Angular Interview Questions:

**#1 Question** What is Angular 2?

**Angular 2** is a completely revived component-based **[Javascript framework](https://www.onlineinterviewquestions.com/javascript/" \t "_blank)** in which an application is a tree of loosely coupled components. It is a more advanced version of angularJS. It is more of an "all in one" framework so it also helps in creating a single website without getting trapped into different JS frameworks. An Angular 2 is a modular framework in which our code is divided into individual procedures that offer a similar kind of functionality, hence improving the testing, up gradation and maintenance of the application. It has a lot of useful features such as- server-side rendering, cross-platform, and supports more languages than any other framework. It is a new typescript framework built around the concept of components which is paving the way for a better and spacious development. We can even make hybrid applications using Angular 2 which gives us a sharp edge by providing us the flexibility to use the same standard codes for developing other applications.

## #2 Question List some advantages of Angular 2 over Angular1.

**Angular 2** is a re-written version of Angular1 and not an update. The best way to compare Angular 2 and Angular 1 is by finding out the new features in Angular 2. This way we will be able to find out the advantages of Angular 2 over Angular1 precisely. So, some of the advantages of Angular 2 are:-

|  |  |
| --- | --- |
| **Angular 2** | **Angular1** |
| Angular 2 is a mobile-oriented framework | Whereas Angular1 was not developed with mobile base in mind. |
| Angular 2 is a versatile framework, i.e. we have more choices for languages. We can use ES5, ES6, Typescript or Dart to write an Angular 2 code | Whereas an Angular1 code can written by using only ES5, ES6 and Dart. We don’t have many choices of language in Angular1. |
| Nowadays, the controllers are replaced by components and Angular 2 is completely component based. | Whereas Angular1 was based on controllers whose scope is now over. |

**#3 Question** What are the new features of Angular 2?

Angular 2 is a platform that encompasses a wide range of capabilities. Some new features were added in Angular 2 which includes:

* **Universal server rendering-** It is the library which is used to make building universal apps a smooth experience. It is an important feature of Angular 2.
* **A mobile toolkit-** It provides all the mobile toolkit and techniques to build high-performance mobile applications. The web applications which are developed using the mobile toolkit can be loaded on any device with or without internet connection which is a great advantage.
* **A command line interface-**it can generate components, routes, services, and pipes with the help of commands.
* **Data binding-** data binding has been improved in Angular 2. So, whatever DOM element property you need to bind, you just wrap it in square brackets. E.g.-

<img[src]='product.image' />

* **Modular-** various modules have been removed from angular’s core, which has resulted in better performance.
* **Modern-** Angular 2 has been targeted as modern browsers in which various hacks that make angular harder to develop have been removed.

## #4 Question How do you define transition between two states in angular?

Transitions between two states take place so that we can build simple animations between two states driven by a model attribute. Transition basically means navigating from the current state to a new state. In angular, the transition is an animation-specific function which is used in angular’s animation DSL language. Transition declares the sequence of animation steps that will be executed when the entered value is satisfied. A function is provided an argument for a transition and it will be executed each time a state change occurs. In this, if the function is true, then the animation will run else it won’t get executed.

These animation transitions are placed within the animation triggers. The transition depends upon what the animation was in the previous state and what it will become in the next state. In other words, if a transition is defined that matches the old/current state criteria then the associated animation will be triggered.

**Syntax:**

function transition (stateChangeExpr: string,steps: AnimationMetadata |

AnimationMetadata []):AnimationTransitionMetadata;

## #5 Question How to declare a component in Angular 2?

Components in Angular 2 are simply directives that are always associated with a direct template. Angular 2 components have an extremely well defined life-cycle. When working with angular components, we can make use of interfaces which allows us to implement functionality for different times in a components lifecycle. A component must belong to an NgModule in order for it to be usable by another component or application. Components can even control their runtime behaviour by implementing various Life-cycle hooks.

##### Declaration of component:

@component ({selector: 'great', template: 'hello {{name}}!'})

Class greet{

Name: string = 'world';

}

Components always have a template and only one component can be instantiated per an element in a template. When a component is instantiated, angular creates a change detector, which is responsible for propagating the component’s building.

**#6 Question** What is the difference between observable and promises?

The differences between observable and promises are:

1. Observable is a more powerful way of handling HTTP asynchronous requests. Whereas, A promise handles a single event when an asynchronous operation completes or fails.
2. An observable is like a stream which allows passing zero or more events where the callback is called for each event. Whereas, A promise eventually calls the success or failed callback even when you don’t need the notification or the result it provides anymore.
3. Observable works with multiple values for a particular time. Whereas, Promises works with and even returns a single value at a time.
4. Observables can be canceled. Whereas, Promises cannot be canceled.
5. Observable supports map, filter, reduce and similar operators. Whereas, Promises have more readable codes with try/catch and async/await.
6. In observable, one operator ‘retry’ can be used to retry whenever needed. Whereas, Promises cannot be retried. A promise should have access to the original function that returned the promise in order to have a retry capability.

**#7 Question** List the differences between Angular 2 components vs. directives.

Apart from components, directives are also used in Angular 2 which allows us to attach behavior to elements in DOM. There are certain differences between the components and directives in Angular 2. They are:

1. In Angular 2, a component is a directive with a view whereas a directive is a decorator with no view. Components are the specific type of directive that allows us to utilize web component functionality throughout our application. Whereas, Directive is the mechanism by which we attach behavior to elements.
2. A component is used to break up the application into smaller components. Whereas, Directive is used to design the re-usable components.
3. Components can be used to define pipes. Whereas, We cannot define pipes using directives.
4. Components can be present per DOM element. Whereas, Directive is used to add behavior to an existing DOM element.

## #8 Question What is ECMAScript?

ECMAScript is a standard for scripting languages. It is a subset of **Javascript**. Languages such as ActionScript, JavaScript use ECMAScript as its core. ECMA stands for European Computer Manufacturer’s Association. Coders commonly use ECMAScript for client-side scripting on the World Wide Web. It is also used for server applications and services. It includes structured, dynamic, functional, and prototype-based features. The ECMAScript was developed by Brendan Eich of Netscape. The ECMAScript is standardized by the ECMA international standards organization in the ECMA-262 and ECMA-402 specifications. It is a programming language which is designed specifically for acting on an existing entity or system. It provides the rules, details, and guidelines that a scripting language must observe to be considered ECMAScript compliant.

## #9 Question What is Traceur Compiler?

Traceur is a compiler which takes ECMAScript and compiles it down to regular Javascript that runs in your browser. Traceur can be used in several ways like- typing or pasting the ES6 code into the read-eval-print-loop page, or by including traceur in the web page and compiling ES6 code content on the fly, or many other ways. Even traceur is written in ES6, compiled to ES5. The main goal of a traceur compiler is to inform the designs of Javascript features and allows us to write the code in a better manner. Nowadays, traceur compilers are broadly used in Angular 2 platform. It also supports transpilling and type checking via type annotations.

**#10 Question** List the modern browsers supported by Angular 2.

Angular supports most of the recent browsers some of which are:

* Google Chrome
* Firefox
* Edge
* IE for versions 9-11
* Safari
* iOS 7.1
* Android 4.1
* IE Mobile

## #11 Question When to use Ngoninit and constructor in Angular 2?

Constructors are used for initializing class members and also for dependency injection. Ngonlnit is used for the initialization work. Both of these methods are called when the component is created. It is really important that we should know, when to and how to use them. These are used for providing the best structure for your component’s code. A constructor method is a pre-defined method in the constructor class which is only called when the class is instantiated. It is also used for properly initializing the fields. The constructor in Angular 2 is used to create a new instance of the class. Ngonlnit is the class we import when we implement the constructor in order to use it in a class. The method used in this case is ngOnlnit(). This method helps in initializing the directive or the component after the data-bound properties are displayed and the directive or components input is set.

## #12 Question How to cache an observable data in Angular 2?

Caching of an observable data is done with the help of “observable.cache”. We can use caching in order to cache the response in the memory and then, on the next subscription, instead of requesting the remote server again. This operator is used at the end of the string. Caching is important for the performance, especially on bandwidth restricted devices and slow networks. You should have a good understanding of caching while working with promises but while translating it to observable, it is a bit difficult. Therefore, when interacting with observables, we typically set up a subscription on the consumer side and react to values coming through the pipe. We can easily add caching to the observables by adding publishReplay(1) and refCount.

**#13 Question** List out the differences between ActivatedRoute and RouterState, with reference to Angular 2.

Here are some of the differences between ActivatedRoute and RouterState with reference to Angular 2:-

1. ActivatedRoute consists of the information about a route associated with a component loaded in an outlet. Whereas, RouterState represents the state in which the writer actually is.
2. We need ActivatedRouteSnapchat to traverse all the activated routes. Whereas, during a navigation, after redirects have been applied, the router creates a RouterStateSnapshot.
3. ActivatedRouteSnapshot has old data. When route changes, ActivateRouteSnapshot has data from previous route. Whereas, the RouterState cares about application components, or, to be more specific, about their arrangements.

## #14 Question What would you have in a shared module in Angular 2?

Shared module is used to import the services in both eager and lazy loaded module. We all know that lazy loaded modules create their own branch on the dependency injection tree. Shared module consists of the services that are registered by the angular in the root app injector. For this, we need not import it in the lazy module because lazy loaded modules already have access to the services defined at the root. Components, pipes and directives are also defined in the shared module. Other modules that import the shared module can use it in their templates. This means that the modules can be imported normally in the lazy loaded module. The shared module contains the code that will be used across the applications and featured modules. It also consists of the common template components. “Dumb components” should also be present in the shared module. It typically consists of some common angular modules too. When you are importing the shared module, you will also need to import the module with its providers, because there is no app module in the test.

## #15 Question What do you mean by a structural directive in Angular 2?

Structural directives are used to manipulate DOM in angular. Structural directives are responsible for HTML layout. By adding, removing, or manipulating LMNs in angular, they shape or reshape the structure of DOM. This structural directive is applied to a host element with the help of other directives. The directives then do whatever it is supposed to do with that host element and its descendants. Structural directives can be easily recognized. It can also delay the instantiation of a component or an element. It can also be used for cosmetic effect or manually handling the timing of the loading of components. Structural directives are bound to a template. The two most common structural directives are “ngIf” and “ngFor”. The process occurring in a structural directive is dynamic.

## #16 Question What do you understand by a template variable? How is it used?

A template in Angular 2 is used to instantiate embedded views. A template variable can be accessed in two ways. Either by placing a directive on an element and have the template variable for this embedded view injected into the constructor of the directive using the template variable token, or you can query for the template variable from a component or a directive via the query. A template variable in Angular 2 is a reference to a DOM element or directive within a template. Template variables are used to access the values of DOM element properties. It is declared with the help of “#” and “ref-“as a prefix. For example: – #myVar and ref-myVar. Template variable names cannot be made duplicate as in this way, it might give unpredictable values. The scope of a reference variable is the entire template. It can be used anywhere inside a template. In Angular 2, a component needs to have a view and to define a view, a template variable is used. It allows us to express data and property binding, event binding and template concerns.

**#17 Question** Explain the concept of lazy loading in Angular 2.

Lazy loading is a module which is used to decrease the start-up time. When lazy is used, then our system application does not need to load everything at once. It only needs to load what the user expects to see when the application first loads. The modules which are lazily loaded will only be loaded when the user navigates to their routes. Lazy loading improves the performance of our system applications. It keeps the initial payload small and these smaller payloads lead to faster download speeds. It helps in lowering the resource cost, especially on mobile networks. If a user doesn’t visit a section of the application, they won’t ever download those resources. The concept of lazy loading in angular requires us to format the application in a certain way. All the assets that are to be lazy loaded should be added to its own module. Lazy loading is setup in the main routing file. Lazy loading overcomes the problem of slow loading of applications in their own way which hence improves the loading time of the application.

Lazy loading can be done only in four steps:–

1. Update your route file
2. Install angular-router-loader and add the loader to your webpack configuration file.
3. Define the lazy routes
4. Import the routes to the module.

**#18 Question** What is the difference between constructor and ngOnlnit in Angular js?

The comprehensive comparison that taps into components initialization process is given below:-

1. ngonInit is just a method in a class which structurally is not different to any other method in a class. Whereas, a constructor is a completely different thing. It will be called when an instance of a class is created.
2. A class constructor in angular is used to inject dependencies, which is called constructor injection pattern. Whereas, when ngOnInit is called, it has finished creating a component DOM, injected all required dependencies through constructor and processed input bindings.
3. A constructor is a default method of the class that is executed when the class is instantiated. Whereas, ngOnInit is a life cycle hook called by Angular 2 to indicate that angular is done creating the component.
4. ngOnInit relies on the binding of the component. Whereas, it is not the case when a constructor is used.

**#19 Question** What is the meaning of component lifecycle in Angular 2?

The component lifecycle hooks overview the life cycle sequence and the interfaces. Angular manages the life cycle of a component. Angular creates it, renders it. It can also create and render its children. It also checks when its data-bound properties change. It can even destroy it before removing it from the DOM. The life cycle hook offered by angular provides the visibility into these key life moments and the ability to act when they occur. The components go through an entire set of processes or life cycle right from its initiation to the end of the application.

There are a number of lifecycle hooks which are listed below:–

1. ngOnChanges
2. ngOnInit
3. ngDoCheck
4. ngAfterContentInit
5. ngAfterContentChecked
6. ngAfterViewInit
7. ngAfterViewChecked
8. ngOnDestroy

## #20 Question What is the use of ngForTrackBy directive?

For iterating over a collection in Angular 2, the ngFor directive is used which instantiates a template once per item from the collection. If a data needs to be changed at some point in the collection, then a problem occurs because angular cannot keep a track of items in the collection and has no knowledge of the items which were added or deleted. This results in the deletion of all the DOM elements that are associated with the data and are again created. If the collection is big, then it becomes more complicated because a lot of DOM manipulation occurs which are expensive. So, to solve this problem, a trackBy function is used which takes the index and the current item as arguments and returns the unique identifier for this item.

**#21 Question** List the key components of Angular 2?

**The Angular 2 comprises of the following key components:**

* Module – This is used to break the application into the logical pieces of the program code and each piece of code or module is designed to perform a single and unique task.
* Component – This is used to bring the modules together.
* Templates – This is used to define the Views of an Angular JS application.
* Metadata – This is used to add more data to an Angular JS application.
* Service – This component is used to develop the components, which can be used to share in the entire application.

## #22 Question

### What is a template in Angular 2?

The template in Angular 2 is used to define the views of the AngularJS Application.

## #23 Question How will you convert a string into a percentage?

To convert a string into a percentage format, a percent filter is used.

**#24 Question** Explain component specific hooks?

Below are few component specific hooks in Angular2.

* ngafterContentinit: It initializes the component content
* ngAfterConctentChecked: It checks the binding of the external content.
* ngafterViewinit: It creates the component view.
* ngAfterviewChecked: It checks the bindings of the component’s view.

## #25 Question What is CLI?

CLI is the acronym of Command Line Interface, which can be used to create the Angular JS application.Using CLI, you can also create a unit and end-to-end tests for the Angular application.

## #26 Question What is AOT compilation?

AOT stands for Ahead of Time.It is the compilation in which Angular compiles the components and templates to JavaScript and HTML while developing.

## #27 Question What are Event emitters?

An Event emitter is a class defined in core module that can be used by components and directives to emit custom events.

## #28 Question

### What is Angular @ RouteParams?

The RouteParams are used to map the given URL’s based on the route URLs and they become optional parameters for that route.

**#29 Question** Explain Angular 2 hidden property?

The hidden property in Angular 2 is a special case.

* The property is more powerful and is used to bind any property of the elements.
* It is considered the closest cousin of **ngshow** and **nghide**.
* It sets the display property “display: none”.

## #30 Question Why are decorators used in Angular 2?

In Angular 2, decorators are used as an identifier of class or type of the object that is created by the TypeScript.The Angular 2 identifies the class below decorator call as the definition of the class and extends the decorator specific properties with the class definition.

## #31 Question Explain host decorator in Angular 2?

The host decorators in Angular 2 bind the properties of components with UI element values.The properties inside a component class definition which are decorated with @HostBinding are accessed in a template from the assigned property that is @HostBinding()title=’Our title'( whatever the title is).

## #32 Question What are Pipes in Angular 2?

Pipes in Angular 2 are used in templates in order to convert them into a content that is user-friendly and readable one within the interpolation braces that is {{release| date}}, here the symbol “|” denotes the pipe.

**#33 Question** How can you handle errors in Angular 2 Applications?

The Angular 2 Applications provide with the option of error handling.The errors in Angular 2 can be handled by including the ReactJS catch library and later using the catch function.

* The catch function, which is used after adding the catch library contains the link to the Error handler function.
* And in this error, handler function, the errors are sent to the error console, and also the errors are thrown back to continue the execution.
* So, whenever an error occurs it will be redirected to the error console of the web.

## #34 Question Can you automate porting Angular 1 code to Angular 2?

No, currently there is not any such tool available that ports the Angular 1 code to the Angular 2 code.  
In the process of porting, the Angular 1 code to Angular 2, the side by side manual conversion of Angular 1 directives to the Angular 2 components takes place because they are two different frameworks and hence requires different approaches to solve the same problem.

## #35 Question Which module does is required for every Angular 2 app?

**AppModule**is required for every Angular 2 app.

## #36 Question What is .angular-cli.json. Where can I find it.

**angular-cli.json** is used to configure a project in angular2. You can find it in the root folder of your angular2 Project.

## #37 Question Is Angular Modules and ES modules are the same?

No, Both are different.

**Question # 1) What is Angular 2?**

Answer # AngularJS is by far the most popular JavaScript framework available today for creating web applications.  
And now Angular 2 and TypeScript are bringing true object oriented web development to the mainstream, in a syntax that is strikingly close to Java 8. The main features of Angular 2 are Components, TypeScript, Services.

**Question # 2) What is Angular 2 and TypeScript?**

Answer # Angular 2 is built with features of ES6 (and ES7), Web Components in mind, and targeting evergreen browsers.  
TypeScript is a typed super set of JavaScript which has been built and maintained by Microsoft and chosen by the AngularJS team for development.

**Question # 3) Why do we use Angular 2?**

Answer # Angular 2 is a structural framework for dynamic web apps.  
With AngularJS, designers can use HTML as the template language and it allows for the extension of HTML’s syntax to convey the application’s components effortlessly.  
Angular makes much of the code you would otherwise have to write completely redundant.

**Question # 4) Is ES6 Typescript?**

Answer # ES6 (also called ES2015) is the next iteration of JavaScript, but it does not run in today’s browsers.  
There are quite a few transpilers that will export ES5 for running in browsers.  
TypeScript provides an optional typing system while pulling in features from future versions of JavaScript (ES6 and ES7).

**Question # 5) What is the use of typescript?**

Answer # TypeScript is a strict superset of ECMAScript 2015, which is itself a superset of ECMAScript 5, commonly referred to as JavaScript.  
As such, a JavaScript program is also a valid TypeScript program, and a TypeScript program can seamlessly consume JavaScript.

**Question # 6) What are the key components of Angular 2?**

Answer # Angular 2 has the components like Modules, Component, Templates, Metadata, and Service.  
The Angular 2 Component combines the AngularJS Directive, Controller, and Scope.  
Components are the fundamental building blocks of Angular applications.  
They display data on the screen, listen for user input, and take action based on that input.

**Question # 7) What are the modules in angular 2?**

Answer # In Angular, a module is a mechanism to group components, directives, pipes and services that are related,  
in such a way that can be combined with other modules to create an application.  
An Angular application can be thought of as a puzzle where each piece (or each module) is needed to be able to see the full picture.

**Question # 8) What is a component in angular 2?**

Answer # Component decorator allows you to mark a class as an Angular component and provide additional metadata that determines how the component should be processed,  
instantiated and used at runtime. An Angular application is a tree of Angular components. Angular components are a subset of directives.

**Question # 9) In Angular 2 Module consist of what arrays?**

Answer # Angular 2 Module consist of Bootstrap Array, Export Array, Import Array.

**Question # 10) What is Bootstrap array in Angular 2?**

Answer # In [Angular 2](https://thinkster.io/tutorials/learn-angular-2), Bootstrap array is used to inform Angular JS which components need to be loaded, so that its functionality can be accessed in the application.

**Question # 11) What is Export array in Angular 2?**

Answer # Export array is used to export components, directives, and pipes which can then be used in other Angular JS modules.

**Question # 12) What is Import array in Angular 2?**

Answer # The import array can be used to import the functionality from other Angular JS modules.

**Question # 13) What are angular 2 directive?**

Answer # The three types of directives in [Angular 2](https://codingcompiler.com/angular-2-interview-questions/) are attribute directives, structural directives, and components.  
Structural directives change the DOM layout by adding and removing DOM elements.  
Attribute directives change the appearance or behavior of an element.

**Question # 14) What is Link function in Angularjs directive?**Answer # Directives that want to modify the DOM typically use the link option to register DOM listeners as well as update the DOM.  
It is executed after the template has been cloned and is where directive logic will be put.

link takes a function with the following signature, function link(scope, element, attrs, controller, transcludeFn) { … }, where:

scope is an AngularJS scope object.  
element is the jqLite-wrapped element that this directive matches.  
attrs is a hash object with key-value pairs of normalized attribute names and their corresponding attribute values.  
controller is the directive’s required controller instance(s) or its own controller (if any). The exact value depends on the directive’s require property.  
transcludeFn is a transclude linking function pre-bound to the correct transclusion scope.

**Question # 15) What is a DOM in Angular 2?**

Answer # DOM stands for Document Object Model. Angular 2 directives are used to bind application data to the attributes of HTML DOM elements.

**Question # 16) What is the meaning of Ng in Angular 2?**

Answer # ng is the core module and stands for Angular. The ng module is loaded by default when an AngularJS application is started.  
The module itself contains the essential components for an AngularJS application to function.

**Question # 17) What is the use of custom directive in Angular 2?**

Answer # Angular 2 Custom directives are used in [AngularJS](https://codingcompiler.com/angularjs-interview-questions/) to extend the functionality of HTML. Custom directives are defined using “directive” function.  
A custom directive simply replaces the element for which it is activated.

**Question # 18) What is Routing in Angular 2?**

Answer # [Routing](https://angular.io/guide/router)is used to separate different parts of our app, generally by using the URL to denote our location.

**Question # 19) What are the main components of Routing in Angualr 2?**

Answer # There are four main components that can be used to configure**routing in Angular 2**:

Routes: – It describes our application’s routes.  
Router Imports: – It uses to import our application’s Routes.  
RouterOutlet: – It is a “placeholder” component that gets expanded to each route’s content.  
RouterLink: – IT is used to link to application’s routes.

**Question # 20) What is Tsconfig JSON in Angulr 2?**

Answer # The presence of a tsconfig.json file in a directory indicates that the directory is the root of a TypeScript project.  
The tsconfig.json file specifies the root files and the compiler options required to compile the project.

**Question # 21) what is ngOnChanges in Angular 2?**

Answer # ngOnChanges is called right after the data-bound properties have been checked and before view and content children are checked  
if at least one of them has changed. The changes parameter contains the changed properties.

**1. What is Angular 2?**

Angular 2 is the next version of Google’s massively popular MV\* framework for building complex applications in the browser (and beyond).

Angular 2 comes with almost everything we need to build a complicated frontend web or mobile apps, from powerful templates to fast rendering, data management, HTTP services, form handling, and so much more.

AngularJS is by far the most popular JavaScript framework available today for creating web applications. And now Angular 2 and TypeScript are bringing true object oriented web development to the mainstream, in a syntax that is strikingly close to Java 8.

**2. What is Advantages of Angular 2?**

There is many more advantage of Angular 2.

1. The Angular 2 has better performance.  
2. The Angular 2 has more powerful template system.  
3. The Angular 2 provide simpler APIs, lazy loading and easier to application debugging.  
4. The Angular 2 much more testable.  
5. The Angular 2 provides to nested level components.  
6. The Angular 2 execute run more than two programs at the same time.  
7. Angular1 is controllers and $scope based but Angular2 is component based.  
8. The Angular2 structural directives syntax is changed like ng-repeat is replaced with \*ngFor etc.  
9. In Angular2, local variables are defined using prefix (#) hash.

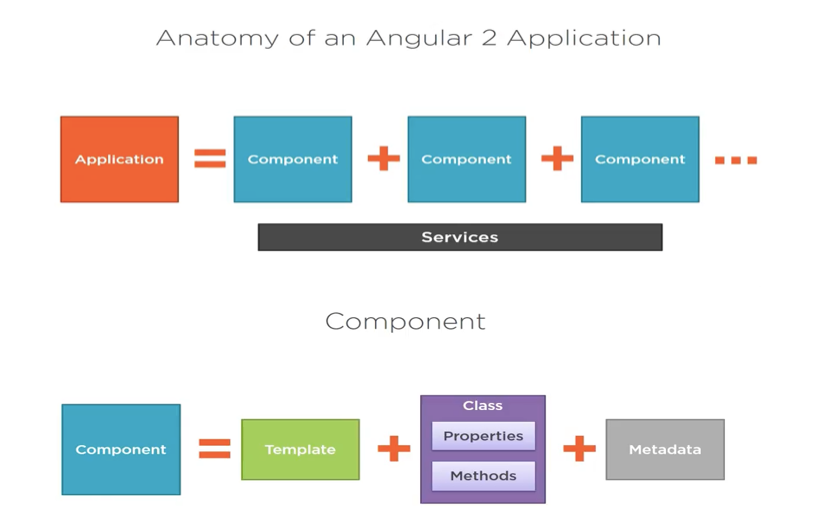
The Angular 2 architecture diagram identifies the eight main building blocks as.

1. Module  
2. Component  
3. Template  
4. Outpouts  
5. Data Binding  
6. Directive  
7. Service  
8. Dependency Injection

The Angular 2 framework consists of several libraries, the some of them working as core and some are optional.

**3. What is the anatomy of angular 2 application?**

An application is consists of a set of components, and some services, each component is comprised of a template,Classes and metadata (look to the following figure ).



**4. what is an Angular 2 component?**

In Angular 2, “everything is a component.” Components are the main way we build and specify elements and logic on the page, through both custom elements and attributes that add functionality to our existing components.

Each component is comprised of a template, which is the HTML for the user interface. Add to that a class for the code associated with a view. The class contains the properties and methods, which perform actions for the view,A component also has metadata, which provides additional information about the component to Angular.

**5. Can we write both Angular 1 and Angular 2 codes in a single project?**

1. Angular frameworks provide the support of mixing code of Angular 1 and Angular 2 in the same application.  
2. We can write the mixing components of Angular 1 and Angular 2 in the same view.  
3. We can inject services across frameworks of Angular 1 and Angular 2 in the same application.  
4. Both Angular 1's and Angular 2's data binding works across frameworks in the same view.

**6. What is the languages that you can use to build angular 2 application?**

\* ECMAScript, or ES.

–ES 3 is supported by older browsers.  
–ES 5 is the version currently supported by most modern browsers.  
–The ES 6 specification was recently approved and renamed ES 2015(Most browsers don't yet support ES 2015).

\* TypeScript

–TypeScript is the superset of JavaScript and must be transpiled.  
–TypeScript has great tooling  
–Inline documentation.  
– Syntax checking.  
– Code navigation.  
– Advanced refactorings.

**7. What is ECMAScript ES5/ES6?**

The ECMAScript is a scripting language which is developed by Ecma International Org.

Currently ECMAScript available in multiple versions that are ES5 and ES6 and both of versions fully supported to Chrome, Firefox, Opera, Safari, and IE etc.

**8. What is Traceur compiler?**

The Traceur is a JavaScript compiler. The Traceur compiler is very popular now days use to allow use to use the features from the future. This compiler is fully supported to ES5, ES6 and also to vNext.

The main goal of Traceur compiler is to inform to design of new JavaScript features and wrote the programming code of new efficient and good manners.

**9. Differentiate between Angular 2 Component Constructor vs. OnInit event?**

The constructor is a typescript feature. The constructor is only related to class instantiation and it’s nothing to do with Angular 2 and it is use to some initialization processing with respect to class hierarchies for the newly created instance.

The ngOnInit event is an Angular 2 life-cycle event/ method that are called after the first ngOnChanges. ThengOnInit method is use to parameters defined with @Input otherwise the constructor is OK.

**10. What is Angular 2 Components Life cycle?**

In Angular 2 components life-cycle, there are several events occur to complete this life-cycle.

|  |  |
| --- | --- |
| **Events** | **Description** |
| ngOnChanges | Before Ng on init event, the data-bound input property value changes. |
| ngOnInit | After the first ngOnChanges event, the ngOnInit event fire. |
| ngDoCheck | During every Angular change detection cycle ngDoCheck event fire. |
| ngAfterContentInit | After projecting content into the component ngAfterContentInit event fire. |
| IngAfterContentChecked | After every check of projected component content the ngAfterContentChecked event fire. |
| ngAfterViewInit | After initializing the component's views and child views the ngAfterViewInit event fire. |
| ngAfterViewChecked | After every check of the component's views and child views the ngAfterViewChecked event fire. |
| ngOnDestroy | Just before Angular destroys the directive or component the ngOnDestroy event fire. |

**11. What is Angular 2 Directives?**

\* Angular 2 directives meta-data annotation is used to register the directives.  
\* The directives are used to add behaviour to existing DOM elements.  
\* The directives are used to design a reusable component.  
\* More than one directive is used per DOM element.  
\* The directive does not have @View etc.

***Example***

import {Component, View} from 'angular2/core'';

@Component({

selector: 'user-detail'

})

@View({

template: "<div> <h1>{{userName}}</h1> <p>{{phone}}</p>"

})

class userDetail {

constructor(public userName: string, public phone: string) {}

}

<user-detail></user-detail></div>

**12. What is Angular 2 Template?**

A template is a HTML view that tells Angular 2, how to render your components in the views. The Angular 2 templates are very similar to Angular 1 but Angular 2 have some small syntactical changes.

You can see the changes as below

\* {}: Is use to rendering the HTML elements.  
\* []: Is use to binding properties.  
\* (): Is use to handling your events.  
\* [()]: Is use to data binding.  
\* \*: Is use to asterisk Operations like \*ngFor="#item of items”

**13. Differentiate between Angular 2 components vs directives?**

Angular 2 components vs directives

|  |  |
| --- | --- |
| **@Components** | **@Directive** |
| @Component meta-data annotation is used to register the components. | @Directive meta-data annotation is used to register the directives. |
| The components are used to create UI widgets. | The directives are used to add behavior to existing DOM elements. |
| The components are used to split to application into smaller parts. | The directives are use to design a reusable components. |
| Only one component is used per DOM element. | More than one directive are used per DOM element. |
| In the components, @View, template and templateUrl are mandatory in the components. | The directive do not have @View etc. |

***Example for using Component:***

import {Component, View} from 'angular2/core';

@Component({

selector: 'hello-world'

})

@View({

template: "<h1>Hello {{angular}}</h1>"

})

class hello {

constructor(public angular: string) {}

}

<hello-world></hello-world>

***Example for using Directive:***

import {Component, View} from 'angular2/core'';

@Component({

selector: 'user-detail'

})

@View({

template: "<div> <h1>{{userName}}</h1> <p>{{phone}}</p>"

})

class userDetail {

constructor(public userName: string, public phone: string) {}

}

<user-detail></user-detail>

**14. What is Angular 2 @Inputs?**

Angular 2 component is the core components of applications but we need to know how to pass data into components to dynamically.

For the same, we need to define an input (use like @Input decorator) for a component.

How to pass data into components? We can see the below example for passing the user data in to the components.

***For Example,***

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

@Component({

selector: “user-info”,

template: “<div> Hello, This is {{ userInfo.name}}</div>”

})

export class UserInfo {

@Input() userInfo;

constructor() { }

}

<user-info [userInfo]="currentUser"></user-info>

The components is use to render the user information on the view.

**15. What is Angular 2 Outputs?**

In Angular 2, if we want to bind an event on an element, we can use the new Angular 2 events i.e.

**<button (click)="addUser()">Click</button>**

The method addUser() will be called when user clicked on button.

**16. What happen if you want to create a custom event?**

Now come to the outputs, if we want to create our custom event in Angular 2 that time we will use to new @Output decorator.

Example

import { Component} from 'angular2/core';

import { bootstrap} from 'angular2/platform/browser';

@Component({

selector: 'my-app',

providers: [Service],

template: '<div>Hello my name is {{name}}!</div>'

})

class MyApp {

constructor(service: Service) {

**this**.name = service.getName();

setTimeout(() => **this**.name = 'Saurabh Samir,', 1000);

}

}

class Service {

getName() {

**return** 'Hello';

}

}

bootstrap(App);

In the above example, we will need to import Output and Event-Emitter to create our new custom event.

import { Component , Output, EventEmitter} from 'angular2/core';

import { bootstrap} from 'angular2/platform/browser';

@Component({

selector: 'my-app',

providers: [Service],

template: '<div>Hello my name is {{name}}!</div>'

})

class MyApp {

constructor(service: Service) {

**this**.userClicked.emit(**this**.user);

**this**.name = service.getName();

setTimeout(() => **this**.name = 'Saurabh Samir,', 1000);

}

}

class Service {

getName() {

**return** 'Hello';

}

@Output() userClicked = **new** EventEmitter();

}

bootstrap(App);

Now when we are using the components anywhere in our application, we can bind the our custom event i.e.

**<my-app (userclicked)="userClicked($event)"></my-app>**

**17. What is Angular 2 components css styles and styleUrls?**

The Angular 2 components styling can be

1.Inline styles  
2.CSS Style URLs and  
3.Template inline styles

The Angular 2 components allow us to define both type of css that are inline css and styleUrls and the detail about it as given below.

Components Inline CSS Styles

@Component({

selector: 'customers',

templateUrl: 'customers.html',

styles: [

.customer {

padding:0.3em;

background-color: #f5f5f;

box-shadow: inset 1px 1px 1px rgba(0,0,1,0.2);

border-radius:1px;

border: solid 1px #c1c1c;

}]

})

Components CSS styleUrls

@Component({

selector: 'customers',

templateUrl: 'customers.html',

styleUrls: ['customers.css']

})

//customers.css

.customer {

padding:0.3em;

background-color: #f5f5f;

box-shadow: inset 1px 1px 1px rgba(0,0,1,0.2);

border-radius:1px;

border: solid 1px #c1c1c;

}

Components Template inline css styles

<style>

.customer {

padding:0.3em;

background-color: #f5f5f;

box-shadow: inset 1px 1px 1px rgba(0,0,1,0.2);

border-radius:1px;

border: solid 1px #c1c1c;

}

</style>

<div class="customer">

<div (click)="toggle()">

{{IsVisible ? **true : false** }} {{CustomerUID}}

</div>

<div [hidden]="!IsVisible">

<customer></customer>

</div>

</div>

**18. What is ECMAScript ES6/ES7? Breifly explain both. Which version is supported by Angularjs 2.**

ECMAScript is a standard for modern scripting-language specification. Initially, it was JavaScript, now its being changing to ECMAScript.

***6th Edition – ECMAScript 2015:***

ECMAScript 2015 is the 6th Edition. This version has significant contribution for:

\* Class Inheritance  
\* Iterators for loop  
\* Python-style generators and generator expressions  
\* Advanced set for collections  
\* number and math enhancements  
\* Promises  
\* Metaprogramming for virtual objects and wrappers and so on.

Although, ES6 still does not support browser. Therefore, compiler such as Traceur are required to compile EC6 code to pure javascript on the fly in browser.

***7th Edition – ECMAScript 2016:***

The 7th edition, officially known as ECMAScript 2016, has been finalized in June 2016.

**Which version is supported by Angular2?**

Angular has used 6th Edition – ECMAScript 2015 (Es6). However, there are configuration available to try some experimental feature from 7th Edition – ECMAScript 2016 (Es7).

Inheritance of component has become so easy. Now Angular 2.0 can be developed using Object oriented thinking. Inheritance has been possible for javascript. A number of libraries has been developed to support this concept. However, with ES2015 all those nonstandard abstractions can be got rid of. As ES2015 defines an easier inheritance.

**19. Angular 2 toggle button?**

***Syntax:***

**<button (click)="toggle()">Toggle Button</button>**

*//The Example for Angular2 Toggle as given below!*

//Step-1

//Import root component.

import {Component, View, CORE\_DIRECTIVES} from 'angular2/angular2'

//Step-2

//Component

@Component({

selector: 'toggle-app',

bindings: []

})

//Step-3

//View template.

@View({

template: `

<div>

<button (click)="toggle()">Toggle Button</button>

</div>

<div class="border">

<div **\*ng-if**="isActive">

<h1>Hello Angular 2, Toggle Button.</h1>

</div>

</div>

<p>Status(isActive): {{isActive}}</p>

`,

directives: [CORE\_DIRECTIVES]

})

//Toggle class App for active and hide div.

export class App {

isActive: bool = **true**;

toggle() {

this.isActive = **!this**.isActive;

}

}

**20. Angular 2 ngif else expression?**

***Syntax:***

1.<div \*ng-if="your-condition">...</div>

2.<div \*ngif="your-condition">...</div>

3.<div template="ngIf your-condition">...</div>

4.<template [ngif]="your-condition">

<div>...</div>

</template>

***The example as given below.***

//Import root component.

import {Component, View, CORE\_DIRECTIVES} from 'angular2/angular2'

//Component

@Component({

selector: 'toggle-app',

bindings: []

})

//View template.

@View({

template: `

<div>

<button (click)="toggle()">Toggle Button</button>

</div>

<div class="border">

<div \*ng-if="isActive">

<h1>Hello Angular 2, Toggle Button.</h1>

</div>

</div>

<p>Status(isActive): {{isActive}}</p>

`,

directives: [CORE\_DIRECTIVES]

})

export class App {

isActive: bool = **true**;

toggle() {

**this**.isActive = !**this**.isActive;

}

}

**21. What is Angular 2 RouteParams?**

The Route Params :- The route parameter is used to map given URL's parameters based on the rout URLs and It is an optional parameters for that route.

***Syntax :-***

**params : {[key: string]: string}**

***Example,***

@RouteConfig([

{path: '/employ/:id', component: employe, name: 'emp'},

])

Router-outlet directive :- Router-outlet directive is used to render the components for specific location of your applications. Both the template and templateUrl render the components where you use this directive.

***Syntax :-***

**<router-outlet></router-outlet>**

Router-link directive :- Router-link directive is used to link a specific parts of your applications.

***Syntax :-***

**<router-link></router-link>**

***Example,***

<a [router-link]="['/AboutMe']">About Me</a>

The Route-Config :- The route config is used to map components to URLs.

***Syntax :-***

@RouteConfig([

{path: '/', component: Home\_Component, as: 'Home'},

{path: '/AboutMe', component: AboutMe\_Component, as: 'AboutMe' }

{path: '/ContactMe', component: ContactMe\_Component, as: 'ContactMe' }

])

**22. What is Angular 2 hidden property?**

Angular 2 [hidden] is a special case binding to hidden property.

It is closest cousin of ng-show and ng-hide .  
It is more powerful and use to bind any property of elements. Both the ng-show and ng-hide are used to manage the visibility of elements using ng-hide css class. It is also set the display property "display:none".

All the above features are supported in Angular 2 but added some extra feature like animations etc.

**Example for Angular 2**

<div [hidden]="!active">

Hello, This is active!

</div>

**Example for Angular 1**

<div ng-show="active">

Hello, This is active!

</div>

**23. What is Angular 2 templateUrl and styleUrls?**

Angular 2 templateUrl :- The templateUrl is a function which returns HTML template.

Angular 2 styleUrls:- The styleUrls is a component which use to write inline styles, style Urls and template inline style.

The example as given below using templateUrl and styleUrls.

import {Component} from 'angular2/core';

@Component({

selector: 'app'

templateUrl: 'index.html',

styleUrls: ['main\_style.css']

})

export class App\_Component { }

**24. What is Angular 2 events?**

Events in Angular 2 use the parentheses notation in templates, and trigger methods in a component’s class. For example, assume we have this component class:

@Component(...)

class MyComponent {

clicked(event) {

}

}

And this template:

**<button (click)="clicked()">Click</button>**

Our clicked() method will be called when the button is clicked.

**25. How to import css using System import?**

***Syntax :***

System.import('./app/bootstrap/css/boots-trap.css!').then(() => {

System.import('./app/main-app.css!');

});

**26. How to Load external css style into Angular 2 components?**

The styles or styleUrls should only be used for css rules and It is affect the style of the template elements.

This is the best approaches to add styles directly to the components and the view encapsulation is set per component. It is use for some situations.

An example to add external styles to components.

@Component({

selector: 'app',

templateUrl: 'app/login.html',

styleUrls: [

'app/app.css',

'app/main.css'

],

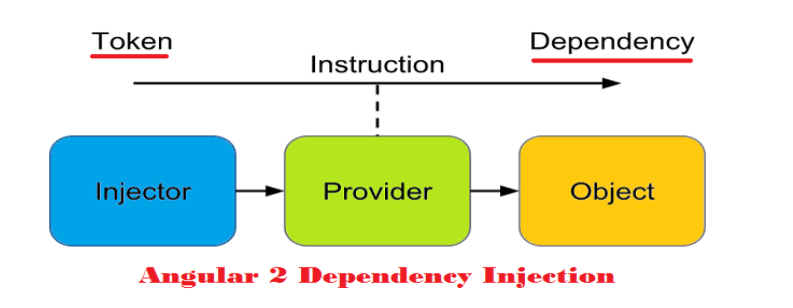
encapsulation: ViewEncapsulation.None,

})

export class Component {}

**27. What is Dependency Injection (DI) in Angular 2?**

Angular 2 Dependency Injection consists of three things.



1.Injector  
2.Provider  
3.Dependency

Injector :- The injector object use to create instances of dependencies.

Provider :- A provider is help to injector for create an instance of a dependency. A provider takes a token and maps that to a factory function that creates an object.

Dependency :- A dependency is the type of which an object should be created.

**28. What is Inter-component communications?**

Component communication can and should be implemented in a loosely coupled manner. A component can declare input and output properties. To pass the data from a parent to a child component, the parent binds the values to the input properties of the child. The child has no need to know who provided the values; it just knows what to do with them.

If a component needs to pass the data to the outside world, it emits the events via the output property. (Emits to whom? It’s none of the component’s business. Whoever is interested can create a listener to the custom component’s event.)

This mechanism allows us to treat components as black boxes, that can get values in or send data out.Illustrating one of the implementations of the Mediator design pattern in Angular 2.

**29. Why TypeScript?**

TypeScript is a superset of JavaScript but like Java it allows us to define new types. Declaring variables with types rather than the generic var opens the door to new tooling support, which we will find to be a great productivity enhancer. TypeScript comes with a static code analyzer, and as we enter code in our TypeScript-aware IDE (WebStorm/IntelliJ Idea, Visual Studio Code, Sublime Text, etc.) we’re guided by context sensitive help suggesting the available methods in the object or types of the function argument. If we accidentally use an incorrect type, the IDE will highlight the erroneous code.

Even if our TypeScript application uses a third-party library written in JavaScript, we can install a type definition file (having the extension .d.ts), containing type declarations for this library. Type declarations for hundreds of popular JavaScript libraries are freely available, and we can easily install them with Typings, a TypeScript Definition Manager. Imagine that we want to use jQuery (written in JavaScript) from our TypeScript code. The type-definition files for jQuery will contain declarations (with types) of all jQuery APIs so our IDE can prompt us with which types to use, or highlight any erroneous code.

**30. What is Uncaught reference error:System is not defined?**

We can Try to include the System JS file in our HTML header file and resolve this error.

**Include SystemJS Script.**

<script src=<https://jspm.io/system@0.18.17.js>></script>

The Example for detail as given below.

**index.html file**

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="style.css" />

<script src=<https://jspm.io/system@0.18.17.js>></script>

<script src=<https://code.angularjs.org/2.0.0-alpha.36/angular2.min.js>></script>

<script>

System.config({

paths: {

'main.js':'main.js'

}

});

System.import('main.js');

</script>

</head>

<body>

<app></app>

</body>

</html>

**main.js file**

import {Component, View, bootstrap} from 'angular2/angular2';

@Component({

selector: 'app',

bindings: [Service]

})

@View({

template: '{{greeting}} I am Samir!'

})

class App {

constructor(service: Service) {

this.greeting = service.greeting();

setTimeout(() => this.greeting = 'Hi,', 2000);

}

}

class Service {

greeting() {

return 'Welcome you!';

}

}

## 1. Angular 4 vs. Angular 2

There is no path breaking difference between angular 2 and angular 4. Angular 4 is simply the next version of Angular 2. The underlying concepts are same. If you know Angular 2, you can easily switch to Angular 4.  
  
Angular 4 is **backward compatible** with Angular 2 for most Applications.   
  
There are some under the hood changes to reduce the **size of the AOT(Ahead-of-Time)** compiler generated code. Migrating to Angular 4 may reduce production bundle.  
  
Angular 4 is **faster** than Angular 2. The apps developed in Angular 4 are five times faster and smaller as compared to Angular 2.  
  
**TypeScript 2.1 and 2.2 compatiblity**. Before Angular 4, only TypeScript 1.8 was supported. So with Angular 4, we have all new features of Typescript available.  
  
Unlike Angular 2, the **animations** have been pulled out of @angular/core and are moved into their package in the Angular 4. If you don't use animations, this extra code will not end up in the production bundle.  
  
We can now use new if/else style syntax with **\*ngif structural directive**. In Angular 2, it was not possible to use "else" statement with nglf, but now it has been made possible in Angular 4.   
  
Angular 4 has introduced a new **titlecase pipe**. It changes the first letter of each word into uppercase  
  
"**As**" keyword is the new addition to the template syntax to simplify the "let" syntax  
  
Integration of **Angular Universal** - The integration of Angular Universal permits developers to run Angular on a server.  
  
The **template tag is now deprecated**: you should use the "ng-template" tag instead as Angular has its own template tag: ng-template now.

## 2. Why Angular 4? What’s New in Angular 4?

**Router ParamMap**  
Starting from version 4, it is possible to use **paramMap** to get the route- and query-parameter from a related route. The use of **Map** brings advantages in terms of type security.   
The old had an unsecure type (type params = {[key: string]: any}) and the value could have all possible types.   
The new way provides either a string, or an array of strings related to the used method (paramMap.get(): string and paramMap.getAll(): string[])  
**Animations**  
Earlier all the functions of animations were the part of @angular/core module, which means the code were always included in the bundle even if animations were not used.  
In Angular 4, Animations are to be provided in the module BrowserAnimationsModule from @angular/platform-browser/animations. This avoids creating bundles with unnecessary large sizes.   
**ngif**  
We can now use new if/else style syntax with \*ngif structural directive.  
**NgComponentOutlet**  
To build and produce components dynamically at runtime involved relatively much programming work. With the introduction of \*ngComponentOutlet-Directive in Angular 4, it is possible to build dynamic components in a declarative way.    
**TypeScript 2.1/2.2**  
We have the support of most latest TypeScript versions in Angular 4 which helps in improving the speed of ngc-Compiler.  
**Angular Universal**  
With Angular Universal, it is possible to render Angular applications on the web server. With that, websites can be optimized better for search engines as JavaScript is no longer necessary for initially rendering the page content.

**3. What is the use of Interceptors?**

Interceptors are used to intercept and/or mutate outgoing requests or incoming responses. It can be really useful for features like caching and logging.  
  
Interceptors can be used on multiple scenarios, i.e. setting the Origin for each outgoing request, adding authentication token to every request etc.

**4. What is Angular?**

Angular is a framework for building client applications in HTML, CSS and Javascript(or language like Typescript which compiles into Javascript).

**5. Why do we need Angular?**

We can develop application using Javascript and Jquery. But as the application become complex, code in Javascript and Jquery become difficult to maintain. We then require to structure the application code properly by incorporating object oriented features. That is why a framework like Angular has been evolved to make web application development and maintenance faster and easier. The application in Javascript is hard to test. Applications build in Angular are easily testable.

* Angular gives our application a clean object oriented structure that is easy to understand and easy to maintain.
* Angular come with a lot of utility code that can be reused in various applications.
* Applications build in Angular are easily testable.

**6. What is Node js?**

It is basically a runtime environment for executing Javascript code outside the browser.

**7. Angular CLI**

Angular CLI stands for Command-line Interface.

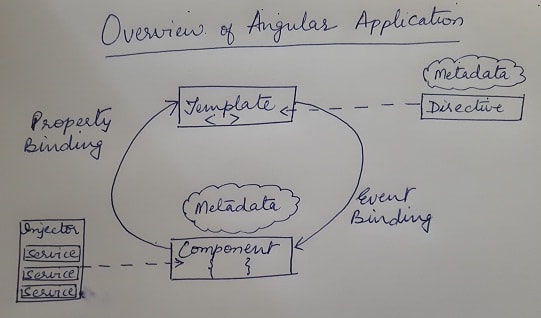
**8. What is TypeScript?**

TypeScript is a superset of Javascript. So any valid Javascript code is also a TypeScript code. TypeScript has many additional features that Javascript doesn't offer:

* **Strong Type** - You can specify variable type at the time of declaring a variable which makes code easier to maintain and catching errors become easier. Although this feature is optional.
* **Object Oriented features** - TypeScript brings many object oriented features which have been missing in Javascript like classes, interfaces, constructor, access modifier, properties etc.
* In TypeScript, We can catch **error** at compile time instead of runtime
* With TypeScript, We also get **intellisense** in the code editor.

Browser don't understand TypeScript code, TypeScript compiles into JavaScript at the time when we build the application.

**9. Architecture of Angular Apps**

Angular applications are designed by composing HTML templates with Angularized markup, writing component classes to manage those templates, adding application logic in services, and boxing components and services in modules.  


**Building blocks of Angular Apps**

**Components** - Angular embraces component based architecture which allows us to work with smaller and maintainable piece of code that can be reused at several places. Each Angular App has one or more components. A component controls a patch of screen called a view. It encapsulates Data, HTML template and Logic for a view (area of the screen that the users see).   
  
Every application has a root component that we call as App component.  
  
**Modules** - It is container for group of related components, i.e. in an employee module, we can have components for displaying employees details. Every Angular app has at least one NgModule class, the root module, conventionally named AppModule.  
  
**Templates** - You define a component's view with its companion template. A template is a form of HTML that tells Angular how to render the component.  
  
**Metadata** - Metadata tells Angular how to process a class.  
  
**Example**  
  
In fact, CourseListComponent really is just a class. It's not a component until you tell Angular about it.  
  
To tell Angular that CourseListComponent is a component, attach metadata to the class.  
  
In TypeScript, you attach metadata by using a decorator. Here's some metadata for CourseListComponent:

@Component({  
  selector:    'app-course-list',  
  templateUrl: './course-list.component.html',  
  providers:  [ courseService ]  
})  
  
export class CourseListComponent implements OnInit {  
/\* . . . \*/  
}

**Directives** - A directive is a class with a @Directive decorator. A component is a directive-with-a-template; a @Component decorator is actually a @Directive decorator extended with template-oriented features.  
  
Directives either alter the layout structure (for example, ngSwitch) or modify aspects of DOM elements and components (for example, ngStyle and ngClass).  
  
Of course, you can also write your own directives. Components such as HeroListComponent are one kind of custom directive.  
  
**Services** - A service is typically a class with a narrow, well-defined purpose. Components are big consumers of services. Component classes should be lean. They don't fetch data from the server, validate user input, or log directly to the console. They delegate such tasks to services. A component's job is to enable the user experience and nothing more. It mediates between the view (rendered by the template) and the application logic.   
  
Angular factor your application logic into services and make those services available to components through dependency injection.  
  
**Dependency injection** is a way to supply a new instance of a class with the fully-formed dependencies it requires. Most dependencies are services. Angular uses dependency injection to provide new components with the services they need.

**10. Steps to follow in order to use component.**

* Create a Component
* Register it in a module
* Add an element in an HTML markup

**1. What is Angular 2?**

**Answer:**  
  
Angular 2 is a JavaScript framework for developing desktop as well as mobile application. It is an open source framework which is built by the developer at Google. Angular 2 helps us to develop web application in HTML and JavaScript. It follows simpler, faster and modular design approach.   
  
Angular 2 framework is written in ECMAScript 6 (ES6). The ES6 compiler manages the versioning related problem.

**2. What are the advantages of using Angular 2 over Angular JS?**

**Answer:**  
  
**Following are the advantages of using Angular 2 over Angular JS:**

* Angular JS uses only JavaScript but Angular 2 provides the possibility to use different languages like TypeScript, ES5, ES6, Dark etc.
* Angular JS was not built with mobile support in mind, where Angular 2 is mobile oriented.
* Angular 2 based on components, and it provides better performance than Angular JS.
* The use of dependency injection is enhanced in Angular 2.
* Angular 2 has the flexible routing with lazy loading features.
* Angular 2 is faster

**3. What is routing in Angular 2?**

**Answer:**  
  
Routing in Angular 2 is a mechanism for navigating between pages and displaying an appropriate component/page on browser. The Router helps mapping application URL to application components.    
  
**Following are some of the main components used to configure routing:**

* Routes
* Router Imports
* RouterOutlet
* RouterLink

## 4. What is Module in Angular 2?

**Answer:**  
  
In Angular 2, Module allows to put logical boundaries in our application. It is a fundamental feature of Angular 2 that groups related components, directives, pipe and services together.  
  
Every Angular 2 application has at least one module, the root module, conventionally named AppModule. Some important features like lazy loading are done at the Angular Module level.

## 5. What is Component in Angular 2?

**Answer:**  
  
In Angular 2 application everything is component. Components are a logical piece of code for Angular JS application. Each component is mainly used to build HTML elements and provides logical boundary of functionality for the Angular 2 application. The components encapsulate all the logic, allowing you to reuse them across your application.  
  
**A Component consists of the following:**  
  
**Template:** It is used to render the view for the application.  
  
**Class:** It is like any class of c and c++  
  
**Metadata:** It has the extra data defined for the Angular class. It is defined with a decorator.

#### **Example:**

import { Component } from '@angular/core';  
  
@Component ({  
   selector: 'Home',  
   template: ` <div>  
      <h1>{{homeTitle}}</h1>  
      <div>To Home Page</div>  
   </div> `,  
})  
  
export class HomeComponent {  
   homeTitle: string = 'Welcome';  
}

**6. Explain the life cycle hooks of the Angular 2 application.**

**Answer:**  
  
The lifecycle events of Angular 2 component/directive are managed by @angular/core. It creates the component and renders it, processes changes when it’s data-bound properties change, and then destroys it before removing its template from the DOM. Angular has a set of lifecycle events. These events can be tapped and perform operations when required. The constructor executes prior to all lifecycle events. The constructor is meant for light weight activities and usually used for dependency injection.  
  
**Following events are the applicable for both component and directive.**

* **ngOnChanges:** It is called when the value of a data bound property changes.
* **ngOnInit:** This is called after the first ngOnChange events.
* **ngDoCheck:** This is called after every change detection run
* **ngAfterContentInit:** It is called after every component content initialized.
* **ngAfterContentChecked:** It is called after every check of component content.
* **ngAfterViewInit:** This is called after angular initializes the component view’s and child view.
* **ngAfterViewChecked:** This is called after every check of a component's view(s).
* **ngOnDestroy:** It is called just before the component is destroyed.

## 7. What is Lazy loading?

**Answer:**  
  
In lazy loading our application does not load everything at once. It loads only those things what the user expects to see when the app first loads. It helps us decrease the startup time.  
  
Lazy loading enables us to load only the module user is interacting and keep the rest to be loaded at run time on demand.

## 8. How to handle error in Angular 2 application?

**Answer:**  
  
In Angular 2 applications, error handling is done by including the RxJS catch library and then using the catch function.  
  
In the error handler function, we send the error to the console. We also throw the error back to the main program so that the execution can continue.  
  
The catch function contains a link to the Error Handler function.

## 9. What are the event emitters in Angular 2 and how it works?

**Answer:**  
  
EventEmitter is an angular 2 abstraction and its only purpose is to emit events in components.   
  
Any change occurred in the component always gets propagated from the current component to all its children in hierarchy. If the change from one component needs to be reflected to any of its parent component in hierarchy, we can emit the event by using Event Emitter API.  
  
In short, EventEmitter is used by components and directives to emit custom events.  
  
The EventEmitter class is defined in '@angular/core' module.  
  
@output() somethingChanged = new EventEmitter();  
  
We use somethingChanged.emit(value) method to emit the event.

## 10. What is Pipes in Angular 2?

**Answer:**  
  
Filters in AngularJS 1.x are known as Pipes in Angular 2.  
  
Pipes allow us to change the data inside the template. Normally, a pipe takes the data and transforms this input to the desired output. There are many built-in pipes in Angular 2.   
  
**There are multiple built-in pipes in Angular 2.**

* DatePipe
* UpperCasePipe
* LowerCasePipe
* CurrencyPipe
* PercentPipe

#### **Example:**

<div>    
    <h4>Upper Case Pipe Example</h4>    
    My name is {{name | uppercase}}!    
</div>

## 11. What is dependency injection?

**Answer:**  
  
Dependency injection is an important application design pattern and Angular has its own dependency injection framework. It provides the ability to add the functionalities to the components at runtime.   
  
**Angular 2 Dependency Injection consists of three things.**

* **Injector:** The injector object provides us a mechanism by which the desired dependency is instantiated.

* **Provider:** A Provider is a medium by which we register our dependency that need to be injected.
* **Dependency:** The dependency is an object of the class that we want to use.

## 12. What are the differences between Constructors and OnInit?

**Answer:**  
  
**Constructor:**  
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. It is a feature of a class rather than an Angular feature.  
  
**OnInit:**  
ngOnInit is purely there to give a signal that Angular has finished initializing the component. ngOnInit is called just after the constructor call. The constructor should only be used to initialize class members. But when we have tasks related to Angular’s bindings we should use ngOnInit life cycle hook.

#### **Example:**

import {Component, OnInit} from '@angular/core';  
export class App implements OnInit{  
    constructor(){  
    }  
    ngOnInit(){  
    }  
}

## 13.  What is the use of codelyzer in angular 2 applications?

**Answer:**  
  
All the application follow the some coding conventions and guidelines to maintain code in better way. Codelyzer is the open source tool that check if the coding conventions and guidelines are followed or not in our application.   
  
Angular CLI supports the codelyzer. So codelyzer can be run via Angular CLI or npm directly.

## 14. What are differences between Components and Directives?

**Answer:**  
  
**Followings are the difference between components and Directives.**

|  |  |
| --- | --- |
| **Components** | **Directives** |
| A @Component requires a view. | @Directive does not require a view. |
| The components are used to split the application into smaller parts. | The directives are used to design reusable components. |
| A component, rather than adding/modifying behavior, actually creates its own view. | Directives add behavior to an existing DOM element or an existing component instance. |
| Only one component is used per DOM elements. | More than one directive can be used per DOM elements. |
| @Component meta-data annotation is used to register the components. | @Directive is used to register the directive. |

## 15. What are the security threats should we be aware of an Angular 2 application?

**Answer:**  
  
In a web application, security is the major issue. Angular 2 has the built-in protections against common web application vulnerabilities and cross-site scripting attack.  
  
**Following are the some basic guidelines to mitigate security risk.**

* Cross-site scripting (XSS)
* Consider to using AOT compilation or offline compilation.
* Try to avoid insecure direct object references.
* Try to avoid external URLs if not trusted.
* Avoid direct use of DOM API.
* Cross-site request forgery (CSRF or XSRF)
* Avoid using dynamic HTML content in our application.

**16. What is an Angular 2 services?**

**Answer:**  
  
Angular 2 Services are JavaScript functions. It is used when a common functionality needs to be provided to various modules, i.e. database functionality that could be reused among various modules. Services allow for greater seperation of concerns for your application and better modularity by allowing you to extract common functionality out of components.  
  
**Following are the steps to create the services in Angular 2.**

* Create the Service File  
  name.service.ts
* Import the injectable member  
  import { Injectable } from '@angular/core';
* Add the Injectable Decorator  
  @Injectable()
* Export the Services Class  
  export class ExampleService {  
        someMethod() {  
          return 'Welcome!';  
      }  
  }

**17. What are the features of Angular 2 Service?**

**Answer:**  
  
**Following are the features of services in Angular 2.**

* Services are singleton object i.e. only one instance of service exists throughout the application.
* Services are capable of returning the data in the form promises or observables.
* Service class is decorated with @Injectable() decorator.

**18. What are the differences between Observables and Promises?**

**Answer:**  
  
Both Observables and Promises come with abstractions that help to deal with asynchronous nature of our application.   
  
**Observables:**  
  
Observable is just a function that takes an observer and returns a  function. It is nothing more than a specific type of function with a specific purpose.  
  
It accepts an observer: an object with 'next', 'error' and 'complete' methods on it. And returns a cancellation function.  
  
Observer allows to subscribe/unsubscribe to its data stream, emit next value to the observer, notify the observer about errors and inform the observer about the stream completion  
  
Observer provides a function to handle next value, errors and end of stream (ui events, http responses, data with web sockets).  
  
Works with multiple values over time  
  
It is cancel-able/retry-able and supports operators such as map, filter, reduce etc.    
  
**Promise:**  
  
A promise represents a task that will finish in the future  
  
Promises is resolved by a value  
  
Promises get rejected by exceptions  
  
Not cancellable and it returns a single value

**19. What is CLI in Angular 2?**

**Answer:**  
  
CLI stands for command line interface. The CLI makes it easy to create an Angular JS application.   
  
Angular 2 CLI offers "generate" command. With "generate", you can create perfect components, services, modules, directives, interfaces, and pipes in seconds using only one line of text.  
  
The great thing about the CLI is that it will create some baseline test for your classes as you generate them, making it even easier to write unit tests.

**20. How can we achieve Internationalization using Angular 2?**

**Answer:**  
It can be achieved using the directive i18n.

**21. How to create and use a custom pipe?**

**Answer:**   
  
Pipes are used in templates to convert output to user friendly and readable form within interpolation braces i.e. {{ release | date }}. The '|' is denoted as pipe.  
  
A pipe takes in data as input and transforms it to a desired output.  
  
We have a stock of pipes such as DatePipe, UpperCasePipe, LowerCasePipe, CurrencyPipe, and PercentPipe in Angular.  
  
You can write your own custom pipes.  
**Steps to create custom pipe:**

* Creates a TypeScript class.
* Decorate the class with "@Pipe" decorator.
* Implements PipeTransform interface in TypeScript class.
* Override the transform() method
* Configure the class with @NgModule

**Example of custom pipe**

import { Pipe, PipeTransform } from '@angular/core';  
@Pipe({name: 'exponentialStrength'})  
  
export class ExponentialStrengthPipe implements PipeTransform {  
  transform(value: number, exponent: string): number {  
    let exp = parseFloat(exponent);  
    return Math.pow(value, isNaN(exp) ? 1 : exp);  
  }  
}

**22. Explain the data binding Angular 2.**

**Answer:**  
  
Data binding is the connection bridge between view and business logic in the application. It is the automatic synchronization between Model and the View.   
  
**Following are the types of data binding available in Angular 2.**

* **Interpolation**  
  It is the easiest way for data binding.   
  **Example:** <h3>{{vm.student.name}}</h3>
* **One Way Binding**  
  In Angular 1, ng-bind directive is used for data binding. But in Angular 2, we use HTML DOM element property for one way binding.  
  **Example:** <h1 [innerText]="student.name"></h1>
* **Two Way Binding**  
  Two-way data binding combines the input and output binding into a single notation using the ngModel directive.  
  **Example:** <input [(ngModel)]="student.name"/>
* **Event Binding**  
  Event binding can be done by using any valid HTML event like click, focus, blur etc.  
  **Example:** <button (click)="sendForm()">Send</h1>

**23. What is the event binding in Angular 2?**

**Answer:**   
  
Event binding can be done by using any valid HTML event like click, focus, blur etc. Event binding allows us to send information from the view to the component class. Such information always invoked by click, blur, typing etc. In angular 2, ng-click is replaced by (click), ng-submit is replaced by (submit).  
  
In the following example, button click event have been defined. On button click, a message is written to be displayed on screen.

**app.component.ts**

import { Component } from '@angular/core';    
@Component({    
  selector: 'test-app',    
  templateUrl: './app/databindingexample.html'    
})    
export class AppComponent {     
    msgText = '';    
    onClickMe() {    
        this.msgText = "Welcome!";    
    }    
}  
databindingexample.html  
<div>    
    <h5>Event binding Example</h5>    
     <button (click)="onClickMe()">Click me!</button>    
    <br/><br/>    
    <span> {{msgText}} </span>    
</div>

**24. What are directives in Angular 2?**

**Answer:**  
  
Directives allow to attach behavior to elements in the DOM. It is a class which contains metadata attached to the class by the @Directive decorator.  
  
Directive decorator provides additional metadata that determines how the directive should be processed, instantiated and used at runtime.  
  
**There are three types of directive available in Angular 2.**

* **Components:** Components are directive that have the template.
* **Structural directive:** Structural directives change the DOM layout by adding and removing DOM elements.
* **Attribute directive:** Attribute directives change the appearance or behavior of an element, components or other directive.

**25. Explain Meta-data in Angular 2.**

**Answer:**  
Meta-data is the way for processing the class and the component. It is used to decorate the class. There are multiple decorators used to pass the variable to a class.   
@Component decorator are used to pass the variable to the class. All the decorator are used by the meta-data to attach variable to the component or module class.

**26. What is AOT compilation in Angular 2?**

**Answer:**  
  
AOT compilation stands for "Ahead of Time compilation". The AOT compiler converts the Angular HTML and TypeScript code into efficient JavaScript code during the build time instead of run time. The AOT complier provides a more robust application with higher security because of lesser client-side HTML and JavaScript code.   
  
AOT compiler helps us to reduce the amount of synchronous request sent by the application.  
  
**Advantages of AOT Compilation:**

* It provides the Angular 2 application to download faster.
* AOT compilation provides the faster rendering of the application.
* Detect template errors earlier
* Provides better security
* Lesser Http request

**27. What is @Input in Angular2?**

**Answer:**  
  
We use the @Input decorator to define an input for a component.   
  
@Input decorator allows one way communication from parent to child. It binds a property within one component (child component) to receive a value from another component (parent component). To act as input property, the component property should be annotated with @Input decorator. A component can receive a value from another component using component property binding.

**28. What is @Output in Angular 2?**

**Answer:**  
  
@Output decorator allows one way communication from child to parent component.   
  
It binds a property of a component to send data from one component (child component) to calling component (parent component). @Output binds a property of the type of angular EventEmitter class. This property name becomes custom event name for calling component.

**29. What is PrimeNG? How can it be used with angular2?**

**Answer:**  
  
PrimeNG is a collection of rich UI components for Angular 2. PrimeNG is a part of PrimeFaces library for Angular 2 application. PrimeFaces library is pretty popular for developing UI of JSF-based Web application. PrimeNG is developed by PrimeTek Informatics, a company with years of expertise in developing open source UI components. Using PrimeNG, an Angular 2 developer can use predefined components for development like tables etc. that saves time and effort.

**30. What are the cookies in Angular 2?**

**Answer:**  
  
Cookies are small packages of information that are typically stored by our browser and websites.  
  
We cannot store more than 20 cookies per web server and no more than 4KB information in each cookie.  
  
In order to use the cookies in Angular 2, we have to install angular2-cookie library.  
  
**Following are the cookies method.**

* get()
* getObject()
* getAll()
* put()
* putObject()
* remove()
* removeAll()

What is Angular 2?  
What are the fundamentals concepts of Angular 2?  
What Are The New Features Of Angular 2? Why You Used Angular 2?  
What are advantages of Angular 2?  
How Can We Setting Up Our Development Environment For Angular 2?  
What are TypeScript Types? In Detail?  
What is AOT Compilation? – Pros and Cons of Ahead-of-Time!  
What is Lazy Loading and How to enable Lazy Loading?  
How would you Optimize the Angular 2 Application for Better Performance?  
What are the Securities Threats should we be Aware of in Angular 2 Applications?  
What are major changes in Angular 2?  
What are the core components of Angular 2?

What Is Typescript ?

What Is The Need For Typescript In Angular2?

What Is Ecmascript ?

What Is @ngmodule?

What Is Traceur Compiler ?

What Is Component In Angularjs 2 ?

What Is @inputs In Angular 2?

What Is @outputs In Angular?

What Is Primeng? How Can It Be Used With Angular2?

What Are Differences Between Components And Directives?

We Already Use Angular 1, Why Do We Need An Angular 2?

What Is An Angular 2 Component?

What Is The Languages That You Can Use To Build Angular2 Application?

How Can We Setting Up Our Development Environment For Angular 2?

What Is Npm?

How Can We Setting Up Angular 2 Application?

What Are The Security Threats Should We Be Aware Of In Angular 2 Application?

What Are The Advantages Of Using Angular 2 Over Angular 1?

How Routing Works In Angular 2.?

What Are Event Emitters And How It Works In Angular 2?

What Is The Use Of Codelyzer In Angular 2 Application.?

How Would You Optimize The Angular 2 Application For Better Performance?

How Would You Define Custom Typings To Avoid Editor Warnings?

What Is Shadow Dom? How Is It Helping Angular 2 To Perform Better?

What Is Aot Compilation?

What Are The Advantages And Disadvantages Of Aot Compilation?

What Are The Core Differences Between Observables And Promises?

## Angular 2 Interview Questions

Difference Between Constructor And Ngoninit?  
What set of modern browsers are supported in Angular 2?  
What are differences between Angular 2 and Angular 1?  
Can you write both Angular 1 and Angular 2 codes in a single project?  
What are differences between Constructors and OnInit?  
What did you like about working with Angular 2?  
When will ngInit be called?  
How would you make use of NgOnInit()?  
What is the difference between NgOnInit() and constructor() of a component?  
What is the possible order of life-cycle hooks?  
What is the Best way to Declare and Access a Global Variable in Angular 2?  
What is Angular 4? What’s New in Angular 4?  
What is Angular 5? What’s New in Angular 5?

### Angular 2 Components Interview Questions

What is Components in Angular 2?  
What are differences between Components and Directives?  
What are Components Life-Cycles?  
What is @Inputs in Angular 2?  
What is Outputs in Angular 2?  
What is hidden property in Angular 2?  
How do components communicate with each other?  
How would you create a component to display error messages throughout your application?  
What are the difference between Renderer and ElementRef in angular 2?

### Angular 2 Services Interview Questions

What is an Angular 2 Service?  
What are the features of Angular 2 Service?  
What are the differences between Observables & Promises?  
How To Create & Call Angular 2 Services in Components?  
How do we create a singleton service in Angular 2?  
How HTTP Client is Interact with Angular 2 Servers?  
What the best way to inject one service into another in angular 2?

## Advanced Angular 2 Interview Questions

**Angular 2 Pipes Interview Questions**

What is Pipes? Why use Pipes?  
What is a pure and impure pipe?  
What is Async Pipe?  
How to create and use a custom Pipes?  
Pipes Examples

**Angular 2 Directives Interview Questions**

What is Directives in Angular 2?  
What is router-outlet directive in Angular 2?  
What is a structural directive?  
How do you identify a structural directive in html?

**Angular 2 Template Interview Questions**

What is Template Angular 2? Why Use?  
What are differences between template and templateUrl?  
What are differences between templateUrl and styleUrls?  
What is a template variable. How would you use it?  
How to use {}, [], [] and [()] in Angular2 Template?  
How can you add an active class to a selected element in a list component?

**Angular 2 Styling Interview Questions**

How to use styleUrls and styles in Angular 2?  
How to import css using system import?  
How to Load external css style into Angular 2?

**Angular 2 Routing Interview Questions**

What is Routing Concepts in Angular 2?  
What is Routes?  
What is Router Imports?  
What is RouterOutlet?  
Is it possible to have a multiple router-outlet in the same template?  
What is RouterLink?

**Angular 2 State Management Interview Questions**

What are cookies in Angular 2?  
How would you pass data from a parent component to a child component?  
How would you pass data from a child component to a parent component?

**Angular 2 Dependency Injection Interview Questions**

What is Dependency Injection (DI) in Angular 2?  
What is @Injectable()? Why Use?  
What is @Inject()? Why Use?  
@Injectable() vs. @Inject()?  
How to use Dependency Injection (DI) correctly in Angular 2?  
What are the difference between @Inject and @Injectable?

**Angular 2 API Interview Questions**

Why would you use renderer methods instead of using native element methods?  
How would you control size of an element on resize of the window in a component?

**Angular 2 Ng-Modules Interview Questions**

What is the purpose of NgModule??  
How do you decide to create a new NgModule?  
What would you have in a shared module?  
What is the purpose of exports in a NgModule?  
Angular 2 Modules vs. JavaScript Modules  
What is the difference between a module’s forRoot() and forChild() methods and why do you need it?

**Angular 2 NgZones Interview Questions**

What are Zones? What is Change Detection?  
What is NgZone run outside Angular 2?  
What would be a good use for NgZone service?

**Angular 2 Traceur Compiler Interview Questions**

What is Traceur compiler? Why in Angular 2?

**Angular 2 Testing Interview Questions**

What Are Isolated Unit Tests?  
What Are Angular Testing Utilities?  
Isolated Unit Tests vs. Angular Testing Utilities!