**Angular Interview Questions:**

**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**: **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

**Question**: **Why prioritize TypeScript over JavaScript in Angular?**  
**Answer**: TypeScript is developed by Microsoft and it is a superset of JavaScript. The issue with JS is that it isn’t a true OOP language. As the JS code doesn’t follow the Prototype Pattern, the bigger the size of the code the messier it gets. Hence, it leads to difficulties in maintainability as well as reusability. To offset this, TypeScript follows a strict OOP approach.

### 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. |

[**Difference between Constructor and ngOnInit**](https://stackoverflow.com/questions/35763730/difference-between-constructor-and-ngoninit)

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. Angular or better Dependency Injector (DI) analyzes the constructor parameters and when it creates a new instance by calling new MyClass() it tries to find providers that match the types of the constructor parameters, resolves them and passes them to the constructor like

new MyClass(someArg);

ngOnInit is a life cycle hook called by Angular2 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):

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

then to use the method of OnInit we have to implement in the class like this.

export class App implements OnInit{

constructor(){

//called first time before the ngOnInit()

}

ngOnInit(){

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

}

}

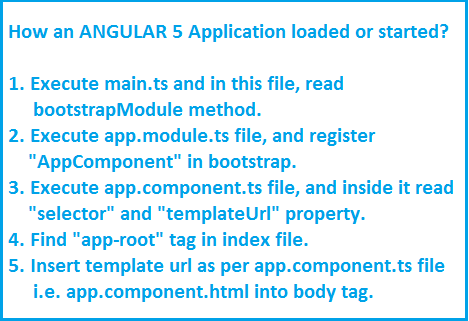
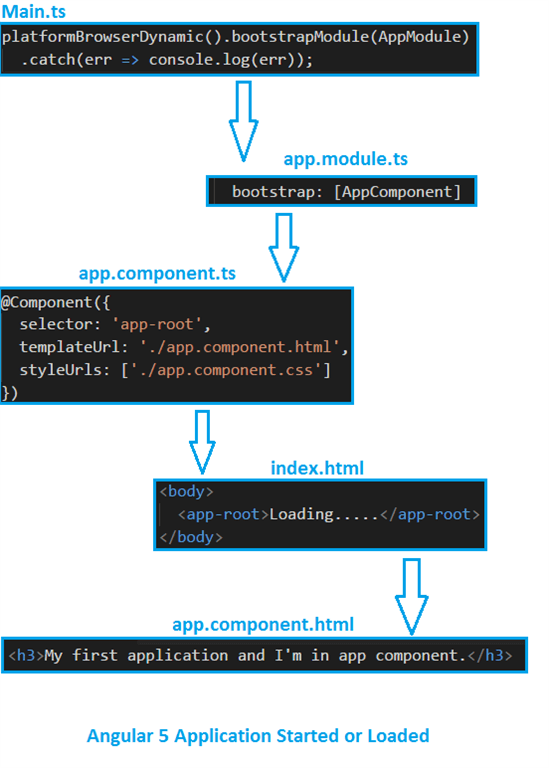
Implement this interface to execute custom initialization logic after your directive's data-bound properties have been initialized. ngOnInit is called right after the directive's data-bound properties have been checked for the first time, and before any of its children have been checked. It is invoked only once when the directive is instantiated.

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".

So you should use constructor() to setup Dependency Injection and not much else. ngOnInit() is better place to "start" - it's where/when components' bindings are resolved.

**Which is the first file loaded after using ng serve command?**

When you use the [ng serve](https://angular.io/cli/serve) command to build an app and serve it locally, the server automatically rebuilds the app and reloads the page when you change any of the source files. Whenever ng-serve builds our application, it creates “bundles” and automatically adds these to our index.html file at runtime. So, from these bundles, the first code has to be executed from “main.ts” file, i.e., “main.ts” file is the main file from where the execution of an Angular application will start.



# [What is the difference between Promises and Observables?](https://stackoverflow.com/questions/37364973/what-is-the-difference-between-promises-and-observables)

**Promise**

A Promise handles a **single event** when an async operation completes or fails.

Note: There are Promise libraries out there that support cancellation, but ES6 Promise doesn't so far.

**Observable**

An Observable is like a **Stream** (in many languages) and allows to pass zero or more events where the callback is called for each event.

Often Observable is preferred over Promise because it provides the features of Promise and more. With Observable it doesn't matter if you want to handle 0, 1, or multiple events. You can utilize the same API in each case.

Observable also has the advantage over Promise to be **cancelable**. If the result of an HTTP request to a server or some other expensive async operation isn't needed anymore, the Subscription of an Observable allows to cancel the subscription, while a Promise will eventually call the success or failed callback even when you don't need the notification or the result it provides anymore.

Observable provides **operators** like map, forEach, reduce, ... similar to an array

There are also powerful operators like retry(), or replay(), ... that are often quite handy.