**Angular Interview Questions:**

1. **What is Angular?**  
   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.

1. **Define the ng-content Directive?**

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 will not work the way it worked for HTML elements. In order to make it work just as the HTML example mentioned above, we need to use the ng-content Directive. Moreover, it is helpful in building reusable components.

1. **Explain the various features of Angular.**

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.

1. **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.

1. **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. On the other hand, 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.



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

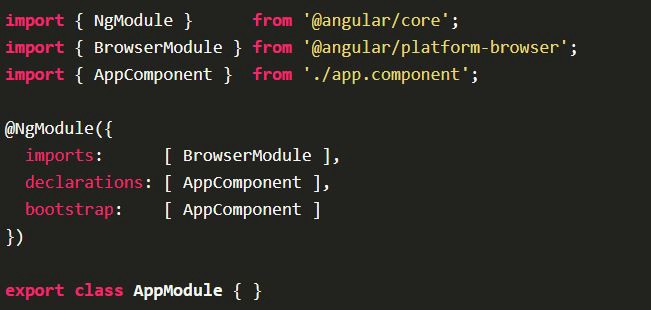
An Angular application consists largely of components and their HTML templates. Before the browser can render the application, the components and templates must be converted to executable JavaScript by the *Angular compiler*.

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.

1. **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.



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

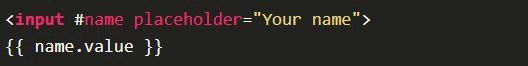
1. **What are all the *metadata* properties of NgModule? And what are they used for?**

@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.

1. **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.

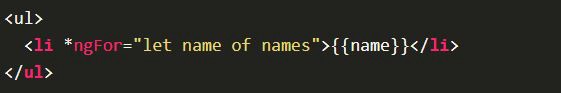


In the above code the #name declares a variable on the input element. Here the name refers to the *input* element. Now we can access any property of the inputDOM, using this reference variable. For example, we can get the value of the inputelement as name.value and the value of the placeholder property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

1. **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.



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.

1. **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.

1. **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.

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

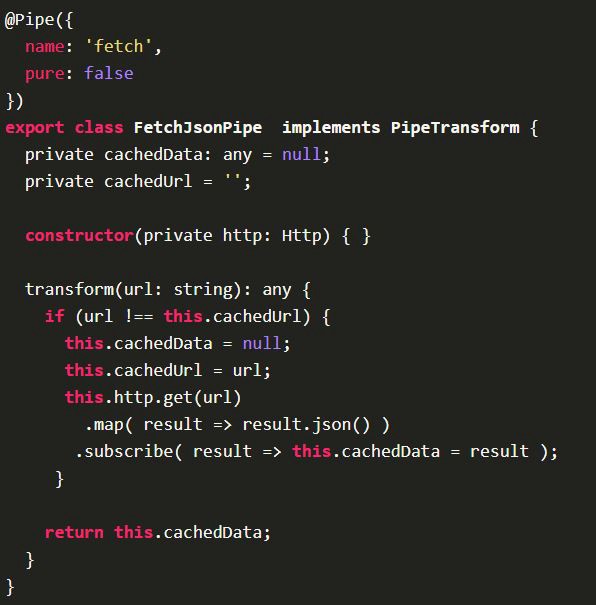
Services encapsulates business logic and separates them from UI concerns or the controller concerns, which governs them both.

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.

1. **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:



1. **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.

1. **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.

1. **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** |

1. **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. |

1. **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 |

1. **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.

## ****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; }; }

## ****List some tools for testing angular applications?****

1. Karma
2. Angular Mocks
3. Mocha
4. Browserify
5. Sion
6. **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.

1. **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.



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.

1. **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. **Demonstrate navigating between different routes in an Angular application.**

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']);

}

}

1. **Could you explain services in Angular?**  
   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.

1. **Discuss the advantages and disadvantages of using Angular?**  
   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

1. **Enumerate some salient features of Angular 7.**  
   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.

1. **What is string interpolation in Angular?**  
   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.

1. **Explain Angular Authentication and Authorization.**  
   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

1. **Can you explain the concept of scope hierarchy in Angular?**  
   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

1. **How to generate a class in Angular 7 using CLI?**

ng generate class Dummy [options]

This will generate a class named Dummy.

1. **Explain what is the difference between Angular and backbone.js?**  
   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.

1. **How do Observables differ from Promises?**  
   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.
2. **Please explain the difference between Angular and AngularJS?**

 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.



1. **Could you explain the concept of templates in Angular?**  
   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.
2. **What should replace the “?”?**  
   Directives. The image represents the types of directives in Angular; Attribute, structural, and custom.
3. **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.

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

1. **What are the building blocks of Angular?**  
   There are essentially 9 building blocks of an Angular application. These are:
2. **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.
3. **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.
4. **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.
5. **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.
6. **Metadata –** In order to let Angular know how to process a class, metadata is attached to the class. For doing so decorators are used.
7. **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.
8. **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.
9. **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.
10. **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.
11. **Please explain the differences between Angular and jQuery?**  
    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

1. **Could you explain the difference between Angular expressions and JavaScript expressions?**  
   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

1. **Can you give us an overview of Angular architecture?**



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

1. **What is Angular Material?**  
   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.
2. **What is AOT (Ahead-Of-Time) Compilation?**  
   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.
3. **What is Data Binding? How many ways it can be done?**

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.
4. **What is demonstrated by the arrow in the following image?**



1. **Can you draw a comparison between the service() and the factory() functions?**

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.

1. **Please explain the digest cycle in Angular?**  
   The process of monitoring the watch list 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.

1. **Could you explain the various types of filters in Angular.**  
   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

1. **What is new in Angular 6?**  
   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”

1. **What is ngOnInit ()? How to define it?**  
   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

}

1. **What is SPA** **(Single Page Application) in Angular? Contrast SPA technology with traditional web technology?**  
   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.

1. **What is the code for creating a decorator?**  
   Let’s create a decorator called Dummy:

function Dummy(target) {

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

}

1. **What is the process called by which TypeScript code is converted into JavaScript code?**  
   It is called Transpiling. Even though TypeScript is used for writing code in Angular applications, it gets internally transpiled into equivalent JavaScript.
2. **What is ViewEncapsulation and how many ways are there do to do it in Angular?**  
   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. **Why prioritize Typescript over JavaScript in Angular?**  
   Typescript is developed by Microsoft and it is a superset of JavaScript. The issue with JS is that it is not 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.
2. **Angular Lifecycle Hooks:**

## Lifecycle sequence

After creating a component/directive by calling its constructor, Angular calls the lifecycle hook methods in the following sequence at specific moments:

|  |  |
| --- | --- |
| **Hook** | **Purpose and Timing** |
| ngOnChanges() | Respond when Angular (re)sets data-bound input properties. The method receives a [SimpleChanges](https://angular.io/api/core/SimpleChanges) object of current and previous property values.  Called before ngOnInit() and whenever one or more data-bound input properties change. |
| ngOnInit() | Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties.  Called once, after the first ngOnChanges(). |
| ngDoCheck() | Detect and act upon changes that Angular can't or won't detect on its own.  Called during every change detection run, immediately after ngOnChanges() and ngOnInit(). |
| [ngAfterContentInit()](https://angular.io/api/router/RouterLinkActive#ngAfterContentInit) | Respond after Angular projects external content into the component's view / the view that a directive is in.  Called once after the first ngDoCheck(). |
| ngAfterContentChecked() | Respond after Angular checks the content projected into the directive/component.  Called after the [ngAfterContentInit()](https://angular.io/api/router/RouterLinkActive#ngAfterContentInit) and every subsequent ngDoCheck(). |
| [ngAfterViewInit()](https://angular.io/api/forms/NgForm#ngAfterViewInit) | Respond after Angular initializes the component's views and child views / the view that a directive is in.  Called once after the first ngAfterContentChecked(). |
| ngAfterViewChecked() | Respond after Angular checks the component's views and child views / the view that a directive is in.  Called after the [ngAfterViewInit()](https://angular.io/api/forms/NgForm#ngAfterViewInit) and every subsequent ngAfterContentChecked(). |
| ngOnDestroy() | Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks.  Called just before Angular destroys the directive/component. |

1. **Inline and Block elements:**

**Block-level Elements**

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

Example

<div>Hello World</div>

Block level elements in HTML:

<address> <article> <aside> <blockquote> <canvas> <dd> <div> <dl> <dt> <fieldset> <figcaption> <figure> <footer> <form> <h1> --- <h6> <header> <hr> <li> <main> <nav> <noscript> <ol> <p> <pre> <section> <table> <tfoot> <ul>

**Inline Elements**

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline <span> element inside a paragraph.

Example

<span>Hello World</span>

Inline elements in HTML:

<a> <abbr> <acronym> <b> <bdo> <big> <br> <button> <cite> <code> <dfn> <em> <i> <img> <input> <kbd> <label> <map> <object> <output> <q> <samp> <script> <select> <small> <span> <strong> <sub> <sup> <textarea> <time> <tt> <var>

-----------------------------------------------------------------------------------------------------------------------------------------

**Online Code Executor for Angular:** <https://codesandbox.io/s/angular-ymtl5>

**Author:** Pranam K

**Email ID:** [pranam707@gmail.om](mailto:pranam707@gmail.om)

**Contact Number:** +91 8762632589

Follow me on:

**LinkedIn:** <https://www.linkedin.com/in/pranam-bhat-11670689/>

Connect me on:

**Instagram:** <https://www.instagram.com/pranam_speaks/>

**Quora:** <https://www.quora.com/profile/Pranam-Bhat>

**Facebook:** <https://www.facebook.com/pranam.bhat>

**Twitter:** <https://twitter.com/PranamBhat>