[1) What is Angular? 2](#_Toc29803889)

[2) What is building blocks of Angular? 2](#_Toc29803890)

[3) How to set up Angular 8? 3](#_Toc29803891)

[4) Angular 8 project structure? 4](#_Toc29803892)

[5) Angular vs JQuery? 4](#_Toc29803893)

[6) Angular Expression vs Java Script expression?? 4](#_Toc29803894)

[7) What is Angular Architecture??? 5](#_Toc29803895)

[8) What is directive? 6](#_Toc29803896)

[9) Does Angular support Nested Controller? 7](#_Toc29803897)

[10) List down the ways of communication between application modules in angular? 7](#_Toc29803898)

[11) What is ngContent? 7](#_Toc29803899)

[12) How to use routing? 7](#_Toc29803900)

[13) What is Services in angular? 8](#_Toc29803901)

[14) Advantages and disadvantages of Angular? 8](#_Toc29803902)

[15) Features of Angular 7??? 9](#_Toc29803903)

[16) What is String interpolation? 9](#_Toc29803904)

[17) Angular authentication and Authorization? 9](#_Toc29803905)

[18) Scope? 9](#_Toc29803906)

[19) How to generate a class using cli? 10](#_Toc29803907)

[20) Difference between Angular and backbone.js 10](#_Toc29803908)

[21) How do Observables differ from Promises??? As soon as a promise 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. 11](#_Toc29803909)

[22) Angular vs AngularJS 11](#_Toc29803910)

[23) What is template??? 12](#_Toc29803911)

[24) What is difference between Annotation and decorator??? 12](#_Toc29803912)

[25) What is Angular material? 12](#_Toc29803913)

[26) What is AOT – Ahead of Time compilation ?? 13](#_Toc29803914)

[27) What is data binding? 13](#_Toc29803915)

[28) Can you draw a comparison between the service() and the factory() functions? 13](#_Toc29803916)

[29) Please explain the digest cycle in Angular??? 14](#_Toc29803917)

[30) Could you explain the various types of filters in AngularJS. 14](#_Toc29803918)

[31) What is new in Angular 6??? 15](#_Toc29803919)

[32) What is ngOnInit()? How to define it? 15](#_Toc29803920)

[33) What is SPA (Single Page Application) in Angular? Contrast SPA technology with traditional web technology? 15](#_Toc29803921)

[34) What is the code for creating a decorator? 16](#_Toc29803922)

[35) What is the process called by which TypeScript code is converted into JavaScript code? 16](#_Toc29803923)

[36) What is ViewEncapsulation and how many ways are there do to do it in Angular??? 16](#_Toc29803924)

[37) Why prioritize TypeScript over JavaScript in Angular??? 16](#_Toc29803925)

[38) Discuss the lifecycle designed for directive and components in Angular JS especially for the newly introduced version 6.0? 17](#_Toc29803926)

[39) Write the features for Angular 6 over Angular 5. 18](#_Toc29803927)

[40) Referencecs: 18](#_Toc29803928)

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.
2. What is building blocks of Angular?

**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. 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.

**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.

**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.

**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 are attribute and structural directives.

**Metadata** – In order to let Angular know how to process a class, metadata is attached to the class. For doing so decorators are used.

**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.

**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.

**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.

**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.

# How to set up Angular 8?

<https://angular.io/guide/setup-local>

# Angular 8 project structure?

# Angular vs 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

# Angular Expression vs Java Script expression??

### 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 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 }}**

### 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

# What is Angular Architecture???



### Together, a component and template define an Angular view.

### A decorator on a component class adds the metadata, including a pointer to the associated template.

### Directives and binding markup in a component's template modify views based on program data and logic.

### The dependency injector provides services to a component, such as the router service that lets you define navigation among views.

### <https://angular.io/guide/architecture>

### Angular is a platform and framework for building client applications in HTML and TypeScript. Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.

### The basic building blocks of an Angular application are NgModules, which provide a compilation context for components. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules. An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

### Components define views, which are sets of screen elements that Angular can choose among and modify according to your program logic and data.

### Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making your code modular, reusable, and efficient.

### Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.

### The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.

### The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI).

### An app's components typically define many views, arranged hierarchically. Angular provides the [Router](https://angular.io/api/router/Router) service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities.

# What is directive?

### Structural directives shape or reshape the DOM's structure, typically by adding, removing, and manipulating the elements to which they are attached. Any directive with an asterisk, \*, is a structural directive.

# What is component in angular?

### *Components* define areas of responsibility in the user interface, or UI, that let you reuse sets of UI functionality. You've already built one with the product list component.

### A component consists of three things:

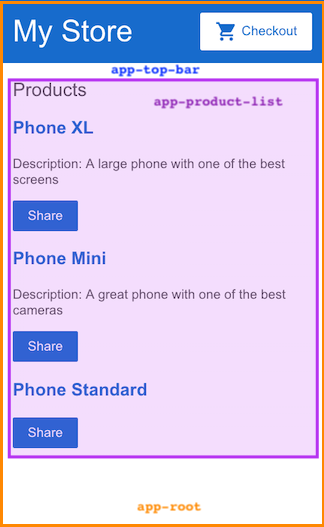
### A component class that handles data and functionality. In the previous section, the product data and the share() method in the component class handle data and functionality, respectively.

### An HTML template that determines the UI. In the previous section, the product list's HTML template displays the name, description, and a "Share" button for each product.

### Component-specific styles that define the look and feel. Though product list does not define any styles, this is where component CSS resides.

### An Angular application comprises a tree of components, in which each Angular component has a specific purpose and responsibility.

### Currently, the example app has three components:



Component file example:

import { [Component](https://angular.io/api/core/Component), [OnInit](https://angular.io/api/core/OnInit) } from '@angular/core';

@[Component](https://angular.io/api/core/Component)({

selector: 'app-product-alerts',

templateUrl: './product-alerts.component.html',

[styleUrls](https://angular.io/api/core/Component#styleUrls): ['./product-alerts.component.css'] })

export class ProductAlertsComponent implements [OnInit](https://angular.io/api/core/OnInit) { constructor() { } ngOnInit() { } }

1. Notice the @[Component](https://angular.io/api/core/Component)() decorator. This indicates that the following class is a component. It provides metadata about the component, including its selector, templates, and styles.
   * The selector identifies the component. The selector is the name you give the Angular component when it is rendered as an HTML element on the page. By convention, Angular component selectors begin with the prefix app-, followed by the component name.
   * The template and style filenames reference the HTML and CSS files that StackBlitz generates.
2. The component definition also exports the class, ProductAlertsComponent, which handles functionality for the component.

# What is Decorator?

# What are services?

### Services are an integral part of Angular applications. In Angular, a service is an instance of a class that can be made available to any part of your application using Angular's [dependency injection system](https://angular.io/guide/glossary#dependency-injection-di).

### Services are the place where you share data between parts of your application

# What is Module?

# What is ng serve –open

### The ng serve command builds the app, starts the development server, watches the source files, and rebuilds the app as you make changes to those files.

### The --open flag opens a browser to http://localhost:4200/.

# Explain Routing?

The Angular [router](https://angular.io/guide/glossary#router) enables you to show different components and data to the user based on where the user is in the application. The router enables navigation from one view to the next as users perform application tasks:

# Does Angular support Nested Controller?

### 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.

# List down the ways of communication between application modules in angular?

### 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]

1. What is ngContent?

### 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.

### <https://medium.com/@joshblf/wtf-is-ng-content-8382b2a664e1>

# How to use routing?

# What is 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.

# Advantages and disadvantages of 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

# 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 package. Furthermore, Angular 7 comes with a new and enhanced version of the ng-compiler.

# What is String interpolation?

### 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.

# Angular authentication and Authorization?

http://jasonwatmore.com/post/2018/11/16/angular-7-jwt-authentication-example-tutorial

# Scope?

### 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

# How to generate a class using cli?

### Ng generate class MyClass [options]

# Difference 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.

# How do Observables differ from Promises???

### As soon as a promise 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.

# Angular vs 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. On the contrary, Angular has active support with updates rolling out every now and then

# What is template???

# What is difference between Annotation and decorator???

### 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.

# 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.

# What is AOT – Ahead of Time compilation ??

### 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.

# What is data binding?

### 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:

### Event Binding – Enables the application to respond to user input in the target environment

### Property Binding – Enables interpolation of values computed from application data into the HTML

### 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.

* Interpolation {{ }}
* Property binding [ ]
* Event binding ( )

# 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.

# Please explain the digest cycle in Angular???

### 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.

# Could you explain the various types of filters in AngularJS.

### In order to format the value of expression so that it can be displayed to the user, AngularJS has filters. It is possible to add these filters to the controllers, directives, services, or templates. AngularJS 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 AngularJS 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

# 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”

# 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

}

# 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 traditional web technology, SPA technology is faster and easy to develop as well.

### In 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.

# What is the code for creating a decorator?

### We create a decorator called Dummy:

**function** **Dummy**(target) {

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

}

# 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.

# 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

# Why prioritize TypeScript over JavaScript in Angular???

### TypeScript is a superset of Javascript as it is Javascript + Types or extra features like typecasting for variables, annotations, variable scope and much more. The typescript is designed in a way to overcome Javascript shortcomings like typecasting of variables, classes, decorators, variable scope and many more. Moreover, Typescript is purely object-oriented programming that offers a "Compiler" that can convert to Javascript-equivalent code.

# Discuss the lifecycle designed for directive and components in Angular JS especially for the newly introduced version 6.0?

### ngOnInit

### ngDoCheck

### ngOnDestroy

### Constructor

### ngOnChanges

### ngAfterContentInit (only for components)

### ngAfterContentChecked (only for components)

### ngAfterViewInit (only for components)

### ngAfterViewChecked (only for components)

# Write the features for Angular 6 over Angular 5.

### Following are the features:

### **1. Added ng update**

### CLI command updates angular project dependencies to their latest versions. The ng update is a normal package manager tool to identify and update in normal package manager tools to identify and update other dependencies.

### **2. Uses RxJS6**

### This is the third party library (RxJS) and introduces two important changes as compared to RxJS5.

### Introduces a new internal package structure.

### Operator concept

### To update of RxJS6, run the following command:

### npm **install** --save rxjs@6

### To update your existing Angular Project, run the following:

### npm **install** --save rxjs-compat

### **3. The <ng-template>**

### Angular 6 uses <ng-template> instead of <template>

### **4. Service Level Changes**

### If in an earlier version, the user wanted to provide a service to the entire application, the user was required to add it to providers in the AppModule but it is not required in the case of Angular6.

### **5. Renamed Operators**

### Angular 6 has renamed following operators:

### do() => tap()

### catch() => catchError()

### finally() => finalize()

### switch()=>switchAll()

### throw() => throwError

# Referencecs:

<https://hackr.io/blog/angular-interview-questions>

<https://www.edureka.co/blog/interview-questions/top-angularjs-interview-questions-2016/> -- WIP

<https://www.fullstack.cafe/blog/angular-8-interview-questions> -- TODO