘<div>Class decorator</div>’,

})

export class MyComponent {

constructor() {

console.log(‘Hey I am a component!’);

}

}

@NgModule({

imports: [],

declarations: [],

})

export class MyModule {

constructor() {

console.log(‘Hey I am a module!’);

}

}

* Property decorators Used for properties inside classes, e.g. @Input and @Output

import { Component, Input } from ‘@angular/core’;

@Component({

selector: ‘my-component’,

template: ‘<div>Property decorator</div>’

})

export class MyComponent {

@Input()

title: string;

}

* Method decorators Used for methods inside classes, e.g. @HostListener

import { Component, HostListener } from ‘@angular/core’;

@Component({

selector: ‘my-component’,

template: ‘<div>Method decorator</div>’

})

export class MyComponent {

@HostListener(‘click’, [‘$event’])

onHostClick(event: Event) {

// clicked, `event` available

}

}

* Parameter decorators Used for parameters inside class constructors, e.g. @Inject

import { Component, Inject } from ‘@angular/core’;

import { MyService } from ‘./my-service’;

@Component({

selector: ‘my-component’,

template: ‘<div>Parameter decorator</div>’

})

export class MyComponent {

constructor(@Inject(MyService) myService) {

console.log(myService); // MyService

}

}

**29. What do you mean by angular CLI?**

In full form we can say that angular CLI is Command Line Interface. It is a command line interface to scaffold and build angular apps using nodejs style (commonJs) modules. We need to install using below npm command,

npm install @angular/cli@latest

Following are the few commands which will come handy while creating angular projects.

Creating New Project: ng new

Generating Components, Directives & Services: ng generate/g the different types of commands are,

* ng generates class my-new-class: add a class to your application.
* ng generates component my-new-component: add a component to your application.

1. ng generates directive my-new-directive: add a directive to your application.
2. ng generates enum my-new-enum: add an enum to your application.
3. ng generates module my-new-module: add a module to your application.
4. ng generates pipe my-new-pipe: add a pipe to your application.
5. ng generates service my-new-service: add a service to your application.

* Running the Project: ng serve.

**30. State the difference between constructor and ngOnInit?**

TypeScript classes has a default method called constructor. It is normally used for the initialization purpose. Whereas ngOnInit method is specific to Angular, especially used to define Angular bindings. Even though constructor getting called first, it is preferred to move all our Angular bindings to ngOnInit method. In order to use ngOnInit, we need to implement OnInit interface as below,

export class App implements OnInit{

constructor(){

//called first time before the ngOnInit()

}

ngOnInit(){

//called after the constructor and called after the first ngOnChanges()

}

}

**31. What do you mean by service?**

A service is used when a common functionality needs to be provided to various modules. Services will allow for greater separation of concerns for our application and better modularity by allowing you to extract common functionality out of components. Let us create a repoService which can be used across components,

import { Injectable } from ‘@angular/core’;

import { Http } from ‘@angular/http’;

@Injectable({ // The Injectable decorator is required for dependency injection to work

// providedIn option registers the service with a specific NgModule

providedIn: ‘root’, // This declares the service with the root app (AppModule)

})

export class RepoService{

constructor(private http: Http){

}

fetchAll(){

return this.http.get(‘https://api.github.com/repositories’);

}

}

**32. What is mean by dependency injection in Angular?**

Dependency injection in short form DI. It is an important application design pattern in which a class asks for dependencies from external sources rather than creating them itself. Angular comes with its own dependency injection framework. It is used for resolving dependencies. So, we can have our services depend on other services throughout our application.

**33. What is the basic syntax of a Decorator in Angular?**

ANS: @() with optional parameters.

**34. For what [(ngModel)] will be used for?**

In simple words we can tell this as two-way data binding.

**35. Can you tell me the basic parts of an Angular application?**

Following are some of the basic parts of the Angular. They are,

* Modules.
* Component.
* Data Binding.
* Template.
* Directives.
* Dependency Injection.
* Services.
* Routing.

**36. Tell me the purpose of async pipe?**

The AsyncPipe is about subscribing to an observable or promise and returns the latest value it has emitted. Whenever a new value is emitted then the pipe marks the component to be checked for changes. It will take a time to observable which continuously updates the view for every 2 seconds with the current time.

@Component({

selector: ‘async-observable-pipe’,

template: `<div><code>observable|async</code>:

Time: {{ time | async }}</div>`

})

export class AsyncObservablePipeComponent {

time = new Observable(observer =>

setInterval(() => observer.next(new Date().toString()), 2000)

);

}

**37. Can you tell me the option to choose between inline and external template file?**

We can store your component’s template in one of two places. We can define it inline using the template property. we can also define the template in a separate HTML file and link to it in the component metadata. This is done by using the @Component decorator’s templateUrl property.

The choice between inline and separate HTML is a matter of taste, circumstances, and organization policy. But normally we use inline template for small portion of code and external template file for bigger views. By default, the Angular CLI generates components with a template file. But you can override that with the below command,

ng generate component hero -it

**38. Can you tell me the purpose of ngFor directive?**

We use Angular ngFor directive in the template. This is to display each item in the list. For example, here we iterate over list of users,

<li \*ngFor=”let user of users”>

{{ user }}

</li>

The user variable in the ngFor double-quoted instruction is a template input variable.

**39. Can you tell me the purpose of ngIf directive?**

In sometimes an app needs to display a view or a portion of a view only under specific circumstances. The Angular ngIf directive inserts or removes an element. It will be based on a true or false condition. Best example to display a message if the user age is more than 18,

<p \*ngIf=”user.age > 18″>You are not eligible for student pass!</p>

**40. What will happen if you use script tag inside template?**

Angular recognizes the value as unsafe and automatically sanitizes. It will remove the <script> tag but keeps safe content such as the text content of the <script> tag. This way it eliminates the risk of script injection attacks. If you still use it then it will be ignored. A warning appears in the browser console. Best example of innerHtml property binding which causes XSS vulnerability,

export class InnerHtmlBindingComponent {

// For example, a user/attacker-controlled value from a URL.

htmlSnippet = ‘Template <script>alert(“0wned”)</script> <b>Syntax</b>’;

}

**41. What do you mean by interpolation?**

Interpolation is a special syntax. Using this Angular will convert into property binding. It’s a convenient alternative to property binding. It is represented by double curly braces ({{}}). The text between the braces is often the name of a component property. Angular replaces that name with the string value of the corresponding component property. The best example,

<h3>

{{title}}

<img src=”{{url}}” style=”height:30px”>

</h3>

In the example above, Angular evaluates the title and url properties and fills in the blanks, first displaying a bold application title and then a URL.

**42. List some advantages of using Angular framework for building web applications?**

Following are some of the advantages of using angular framework for building web applications. They are,

* It will save lot of time for developers. This is possible by doing a lot of the work for them like writing tedious DOM manipulation tasks.
* TypeScript and the Angular framework will allow us to catch errors much earlier.
* In many cases has faster performance than traditional web development techniques.
* Can give web apps the feel of a desktop application.
* It separates out the code of an application to make it easier for multiple developers to work on an app and easier to test.
* More consistent code base which will be easy to maintain.
* Big developer community.

**43. What function will be called when an object is created in TypeScript? What is it basic syntax in TypeScript code?**

The constructor function will be called.

It‘s syntax is:  Constructor(){}

**44. How will you interact between Parent and Child components in angular?**

When passing data from Parent to Child component, we can use the @Input decorator in the Child component. Similarly, when passing data from Child to Parent component, we can use the @Output decorator in the Child component.

**45. Can you give me an example usage of ngFor for displaying all items from an array ’Items‘ in a list with <li>?**

Following will be the best example.

<li \*ngFor=”let item of Items”>

{{item}}

</li>

**46. List the main difference between constructor and ngOnInit?**

The constructor is a feature of the class. It is not an Angular. The main difference is that Angular will launch ngOnInit. Once it is finished configuring the component, then it is a signal through which the @Input() and other banding properties and decorated properties are available in ngOnInit, but are not defined within the constructor by design.

**47. What are HTTP Interceptors?**

Interceptor is just a fancy word for a function that receives requests or responses before they are processed or sent to the server. You should use interceptors if you want to pre-process many types of requests in one way. For example, you need to set the authorization header Bearer for all requests

token.interceptor.ts

import { Injectable } from ‘@angular/core’;

import { HttpInterceptor, HttpRequest, HttpHandler, HttpEvent } from ‘@angular/common/http’;

import { Observable } from ‘rxjs/Observable’;

@Injectable()

export class TokenInterceptor implements HttpInterceptor {

public intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

const token = localStorage.getItem(‘token’) as string;

if (token) {

req = req.clone({

setHeaders: {

‘Authorization’: `Bearer ${token}`

}

});

}

return next.handle(req);

}

}

And register the interceptor as singleton in the module providers:

app.module.ts

import { NgModule } from ‘@angular/core’;

import { BrowserModule } from ‘@angular/platform-browser’;

import { HTTP\_INTERCEPTORS } from ‘@angular/common/http’;

import { AppComponent } from ‘./app.component’;

import { TokenInterceptor } from ‘./token.interceptor’;

@NgModule({

imports: [

BrowserModule

],

declarations: [

AppComponent

],

bootstrap: [AppComponent],

providers: [{

provide: HTTP\_INTERCEPTORS,

useClass: TokenInterceptor,

multi: true // < – – – – an array of interceptors can be registered

}]

})

export class AppModule {}

**48. How many Change Detectors can there be in the whole application?**

Each component has its own ChangeDetector. All Change Detectors are inherited from AbstractChangeDetector.

**49. List out some of the important practices to secure an Angular application?**

Following are some of the important practices to secure an angular application. They are,

* Check that all requests come from within your own web app and not external websites.
* Sanitize all input data.
* Use Angular template instead of DOM APIs.
* Content Security Policies.
* Validate all data with server-side code.
* Use an offline template compiler.
* Avoid including external URLs in your application.
* Make JSON responses non-executable.
* Keep all libraries and frameworks up-to-date.

**50. What do you mean by ViewEncapsulation and how many ways are there do to do it in Angular?**

In simple words, ViewEncapsulation determines whether the styles defined in a component will affect the entire application or not. Angular supports 3 types of ViewEncapsulation. They are

* Emulated – Styles used in another HTML spread to the component.
* Native – Styles used in another HTML doesn’t spread to the component.
* None – Styles defined in a component are visible to all components of the application.

1. **Question 1. What Is Angular?**

**Answer :**

* + Angular is a most popular web development framework for developing mobile apps as well as desktop applications.
  + Angular framework is also utilized in the cross platform mobile development called IONIC and so it is not limited to web apps only.
  + Angular is an open source framework written and maintained by angular team at Google and the Father of Angular is Misko Hevery.
  + Angular is written in TypeScript and so it comes with all the capabilities that typescript offers.

1. **Question 2. What Is Architecture Overview Of Angular?**

**Answer :**

**Angular Architecture Overview** :

Angular is a most popular web development framework for developing mobile apps as well as desktop applications.

Angular framework is also utilized in the cross platform mobile development called IONIC and so it is not limited to web apps only.

Angular is an open source framework written and maintained by angular team at Google and the Father of Angular is Misko Hevery.

The bootstrapping process creates the components listed in the bootstrap array and inserts each one into the browser (DOM)

 you can identify the seven main building blocks of an Angular Application.

* + Component
  + Templates
  + Metadata
  + Data Binding
  + Directives
  + Services
  + Dependency Injection

 The basic building blocks of an Angular application are NgModules, which provide a compilation context for components.

 Angular app is defined by a set of NgModules and it always has at least a root module that enables bootstrapping, and many more feature modules.

* + Components define Template views
  + Components use services

The Angular Module (NgModules) helps us to organize an application into connected blocks of functionality.

The NgModule properties for the minimum “AppModule” generated by the CLI which are follow as -

**Declarations —** Use to declare the application components.

**Imports —**Every application must import BrowserModule to run the app in a browser.

**Providers —** There are none to start.

**Bootstrap —** This is a root AppComponent that Angular creates and inserts into the index.html host web page.

[HTML 5 Interview Questions](https://www.wisdomjobs.com/e-university/html-5-interview-questions.html)

* **Question 3. How To Update Angular 6 To Angular 7?**

**Answer :**

For updating Angular 6 to Angular 7,

**you should use a command:**

**ng update @angular/cli @angular/core**

* **Question 4. What Is Urlsegment Interface In Angular 7?**

**Answer :**

**UrlSegment Interface :**

UrlSegment interface represents a single URL segment and the constructor, properties, and methods look like below UrlSegment class i.e.

class UrlSegment {

constructor(path: string, parameters: {...})

path: string

parameters: {...}

toString(): string

}

 The UrlSegment is a part of a URL between the two slashes and it contains a path and matrix parameters associated with the segment.

[HTML 5 Tutorial](https://www.wisdomjobs.com/e-university/html-5-tutorial-207.html)

* **Question 5. What Is Angular Compatibility Compiler (ngcc) In Angular 7?**

**Answer :**

**The ngcc Angular node\_module compatibility compiler :**

* + The ngcc is a tool which "upgrades" node\_module compiled with non-ivy ngc into ivy compliant format.
  + This compiler will convert node\_modules compiled with Angular Compatibility Compiler (ngcc), into node\_modules which appear to have been compiled with TSC compiler transformer (ngtsc) and this compiler conversions will allow such “legacy” packages to be used by the Ivy rendering engine.
  + TSC transformer which removes and converts @Pipe, @Component, @Directive and @NgModule to the corresponding definePipe, defineComponent, defineDirective and defineInjector.

[Drupal Interview Questions](https://www.wisdomjobs.com/e-university/drupal-interview-questions.html)

* **Question 6. What Is Do Bootstrap (ng Do Bootstrap ) In Angular 7?**

**Answer :**

**Do Bootstrap Interface** :

Angular 7 added a new life-cycle hook that is called ng Do Bootstrap and an interface that is called Do Bootstrap.

**Example:**

//ng Do Bootstrap - Life-Cycle Hook Interface

classApp Module implements Do Bootstrap {

 ng Do Bootstrap(appRef: ApplicationRef) {

appRef.bootstrap(AppComponent);

  }

}

* **Question 7. What Is Xmb?**

**Answer :**

The XMB is basically a key value pairs with no deeper structure. It does have a mechanism for named placeholders, with descriptions and examples. The  messages for any given other language must be correspond 1:1.

[Drupal Tutorial](https://www.wisdomjobs.com/e-university/drupal-tutorial-545.html) [Angular JS Interview Questions](https://www.wisdomjobs.com/e-university/angular-js-interview-questions.html)

* **Question 8. What Is Xmb Placeholders?**

**Answer :**

The placeholders have one example tag () and a text node. The text node will be used as the original value from the placeholder, while the example will represent a dummy value.

* **Question 9. What's New In Angular 7?**

**Answer :**

The major release and expanding to the entire platform including-

* + Core framework,
  + Angular Material,
  + CLI

[Ext JS Interview Questions](https://www.wisdomjobs.com/e-university/ext-js-interview-questions.html)

* **Question 10. What Is Ivy Rendering Engine In Angular 7?**

**Answer :**

**Ivy Rendering Engine :**

The Ivy rendering engine is a new backwards-compatible Angular renderer main focused on the following.

* + Speed Improvements
  + Size Reduction
  + Increased Flexibility

The template functions for creating dynamically views are no longer nested functions inside each other.

Now we use for loops that are nested inside other loops.

**Example:**

functionAppComponent(rf: RenderFlags, ctx: AppComponent) {

functionulTemplateFun(rf1: RenderFlags, ctx0: any) {

functionliTemplateFun(rf1: RenderFlags, ctx1: any) {...}

  }

}

[Ext JS Tutorial](https://www.wisdomjobs.com/e-university/ext-js-tutorial-1140.html)

* **Question 11. What Is Key Value Pipe In Angular 7?**

**Answer :**

**Key Value Pipe:**

Change you object into an array of key value pairs that output array will be ordered by keys.

 By default it will be by Unicode point value.

**Syntax:**

 {{your input expression | key value [:compareFn] }}

[Javascript Advanced Interview Questions](https://www.wisdomjobs.com/e-university/javascript-advanced-interview-questions.html)

* **Question 12. What Are The Core Dependencies Of Angular 7?**

**Answer :**

**Core Dependencies:**

There are two types of core dependencies: RxJS and TypeScript.

**RxJS 6.3:**

RxJS version 6.3 is used by Angular 7. It has no changes in the version from Angular 6

**TypeScript 3.1:**

TypeScript version 3.1 is used by Angular 7. It is the upgrade from the version2.9 of Angular 6.

[HTML 5 Interview Questions](https://www.wisdomjobs.com/e-university/html-5-practice-tests-207-327405)

* **Question 13. Explain Bazel?**

**Answer :**

Bazel is a test tool just like the Make, Maven and Gradle and it is an open-source build. Bazel utilizes the readable and high-level build language. It handles the project in a great number of languages and builds the product for a large number of platforms. Moreover, it supports multiple users and large codebases over several repositories.

[Javascript Advanced Tutorial](https://www.wisdomjobs.com/e-university/javascript-advanced-tutorial-1214.html)

* **Question 14. How To Generate A Class In Angular 7 Using Cli ?**

**Answer :**

**Create a class using below code:**

ng generate class [options]  
ng g class [options]  
Whose name refers the name of a class.  
Options refer to the project name, spec value in Boolean or type of a file.

* **Question 15. How Can You Create A Decorator In Angular?**

**Answer :**

There are two ways to register decorators in Angular.

**These are:**

* + $provide.decorator
  + module.decorator

[Angular Material Interview Questions](https://www.wisdomjobs.com/e-university/angular-material-interview-questions.html)

* **Question 16. How Can You Handle Events In Angular 7?**

**Answer :**

There are various methods to handle events in Angular 7.

**These are:**

1. Binding to user input events: You are able to use the Angular event binding to answer to DOM event. User input triggers so many DOM events. It is a very effective method to get the input from the user.

**For example:**

<button (click)="onClickMe()">Click me!</button>

2. Get user input from the event object: DOM carries a cargo of the information that possibly is valuable for the components. Here is an example to show you the keyup event of an input box to obtain the user's input after every stroke

**Example**:

src/app/keyup.components.ts (template v.1)

content\_copy

template: `

<input (keyup)="onKey($event)">

<p>{{values}} </p>

3. Key event filtering: Every keystroke is heard by the (keyup) event handler. The enter keys matter the most as it provides the sign of the user that he has done with the typing. The most efficient method of eliminating the noise is to look after every $event.keyCode and the action is taken only when the enter key is pressed.

[Angular Material Tutorial](https://www.wisdomjobs.com/e-university/angular-material-tutorial-1217.html)

* **Question 17. What Is The Difference Between Structural And Attribute Directives In Angular?**

**Answer :**

**Structural directives:**

These are used to alter the DOM layout by removing and adding DOM elements. It is far better in changing the structure of the view. Examples of Structural directives are NgFor and Ngif.

**Attribute Directives:**

These are being used as characteristics of elements. For example, a directive such as built-in NgStyle in the template Syntax guide is an attribute directive.

[1. What's new in Angular 7?](https://www.onlineinterviewquestions.com/angular-7-interview-questions/" \l "collapseUnfiled1)

**Angular Elements** is enabled to support content projection with the help of web standards for custom elements.

* **Angular Material Gets Minor Updates**

Angular Material got better in the display that gives it an elegant look in the new update. Moreover, it also added a new homepage for the material, material.io. In this, you get tooling, design guidance, development components and stay up-to-date with the latest news.

If you are using an Angular Material v7 then you observe a visual difference as library make changes to itself with the updated version of the Material Design.

* **Better Accessibility for Selects**

In the updated version, it includes a lot of new features to enhance accessibility for selects. It adds a new feature of the native select inside mat-form-field. It is far better and outperformed than the mat- select.

Both the select and mat-select are available so you can choose what you want to do.

* **Virtual Scrolling**

The Component Dev Kit (CDK) is available in the market with the great virtual scrolling capabilities that the user can apply by importing the `ScrollingModule`!

<cdk-virtual-scroll-viewport itemSize="20">

<div \*cdkVirtualFor="let dog of dogsArray"> {{dog}}</div>

</cdk-virtual-scroll-viewport>

* **Drag & Drop**

The CDK in the new update also now recommends Drag & Drop, which possess these great hallmarks:

* **Automated render as a user moves items**

It is new feature available only in Angular 7

* **Helper methods for reordering/transferring items in lists**

For reordering or transferring items in lists, you can use a helper method: moveItemInArray and transferArrayItem

* **Enhancing Application Performance**

You will get enhanced application performance in Angular 7

* **A safeguard has come into play for the users of Angular 7**

It gives a portent to new application builders when they are crossing the budget with their bundle size. The warning occurs on 2 MB whereas an error occurs over 5 MB. But you don't need to frighten. You can change the limits simply in your angular.json file. The thing you have to do is add in a bit about the warnings and error sizes with budget details.

"budgets": [

{

"type": "initial",

"maximumWarning": "2mb",

"maximumError": "5mb"

}

]

[2. What is IVY Renderer? Is it supported by Angular 7?](https://www.onlineinterviewquestions.com/angular-7-interview-questions/" \l "collapseUnfiled2)

**Angular** will be releasing a new kind of rendering pipeline and view engine.

The purpose of angular view engine is to translate the templates and components that we have written into the regular HTML and JavaScript so it is easy for the browser to read it comfortably. Ivy is believed to be splendid for the Angular Renderer.

Yes, it is supported by Angular 7.

[3. What are the Core Dependencies of Angular 7?](https://www.onlineinterviewquestions.com/angular-7-interview-questions/" \l "collapseUnfiled3)

**Core Dependencies  of Angular 7**

There are two types of core dependencies: RxJS and TypeScript.

**RxJS 6.3**

RxJS version 6.3 is used by Angular 7. It has no changes in the version from Angular 6

**TypeScript 3.1**

TypeScript version 3.1 is used by Angular 7. It is the upgrade from the version2.9 of Angular 6.

[4. How to update Angular 4,5, 6 to Angular 7?](https://www.onlineinterviewquestions.com/angular-7-interview-questions/" \l "collapseUnfiled4)

* First of all, you need to update the **Angular version** globally by inserting the latest version via the terminal: sudo npm install -g @angular/cli@latest.
* The next step is to upgrade the version locally in your project and need to assure the altering for the new version are displayed in the package.json file ng update @angular/cli
* When it is done, upgrade all your dependencies and dev dependencies in package.json
* To build Angular applications, Angular-devkit was introduced in Angular 6 that needs the dependency on the CLI projects.
* With all of this, you'll also require to upgrade the version for Typescriptnpm install typescript@2.9.2 --save-dev
* Then, you need to relocate the configuration of angular-cli.json to angular.json ng update @angular/cli and ng update @angular/core.
* Use this command: ng update @angular/material in case of Angular material is used.
* The next step is the removal of deprecated RxJS 6 features npm install -g rxjs-tslint rxjs-5-to-6-migrate -p src/tsconfig.app.json
* When it is done, uninstall rxjs-compat as it is not required for Angular npm uninstall --save rxjs-compat
* Also change import { Observable } from 'rxjs/Observable'; to import { Observable } from 'rxjs';
* Finally, you are able to start your Angular 7 application by using ng serve

[5. How to generate a class in Angular 7 using CLI?](https://www.onlineinterviewquestions.com/angular-7-interview-questions/" \l "collapseUnfiled5)

**Create a class in ANgular 7 using below code:**

ng generate **class** <name> [options]

ng g **class** <name> [options]

Where name refers to the name of a class.

Options refer to the project name, spec value in Boolean or type of a file

[6. How can you handle events in Angular 7?](https://www.onlineinterviewquestions.com/angular-7-interview-questions/" \l "collapseUnfiled6)

There are various methods to handle **events in Angular 7**. These are:

**1. Binding to user input events:** You are able to use the Angular event binding to answer to DOM event. User input triggers so many DOM events. It is a very effective method to get input from the user.

For example,

<button (click)="**onClickMe**()">Click me!</button>

**2. Get user input from the event object:** DOM carries a cargo of the information that possibly is valuable for the components. Here is an example to show you the keyup event of an input box to obtain the user's input after every stroke

src/app/keyup.components.ts (template v.1)

content\_copy

template: `

<input (keyup)="onKey($event)">

<p>{{values}} </p>

**3. Key event filtering:** Every keystroke is heard by the (keyup) event handler. The enter keys matter the most as it provides the sign of the user that he has done with the typing. The most efficient method of eliminating the noise is to look after every $event.keyCode and the action is taken only when the enter key is pressed.

[7. Explain Bazel?](https://www.onlineinterviewquestions.com/angular-7-interview-questions/" \l "collapseUnfiled7)

Bazel is a test tool just like the Make, [Maven](https://www.onlineinterviewquestions.com/maven-interview-questions/) and Gradle and it is an open-source build. Bazel utilizes readable and high-level build language. It handles the project in a great number of languages and builds the product for a large number of platforms. Moreover, it supports multiple users and large codebases over several repositories.

[8. What is the difference between Structural and Attribute directives in Angular?](https://www.onlineinterviewquestions.com/angular-7-interview-questions/" \l "collapseUnfiled8)

**Structural directives** are used to alter the DOM layout by removing and adding DOM elements. It is far better in changing the structure of the view. Examples of Structural directives are NgFor and Nglf.

**Attribute Directives** These are being used as characteristics of elements. For example, a directive such as built-in NgStyle in the template Syntax guide is an attribute directive.

[9. How can you create a decorator in Angular?](https://www.onlineinterviewquestions.com/angular-7-interview-questions/" \l "collapseUnfiled9)

There are two ways to register decorators in Angular. These are:

* $provide.decorator, and
* module.decorator

Q1: What are pipes? Give me an example.

Topic: **Angular**  
Difficulty: ⭐

A **pipe** takes in data as input and transforms it to a desired output. You can chain pipes together in potentially useful combinations. You can write your own custom pipes. Angular comes with a stock of pipes such as DatePipe, UpperCasePipe, LowerCasePipe, CurrencyPipe, and PercentPipe.

Consider:

<p>The hero's birthday is {{ birthday | date }}</p>

In this page, you'll use pipes to transform a component's birthday property into a human-friendly date.

🔗 **Source:** angular.io  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q2: What is the minimum definition of a component?

Topic: **Angular**  
Difficulty: ⭐⭐

The absolute minimal configuration for a @Component in Angular is a template. Both template properties are set to optional because you have to define either template or templateUrl.

When you don't define them, you will get an exception like this:

No template specified for component 'ComponentName'

A selector property is not required, as you can also use your components in a route.

🔗 **Source:** stackoverflow.com  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q3: What's the difference between an Angular component and module?

Topic: **Angular**  
Difficulty: ⭐⭐

*Components* control views (html). They also communicate with other components and services to bring functionality to your app.

*Modules* consist of one or more components. They do not control any html. Your modules declare which components can be used by components belonging to other modules, which classes will be injected by the dependency injector and which component gets bootstrapped. Modules allow you to manage your components to bring modularity to your app.

🔗 **Source:** stackoverflow.com  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q4: How can I select an element in a component template?

Topic: **Angular**  
Difficulty: ⭐⭐

You can get a handle to the DOM element via ElementRef by injecting it into your component's constructor:

constructor(myElement: ElementRef) { ... }

🔗 **Source:** medium.com  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

|  |  |
| --- | --- |
| All Angular Q&A |  |

Q5: What is an observer?

Topic: **Angular**  
Difficulty: ⭐⭐

Observer is an interface for a consumer of push-based notifications delivered by an Observable. It has below structure,

interface Observer<T> {

closed?: boolean;

next: (value: T) => void;

error: (err: any) => void;

complete: () => void;

}

A handler that implements the Observer interface for receiving observable notifications will be passed as a parameter for observable as below,

myObservable.subscribe(myObserver);

**Note:** If you don't supply a handler for a notification type, the observer ignores notifications of that type.

🔗 **Source:** github.com/sudheerj  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

|  |  |
| --- | --- |
| Create Full Stack Resume |  |

Q6: What is an observable?

Topic: **Angular**  
Difficulty: ⭐⭐

An Observable is a unique Object similar to a Promise that can help manage async code. Observables are not part of the JavaScript language so we need to rely on a popular Observable library called RxJS. The observables are created using new keyword. Let see the simple example of observable,

import { Observable } from 'rxjs';

const observable = new Observable(observer => {

setTimeout(() => {

observer.next('Hello from a Observable!');

}, 2000);

});`

🔗 **Source:** github.com/sudheerj  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

|  |  |
| --- | --- |
| offline\_bolt | Trending Dev Jobs |

[MEAN Stack Dev @ Fast Growing Startup!](https://buzzybooth.workable.com/jobs/1056651) , **90K-130K** wifi Full-time

**#MongoDB** **#AngularJS** **#Node.js**  location\_on Los Angeles, CA - BuzzyBooth

Q7: What is TestBed?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

The **Angular Test Bed (ATB)** is a higher level *Angular Only* testing framework that allows us to easily test behaviours that depend on the Angular Framework.

We still write our tests in Jasmine and run using Karma but we now have a slightly easier way to create components, handle injection, test asynchronous behaviour and interact with our application.

The TestBed creates a dynamically-constructed Angular test module that emulates an Angular @NgModule.

🔗 **Source:** angular.io  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

|  |  |
| --- | --- |
| All Angular Q&A |  |

Q8: What is Redux and how does it relate to an Angular app?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

**Redux** is a way to manage application state and improve maintainability of asynchronicity in your application by providing a single source of truth for the application state, and a unidirectional flow of data change in the application. ngrx/store is one implementation of Redux principles.

🔗 **Source:** github.com/WebPredict  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q9: What are the Core Dependencies of Angular 7?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

There are two types of core dependencies, RxJS and TypeScript.

* **RxJS 6.3** - RxJS version 6.3 is used by Angular 7. It has no changes in the version from Angular 6
* **TypeScript 3.1** - TypeScript version 3.1 is used by Angular 7. It is the upgrade from the version 2.9 of Angular 6.

🔗 **Source:** onlineinterviewquestions.com  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

|  |  |
| --- | --- |
| Create Full Stack Resume |  |

Q10: Why Incremental DOM Has Low Memory Footprint?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

Virtual DOM creates a whole tree from scratch every time you rerender.

Incremental DOM, on the other hand, doesn’t need any memory to rerender the view if it doesn’t change the DOM. We only have to allocate the memory when the DOM nodes are added or removed. And the size of the allocation is proportional to the size of the DOM change.

🔗 **Source:** blog.nrwl.io  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

|  |  |
| --- | --- |
| All Angular Q&A |  |

Q11: What are the ways to control AOT compilation?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

You can control your app compilation in two ways

1. By providing template compiler options in the tsconfig.json file
2. By configuring Angular metadata with decorators

🔗 **Source:** github.com/sudheerj  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

|  |  |
| --- | --- |
| offline\_bolt | Trending Dev Jobs |

[MEAN Stack Dev @ Fast Growing Startup!](https://buzzybooth.workable.com/jobs/1056651) , **90K-130K** wifi Full-time

**#Angular** **#MongoDB** **#AngularJS**  location\_on Los Angeles, CA - BuzzyBooth

Q12: What is activated route?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

**ActivatedRoute** contains the information about a route associated with a component loaded in an outlet. It can also be used to traverse the router state tree. The ActivatedRoute will be injected as a router service to access the information. In the below example, you can access route path and parameters,

@Component({

...

})

class MyComponent {

constructor(route: ActivatedRoute) {

const id: Observable < string > = route.params.pipe(map(p => p.id));

const url: Observable < string > = route.url.pipe(map(segments => segments.join('')));

// route.data includes both `data` and `resolve`

const user = route.data.pipe(map(d => d.user));

}

}

🔗 **Source:** github.com/sudheerj  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q13: What is router outlet?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

The **RouterOutlet** is a directive from the router library and it acts as a placeholder that marks the spot in the template where the router should display the components for that outlet. Router outlet is used as a component,

<router-outlet></router-outlet>

<!-- Routed components go here -->

🔗 **Source:** github.com/sudheerj  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

|  |  |  |
| --- | --- | --- |
| Create Full Stack Resume | |  |
| All Angular Q&A |  |  |

Q14: What are the utility functions provided by RxJS?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

The RxJS library also provides below utility functions for creating and working with observables.

1. Converting existing code for async operations into observables
2. Iterating through the values in a stream
3. Mapping values to different types
4. Filtering streams
5. Composing multiple streams

🔗 **Source:** github.com/sudheerj  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q15: What is multicasting?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

**Multi-casting** is the practice of broadcasting to a list of multiple subscribers in a single execution. Let's demonstrate the multi-casting feature,

var source = Rx.Observable.from([1, 2, 3]);

var subject = new Rx.Subject();

var multicasted = source.multicast(subject);

// These are, under the hood, `subject.subscribe({...})`:

multicasted.subscribe({

next: (v) => console.log('observerA: ' + v)

});

multicasted.subscribe({

next: (v) => console.log('observerB: ' + v)

});

// This is, under the hood, `s

🔗 **Source:** github.com/sudheerj  
💡 **Don't miss:** [108 More Angular Interview Questions & Answers](https://www.fullstack.cafe/Angular)

Q16: What is subscribing?

Topic: **Angular**  
Difficulty: ⭐⭐⭐

An Observable instance begins publishing values only when someone subscribes to it. So you need to subscribe by calling the **subscribe()** method of the instance, passing an observer object to receive the notifications.

Let's take an example of creating and subscribing to a simple observable, with an observer that logs the received message to the console.

Creates an observable sequence of 5 integers, starting from 1

const source = range(1, 5);

// Create observer object

const myObserver = {

next: x => console.log('Observer got a next value: ' + x),

error: err => console.error('Observer got an error: ' + err),

complete: () => console.log('Observer got a complete notification'),

};

// Execute with the observer object and Prints out each item

myObservable.subscribe(myObserver);

// => Observer got a next value: 1

// => Observer got a next value: 2

// => Observer got a next value: 3

// => Observer got a next value: 4

// => Observer got a next value: 5

// => Observer got a complete notification

Angular 7 Interview Questions



A list of top frequently asked **Angular 7 Interview Questions** and answers are given below.

1) What is Angular 7? How is it different from AngularJS?

Angular7 is the latest and recent version of Angular. AngularJS was the first version of Angular which is also known as Angular 1.0.

Angular7 is the complete rewrite of the Angular1.0. It supports two-way data binding, and some other features like ng update, ng add, Angular Elements, Angular Material + CDK Components, Angular Material Starter Components, CLI Workspaces, Library Support, Tree Shakable Providers, Animations Performance Improvements, and RxJS v6 etc.

2) What is Angular framework?

Angular is a TypeScript-based open-source web framework and a platform. It is used to build web/ mobile and desktop applications.

**Main features of this framework are:** Declarative templates, dependency injection, end to end tooling etc. These features make web development easy in Angular.

3) What is the difference between AngularJS and Angular?

Angular is a complete rewrite of AngularJS. It is a component-based framework in which an application is a tree of individual components.

**Difference between AngularJS and Angular:**

|  |  |
| --- | --- |
| **AngularJS** | **Angular** |
| AngularJS is based on MVC architecture. | Angular is based on Service/Controller. |
| It uses JavaScript to build the application. | It uses TypeScript to build the application. |
| It follows controller concept. | It follows Component based UI approach. |
| It is not a mobile-friendly framework. | It is a mobile friendly framework. |
| It is very difficult to create a SEO friendly application in AngularJS. | By using Angular, you can easily create a SEO friendly application. |

4) What is the difference between structural directive and attribute directive in Angular 7?

**Structural directives** are used to alter the DOM layout by removing and adding DOM elements. These directives are far better in changing the structure of the view. Examples of Structural directives are NgFor and Nglf.

**Attribute Directives** are used as characteristics of elements. For example, a directive such as built-in NgStyle in the template Syntax guide is an attribute directive.

5) What is the difference among "declarations", "providers" and "import" in NgModule?

**Difference among declarations", "providers" and "import" in NgModule:**

* **declarations** are used to make directives (including components and pipes) from the current module available to other directives in the current module. Selectors of directives, components or pipes are only matched against the HTML if they are declared or imported.
* **providers** are used to make services and values known to DI. They are added to the root scope and they are injected to other services or directives that have them as dependency. A special case for providers is lazy loaded modules that get their own child injector. Providers of a lazy loaded module are only provided to this lazy loaded module by default (not the whole application as it is with other modules).
* **import** makes the exported declarations of other modules available in the current module.

6) What are the key components of Angular?

Key components of Angular:

**Components:** Components are the basic building blocks of angular application and used to control HTML views.

**Modules:** Modules are the set of angular basic building blocks like component, directives, services etc. An application is divided into logical pieces and each piece of code is called as "module" and used to perform a single task.

**Templates:** Templates are used to represent the views of an Angular application.

**Services:** It is used to create components which can be shared across the entire application.

**Metadata:** This can be used to add more data to an Angular class.

7) Explain the Architecture overview of Angular.

Angular is the most popular web development framework for developing mobile and web applications. It uses cross platform mobile development called IONIC that's why it is not limited to web apps only.

**There are 7 main building blocks of an Angular application:**

* Component
* Templates
* Metadata
* Data Binding
* Directives
* Services
* Dependency Injection

The basic building blocks of an Angular application are NgModules, which provide a compilation context for components. Angular app is defined by a set of NgModules and it always has at least a root module that enables bootstrapping, and many more feature modules.

* Components define Template views
* Components use services

The NgModules make developers to organize an application into connected blocks of functionality.

The NgModule properties for the minimum "AppModule" generated by the CLI are as follows:

* **Declarations:** Use to declare the application components.
* **Imports:** Every application must import BrowserModule to run the app in a browser.
* **Providers:** There are none to start.
* **Bootstrap:** This is a root AppComponent that Angular creates and inserts into the index.html host web page.

8) How would you update Angular 6 to Angular 7?

You can update Angular 6 to Angular 7 by using the following command:

1. ng update @angular/cli @angular/core

9) What is the UrlSegment Interface in Angular 7?

In Angular 7, the UrlSegment interface represents a single URL segment, constructor, properties and methods like this:

1. **class** UrlSegment {
2. constructor(path: string, parameters: {...})
3. path: string
4. parameters: {...}
5. toString(): string
6. }

The UrlSegment is a part of a URL between the two slashes and it contains a path and matrix parameters associated with the segment.

10) What is Do Bootstrap (ng Do Bootstrap) In Angular 7?

The ng Do Bootstrap is a new life-cycle hook added in Angular 7 and Do Bootstrap is an interface.

**Example**

1. //ng Do Bootstrap - Life-Cycle Hook Interface
2. classApp Module **implements** Do Bootstrap {
3. ng Do Bootstrap(appRef: ApplicationRef) {
4. appRef.bootstrap(AppComponent);
5. }
6. }

11) What are directives in Angular7?

In Angular7, directives are used to add behavior to an existing DOM element or an existing component instance.

**For Example**

1. **import** { Directive, ElementRef, Input } from '@angular/core';
2. @Directive({ selector: '[myHighlight]' })
3. export **class** HighlightDirective {
4. constructor(el: ElementRef) {
5. el.nativeElement.style.backgroundColor = 'green';
6. }
7. }

Now this directive extends HTML element behavior with a green background as below:

Highlight me!

12) What are components in Angular7?

Components are the basic building blocks of an Angular app formed like a tree structure. Components are subset of directives but unlike directives, components always have a template and only one component can be instantiated per an element in a template.

**For example:**

1. **import** { Component } from '@angular/core';
2. @Component ({
3. selector: 'my-app',
4. template: ` <div>
5. <h1>{{title}}</h1>
6. <div>Learn Angular6 with examples</div>
7. </div> `,
8. })
9. export **class** AppComponent {
10. title: string = 'Welcome to Angular world';
11. }

13) What is the difference between component and directive?

A component (@component) is a directive with a template. Some major difference between components and directives are:

|  |  |
| --- | --- |
| **Component** | **Directive** |
| If you register a component, you have to use @Component meta-data annotation | If you register a directive, you have to use @Directive meta-data annotation |
| Components are used to break up the application into smaller components. | Directives are used to design re-usable components. |
| Components are used to create UI widgets. | Directives are used to add behavior to an existing DOM element. |
| Only one component can be present per DOM element. | Many directives can be used per DOM element. |
| @View decorator or templateurl/template are mandatory | Directives don't use View. |

14) What is a template in Angular7?

A template is a HTML view where you display your data by binding controls to Angular component's properties. You can store your component's template in one of two places. You can define it inline using the template property, or you can define the template in a separate HTML file and link to it in the component metadata using the @Component decorator's templateUrl property.

**For example:**

**Using inline template with template syntax**

1. **import** { Component } from '@angular/core';
2. @Component ({
3. selector: 'my-app',
4. template: '
5. <div>
6. <h1>{{title}}</h1>
7. <div>Learn Angular</div>
8. </div>
9. '
10. })
11. export **class** AppComponent {
12. title: string = 'Hello World';
13. }

**Using separate template file such as app.component.html**

1. **import** { Component } from '@angular/core';
2. @Component ({
3. selector: 'my-app',
4. templateUrl: 'app/app.component.html'
5. })
6. export **class** AppComponent {
7. title: string = 'Hello World';
8. }

15) What is a module?

Modules are the logical boundaries in the application. An application is divided into separate modules to separate the functionalities of the application.

**For example:** app.module.ts root module declared with @NgModule decorator

1. **import** { NgModule }      from '@angular/core';
2. **import** { BrowserModule } from '@angular/platform-browser';
3. **import** { AppComponent }  from './app.component';
4. @NgModule ({
5. imports:      [ BrowserModule ],
6. declarations: [ AppComponent ],
7. bootstrap:    [ AppComponent ]
8. })
9. export **class** AppModule { }

**Here, the NgModule decorator has three options:**

* The imports option is used to import other dependent modules. The BrowserModule is required by default for any web based angular application.
* The declarations option is used to define components in the respective module.
* The bootstrap option tells Angular which Component to bootstrap in the application.

16) What are the new features added in Angular7?

Following is a list of new features added in Angular7:

* Angular7 displays an elegant look in the new update.
* It provides virtual scrolling by using the scrolling package .
* It facilitates you to use drag and drop property by using the @angular/cdk/drag-drop module.
* In Angular7, libraries make changes to itself automatically with the updated version of the Material design.
* Angular7 provides better error handling for @Output if property is not initialized.
* Angular7 provides added support for Node v10.

**Some more Angular7 features are:**

**Angular Console:** It is a downloadable console to start and run Angular projects on your local machine.

**@angular/fire:** It is a latest update on npm, and has its first stable release for Angular7.

**NativeScript:** It facilitates you to make a single project that builds for both web and installed mobile.

**StackBlitz:** StackBlitz 2.0 is now released and includes the Angular Language Service and more features like tabbed editing.

**Angular 8 Interview Questions and Answers**

[Anil Singh](https://www.blogger.com/profile/09359926778482233933) [9:59 AM](https://www.code-sample.com/2019/02/angular-8-interview-questions-and.html)

|  |  |
| --- | --- |
| [What Is Angular?](https://www.code-sample.com/2019/02/angular-8-interview-questions-and.html" \l "Q1) | [What's New in Angular 8?](https://www.code-sample.com/2019/02/angular-8-interview-questions-and.html" \l "Q2) |
| [What Are Differences in Angular 8 and Angular 7?](https://www.code-sample.com/2019/02/angular-8-interview-questions-and.html" \l "Q3) | [List of All Angular Version Interview Questions](https://www.code-sample.com/2018/05/angular-2-4-5-6-7-8-interview-questions.html) |

**What Is Angular?**

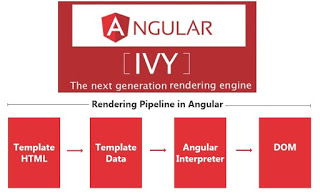
Angular is the most popular web development framework for developing mobile apps as well as desktop applications.

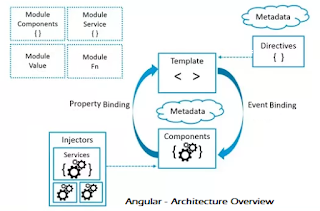
The angular framework is also utilized in the cross-platform mobile development called IONIC and so it is not limited to web apps only.

Angular is an open source framework written and maintained by the angular team at Google and the Father of Angular is [Misko Hevery](http://misko.hevery.com/about/).

**Misko Hevery** - Agile Coach at Google, Attended Santa Clara University and Lives in Saratoga, CA.

Angular is written in TypeScript and so it comes with all the capabilities that typescript offers.





You don’t worry about the TypeScript versions. The compiler manages to the versioning related problems and Angular team working with Traceur compiler team to provide the support to build some extensions.

**What's New in Angular 8?**

Angular 8 being smaller, faster and easier to use and it will making Angular developers life easier. Angular version numbers have three parts**: major.minor.patch**. 

**The latest features introduced during the Angular 8 release candidate stage include:**

1.            Smaller bundles, CLI APIs, and alignment with the ecosystem

2.            AngularJs Migration Improvements

3.            Route Configurations use Dynamic Imports

4.            Differential Loading by Default

5.            Web Worker Support

6.            New Deprecation Guide

7.            Builder APIs in the CLI

8.            Workspace APIs in the CLI

9.            Ivy & Bazel

10.       Ivy aims to change this. Compared with the current Angular View Engine, Ivy provides the following benefits.

                                                                                i.      Generated code that is easier to read and debug at runtime

                                                                              ii.      Faster re-build time

                                                                            iii.      Improved payload - applications size improvements

                                                                            iv.      Improved template type checking

                                                                              v.      Great backward compatibility

**Differential loading of modern JavaScript:-**

Use “**ngx-build-modern**”: Differential Serving for Angular and the CLI.

Must be:-

1.            Angular >= 7.0.0

2.            Angular CLI >= 7.0.0

The important features of “**ngx-build-modern”**:-

       Create optimized bundles for modern browsers

       Create legacy bundles for older browsers

       Make the browser loads the right set of bundles

       Automate this all by providing a CLI extension

For more detail about – [**ngx-build-modern**](https://www.npmjs.com/package/ngx-build-modern)

This release contains the following added and Improvements over the entire Angular platform including:-

=> Added Support for [*TypeScript 3.2*](https://www.code-sample.com/2017/06/angular-2-typescript-tutorial-for.html)

=> Added a Navigation Type Available during Navigation in the Router

=> Added *pathParamsOrQueryParamsChange* mode for *runGuardsAndResolvers* in the Router

=> Allow the passing state to *router link* Directives in the [Router](https://www.code-sample.com/2018/05/angular-5-6-7-routing-and-navigation.html)

=> Allow the passing state to *NavigationExtras* in the Router

=> Restore the whole object when navigating back to a page managed by Angular Router

=> Added support for *SASS*

=> Resolve generated *Sass/Less* files to .css inputs

**=> Added Predicate function mode for *runGuardsAndResolvers*:-**

*This option means guards and resolvers will ignore changes when a provided predicate function returns `false`. This supports use cases where an application needs to ignore some param updates but not others.*

*For example, changing a sort param in the URL might need to be ignored, whereas changing the `project` param might require re-run of guards and resolvers.*

**=> Added functionality to mark a control and its descendant controls as touched: -** add*markAllAsTouched () to AbstractControl*

**=> Added ng-new command that builds the project with Bazel**

**=> Use image based cache for windows BuildKite**

**=> Export *NumberValueAccessor* & *RangeValueAccessor* directives**

**=> Use shared *DomElementSchemaRegistry* instance for improve performance of platform-server*(@angular/platform-server*)**:-

*Right now the ServerRendererFactory2` creates a new instance of the `DomElementSchemaRegistry` for each and every request, which is quite costly (for the Tour of Heroes SSR example this takes around \*\*15%\*\* of the overall execution time)*

**=> Now the Performance Improvements on the core, more consistent about “*typeof checks*”: -**

*When testing whether `value` is an object, use the ideal sequence of strictly not equal to `null` followed by `typeof value === 'object'` consistently. Specifically, there's no point in using double equal with `null` since `undefined` is ruled out by the `typeof` check.*

*\*

*Also avoid the unnecessary ToBoolean check on `value.ngOnDestroy` in `hasOnDestroy()`, since the `typeof value.ngOnDestroy === 'function'` will only let closures pass and all closures are truish (with the notable exception of `document.all`, but that shouldn't be relevant for the `ngOnDestroy` hook)*

**=> In the Compiler-CLI, expose ngtsc as a TscPlugin**

**=> Restore the whole object when navigating back to a page managed by**[**Angular Router**](https://www.code-sample.com/2018/05/angular-5-6-7-routing-and-navigation.html)**:-**

*This feature adds a few capabilities. First, when a `popstate` event fires the value of `history.state` will be read and passed into `NavigationStart`. In the past, only the `navigationId` would be passed here.*

*Additionally, `NavigationExtras` has a new public API called `state` which is any object that will be stored as a value in `history.state` on navigation.*

*For example, the object `{foo: 'bar'}` will be written to `history.state` here: -`router.navigateByUrl('/simple', {state: {foo: 'bar'}});`*

**What Are Components in Angular?**

Components are the most basic building block of a UI in Angular applications and it controls views (HTML/CSS). They also communicate with other components and services to bring functionality to your applications.

Technically components are basically TypeScript classes that interact with the HTML files of the components, which get displayed on the browsers.

The component is the core functionality of Angular applications but you need to know to pass the data into the components to configure them.

Angular applications must have a root component that contains all other components.

Components are created using a @Component decorator that is part of @angular/core module.

You can create your own project using Angular CLI, this command allows you to quickly create an Angular application like - generate components, services, pipes, directive, classes, and modules, and so on as per your requirements.

Explore in detail about [Angular Components click…](https://www.code-sample.com/2018/04/what-are-components-in-angular-6-54-and.html)

**What Is Modules?**

The NgModule is a TypeScript class marked by the @NgModule decorator.

The module is a class and works with the @NgModule decorator function and also takes a metadata object that tells Angular how to compile and run module code.

The Angular module helps you to organize an application into associative blocks of functionality.

An angular module represents a core concept and plays a fundamental role in structuring Angular applications.

The NgModule is used to simplify the ways you define and manage the dependencies in your applications and also you can consolidate different components and services into associative blocks of functionality.

Every Angular application should have at least one module and it contains the components, service providers, pipes and other code files whose scope is defined by the containing NgModule.

The purpose of the module is to declare everything you create in Angular and group them together.

Explore in detail about [Angular module click…](https://www.blogger.com/www.code-sample.com/2018/04/modules-ngmodule-angular-4-5-6.html)

**What Are Angular Directives?**

Angular Directive is a TypeScript class which is declared as a @directive decorator.

The directives allow you to attach behavior to DOM elements and the @directive decorator provide you an additional metadata that determines how directives should be processed, instantiated, and used at run-time.

Explore in detail about [Angular Directives click…](https://www.code-sample.com/2018/05/angular-5-6-7-directives-decorator.html)

**What Is Dependency Injection (DI)?**

Dependency Injection is a powerful pattern for managing code dependencies. DI is a way to create objects that depend upon other objects.

Angular has its own DI framework pattern, and you really can't build an Angular application without Dependency injection (DI).

A DI system supplies the dependent objects when it creates an instance of an object.

Explore in detail about [Angular Dependency Injection (DI) click…](https://www.code-sample.com/2018/05/dependency-injection-di-angular-5-6.html)

**What Is Angular Pipe?**

Pipes transform displayed values within a template.

Use the @Pipe annotation to declare that a given class is a pipe. A pipe class must also implement a PipeTransform interface.

The @Pipe decorator allows you to define the pipe name that is globally available for use in any template in the across Angular apps.

Pipe class implements the “PipeTransform” interfaces transform method that accepts an input value and returns the transformed result.

There will be one additional argument to the transform method for each parameter passed to the pipe.

Explore in detail about [Angular Pipes Decorator click…](https://www.code-sample.com/2018/05/angular-5-6-7-pipes-decorator.html)

**What Is runGuardsAndResolvers function?**

This option means **guards** and **resolvers** will ignore changes when a provided predicate function returns `**false**`. This supports use cases where an application needs to ignore some param updates but not others.

For example, changing a sort param in the URL might need to be ignored, whereas changing the `**project**` param might require re-run of guards and resolvers.

**What Is “typeof checks” in Angular 8?**

**How Performance Improvements on the core in Angular 8?**

When testing whether `**value**` is an object, use the ideal sequence of strictly not equal to `**null**` followed by `**typeof value === 'object'**` consistently. Specifically, there's no point in using double equal with `**null**` since `**undefined**` is ruled out by the `**typeof**` check.

Also avoid the unnecessary ToBoolean check on `**value.ngOnDestroy**` in `**hasOnDestroy**()`, since the `**typeof value.ngOnDestroy === 'function'**` will only let closures pass and all closures are truish (with the notable exception of `**document.all**`, but that shouldn't be relevant for the `**ngOnDestroy**` hook)

**How to restore the whole object when navigating back to a page managed by Angular Router in Angular 8?**

This feature adds a few capabilities.

First, when a `**popstate**` event fires the value of `**history.state**` will be read and passed into `**NavigationStart**`. In the past, only the `**navigationId**` would be passed here.

Additionally, `**NavigationExtras**` has a new public API called `**state**` which is any object that will be stored as a value in `**history.state**` on navigation.

For example, the object **`{name: 'anil'}**` will be written to `**history.state**` here: -`**router.navigateByUrl('/simple', {state: {name: 'anil'}});**`

**What Is the Navigation Type Available during Navigation in the Angular 8 Router?**

**What Is Bazel?**

Google open sourced the software responsible for building most of our projects under the name Bazel. Bazel is a powerful tool which can keep track of the dependencies between different packages and build targets.

**What Are the features of Bazel?**

Bazel is independent of the tech stack.

It has a smart algorithm for determining the build dependencies

**What Are the Angular 8 Best practices?**

Don't modify your copy of Angular

Avoid Angular APIs marked in the documentation as “Security Risk.”

Preventing cross-site scripting (XSS)

Keep current with the latest Angular library releases.

Check the Angular log for security-related updates in the regularly.

Remember, whether a value is safe depends on context, so choose the right context for your intended use of the value.

Normally, Angular automatically sanitizes the URL, disables the dangerous code, and in development mode, logs this action to the console.

**What Are the Angular Security Principles?**

Security Principles of Angular Applications:

1.          You should avoid direct use of the DOM APIs.

2.          You should enable Content Security Policy (CSP) and configure your web server to return appropriate CSP HTTP headers.

3.          You should Use the offline template compiler.

4.          You should Use Server Side XSS protection.

5.          You should Use DOM Sanitizer.

6.          You should Preventing CSRF or XSRF attacks.

**What Is Cross Site Scripting (XSS) Attack?**

The Cross Site Scripting (XSS) attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications.

The Cross Site Scripting (XSS) attacks are common on web browsers and it carried out on websites around 84% (approximately).

**How To Preventing Cross Site Scripting (XSS) in Angular?**

**How Angular Protects Us From XSS Attacks?**

The Cross Site Scripting (XSS) attack is a type of injection and attackers inject your web applications using the client side scripts and malicious code into web pages.

An attacker can insert vulnerability scripts and malicious code in your web applications.

The Angular treats all values as untrusted by default. This is the great advantages of Angular.

When a value is Inserted Vulnerability into the DOM from –

1.          A Template

2.          Property

3.          Attribute

4.          Style

5.          Class Binding

6.          Interpolation

Angular recognizes the value as unsafe and automatically sanitizes and removes the script tag and other security vulnerabilities.

For more detail [about Angular security explore in detail….](https://www.code-sample.com/2017/11/angular-security-xss-csrf.html)

**Stayed Informed** – [*Angular Book - All in One (Including Version 2, 4, 5, 6 and 7)*](https://www.code-sample.com/2018/11/angular-7-6-5-4-interview-questions-book.html)

**Stayed Informed** – [*Angular Interview Question (Including Version 2, 4, 5, 6 and 7)*](https://www.code-sample.com/2018/05/angular-2-4-5-6-7-8-interview-questions.html)

**Angular 8 beta** will be release on March/April 2019.

The following table contains our current target release dates for the next two major versions of Angular:

| **Date** | **Stable Release** | **Compatibility** |
| --- | --- | --- |
| March/April 2019 | 8.0.0 | ^7.0.0 |
| September/October 2019 | 9.0.0 | ^8.0.0 |

Why do we need package.json file?

Let’s dive into the exciting world of JS & Angular 2.

This blog will cover package.json file in nodeJS , dependencies, etc.

Before getting to know why we need it ,we should know what it is.

All NPM (Node Package Manager) packages contain a file, usually in the project root, called package.json file. package.json contains all the information of your web app.It contains all the metadata{set of data which describes and gives info about all other data}.

In layman terms it is the first file your browser looks for, to find all the information of your web app.

There are fields for the description and keywords of your projects like dependencies.

What are Dependencies ?

The dependencies property. The dependencies property of a module’s package.json is where dependencies — the other modules that this module uses — are defined. The dependencies property takes an object that has the name and version at which each dependency should be used.

The dependencies in package.json are shown in an example below:

package.json example

The example has jquery , bootstrap etc installed. To add other dependencies use command :

npm install {dependencies\_name} // don’t use braces

It will install the latest version available for the dependency.

So,Why do we need package.json?

package.json provides a simple way to keep track of packages that are being used in application.