1) What are components/directives/ templates/ modules/ services ?

**Core/Building Blocks of Angular:**

a) **Components (View):** Component is a basic unit of programming in Angular. Component represents the View part of the Angular application. Component is what users see on the screen.

Html 🡪 View

ts -🡪 Programming Logic

css 🡪 Styling Logic

b) **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.

c) **Services (Use DI):**

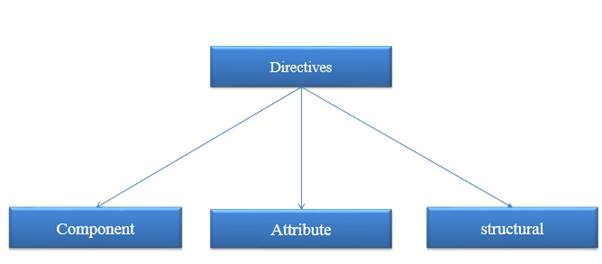
To share common logic across components. E.g. – Authentication, Data Access Code etc.

d) **Directives:**

**Reference URL:** [Angular Directives With Examples (c-sharpcorner.com)](https://www.c-sharpcorner.com/article/angular-directives-with-examples/)

Custom HTML tags or attributes which perform some kind of task.

In Angular, there are three types of directives those are component directive, attribute directive and structural directive

Classification of Directives  


**What is Component directive?**

* Component directive, is nothing but a simple class which is decorated with the @component decorator.
* In Angular, Normal typescript class will become a Component class once it has been decorated with @component decorator
* It is mainly used to specify the html templates.
* It is most commonly used directive in angular project

**Built-In Component Directive @component**

@component decorator provides additional metadata that determines how the component should be processed, instantiated and used at runtime

**What is attribute directive?**

* It is mainly used to change/modify the behavior of the html element.
* As the name tells, it is used to change the attributes of the existing html element. In Angular 2 there are many built in attribute directives. Some of the useful ones are NgClass, NgStyle

**Built-In Attribute Directive: NgStyle, NgClass**

**NgStyle**NgStyle directive is similar to one of data binding technique called style binding in angular, but the main difference is, NgStyle used to set multiple inline styles for html element.

**NgClass**It is similar to NgStyle attribute but here it is using class attribute of the html element to apply the style.

**What is Structural Directive?**

* Structural directives are responsible for HTML layout. They shape or reshape the DOM's *structure*, typically by adding, removing, or manipulating elements.
* To say in simple words, unlike Attribute Directive which we see above, Structural directive is used to add or remove the Dom Element itself in the Dom Layout, whereas attribute directives are used to just change the attribute or appearance of the Dom element.
* Structural directives are easy to recognize by using an asterisk (\*)

**Built-in structural directive - NgIf, NgFor, and NgSwitch**

* NgIf is used to create or remove a part of DOM tree depending on a condition.
* NgFor is used to customize data display. It is mainly used for display a list of items using repetitive loops
* *NgSwitch* is like the JavaScript switch It can display *one* element from among several possible elements, based on a *switch condition*. Angular puts only the *selected* element into the DOM.

e) **Modules (NgModules):**

NgModule represents a “compilation unit” for Angular.

Compilation Unit: Huge Chunk of JavaScript for all .ts files in the application.

Huge Chunk: Compilation Context

Separate Chunk for Each Feature is called as NgModule.

Logical Grouping of Components, Pipes, Directives and Services.

**App Component is part of App Module.**

**If you don’t create child modules, js for all components will be loaded as a part of root module.**

**Modules are logical. Modules cannot be seen on the screen.**

**Data is always transferred between the components.**

2) What is the Difference between ngOnInit() and constructor()

**Reference URL:** <https://javascript.plainenglish.io/difference-between-constructor-and-ngoninit-in-angular-537ecfa6ce1e#:~:text=The%20constructor()%20should%20only,as%20the%20class%20is%20instantiated>.

We use constructor() for all the initialization/declaration.

It’s better to avoid writing actual work code in the constructor.

The constructor() should only be used to initialize class members but shouldn't do actual "work".

So, we should use constructor() to setup Dependency Injection, Initialization of class fields etc.

ngOnInit() is a better place to write "actual work code" that we need to execute as soon as the class is instantiated.

Like loading data from Database — to show the user in your HTML template view. Such code should be written in ngOnInit().

3) What is the difference between string interpolation and property binding?

**Reference URL:** <https://stackoverflow.com/questions/37348563/difference-between-interpolation-and-property-binding>

Angular evaluates all expressions in double curly braces, converts the expression results to strings, and concatenates them with neighboring literal strings. Finally, it assigns this composite interpolated to an element or directive/component property. Property binding does not convert the expression result to a string.

So, if you need to bind something other than a string to your directive/component property, you must use property binding.

4) What is dependency injection in Angular?

**Reference URL:** <https://www.tektutorialshub.com/angular/angular-dependency-injection/>

**Dependency Injection (DI) is a technique in which a class receives its dependencies from external sources rather than creating them itself.**

**Benefits of Dependency Injection**

**loosely coupled**

Component is loosely coupled to the Service. It does not know how to create the Service. Actually, it does not know anything about the Service. It just works with the Service passed onto it. You can pass any Service. The Component does not care.

**Easier to Test**

Component is now easier to Test. Our Component is not dependent on a particular implementation of Service anymore. It will work with any implementation of Service that is passed on to it. You can just create a mockService Class and pass it while testing.

**Reusing the Component**

Reusing of the component is becomes easier. Our Component will now work with any Service as long as the interface is honored.

Dependency injection pattern makes our Component testable, maintainable, etc.

5) What are pipes? When do we use them, give one example.

**Use pipes to transform strings, currency amounts, dates, and other data for display.** Pipes are simple functions to use in template expressions to accept an input value and return a transformed value. Pipes are useful because you can use them throughout your application, while only declaring each pipe once. For example, you would use a pipe to show a date as April 15, 1988 rather than the raw string format.

Angular provides built-in pipes for typical data transformations, including transformations for internationalization (i18n), which use locale information to format data. The following are commonly used built-in pipes for data formatting:

• **DatePipe:** Formats a date value according to locale rules.

• **UpperCasePipe:** Transforms text to all upper case.

• **LowerCasePipe:** Transforms text to all lower case.

• **CurrencyPipe:** Transforms a number to a currency string, formatted according to locale rules.

• **DecimalPipe:** Transforms a number into a string with a decimal point, formatted according to locale rules.

• **PercentPipe:** Transforms a number to a percentage string, formatted according to locale rules.

6) What is RxJS? Name 2-3 methods

**Reference URLs (for Observables):** <https://www.geeksforgeeks.org/angular-7-observables/>

<https://angular.io/guide/comparing-observables>

<https://www.tektutorialshub.com/angular/angular-observable-tutorial-using-rxjs/>

Observables provide support for data sharing between publishers and subscribers in an angular application. It is referred to as a better technique for event handling, asynchronous programming, and handling multiple values as compared to techniques like promises.

A special feature of Observables is that it can only be accessed by a consumer who subscribes to it i.e. A function for publishing values is defined, but it is not executed till it is subscribed by the consumer (it can be any component). Consumer can receive notifications till the function runs or till they subscribed.

**RxJS:**

RxJS (Reactive Extensions for JavaScript) is a library for reactive programming using observables that makes it easier to compose asynchronous or callback-based code.

**RxJS methods:**

* [**of**](https://rxjs.dev/api/index/function/of)

The Of operator is used to convert an argument to an observable.

For more details, please refer the following URL:

[RxJS - Creation Operators - DEV Community](https://dev.to/this-is-learning/rxjs-creation-operators-6bh)

* [**filter**](https://rxjs.dev/api/operators/filter)

Filter items emitted by the source Observable by only emitting those that satisfy a specified predicate (condition).

For more details, please refer the following URL:

[RxJS - Filtering Operators - DEV Community](https://dev.to/this-is-learning/rxjs-filtering-operators-pe1)

* **Map**

The Angular observable Map operator takes an observable source as input. It applies a project function to each of the values emitted by the source observable and transforms it into a new value. It then emits the new value to the subscribers.

**Reference URL:** [https://www.tektutorialshub.com/angular/angular-observa https://www.geeksforgeeks.org/what-is-spa-single-page-application-in-angularjs/#:~:text=Single%20Page%20Applications%20are%20web,performance%20and%20loading%20pages%20faster.ble-map-operator/](https://www.tektutorialshub.com/angular/angular-observable-map-operator/)

**difference between observable and promise**(observable vs promise)

|  |  |
| --- | --- |
| **Observables** | **Promises** |
| Emit multiple values over a period of time. | Emit a single value at a time. |
| Are lazy: they’re not executed until we subscribe to them using the subscribe() method. | Are not lazy: execute immediately after creation. |
| Have subscriptions that are cancellable using the unsubscribe() method, which stops the listener from receiving further values. | Are not cancellable. |
| Provide the map, for, forEach, filter, reduce, retry, and retryWhen operators. | Don’t provide any operations. |

7) What is the use of package.json?

**Package.json is the important file for the project where you import our dependencies used in your project.** Basically, there are **two types of dependencies:**

**Dependency :** These dependencies are needed during project development and also after hosting the project. The packages listed in the dependencies section of package.json are essential to running applications. They are installed using the following command:

npm install <pakagename>

**DevDependency :** The packages listed in the devDependencies section of package.json help you develop the application on your local machine. You don't deploy them with the production application. They are installed using the following command:

npm install --save-dev <package-name>

8) What is spa in angular? how to implement it?

**Reference URL:** <https://www.geeksforgeeks.org/what-is-spa-single-page-application-in-angularjs/#:~:text=Single%20Page%20Applications%20are%20web,performance%20and%20loading%20pages%20faster>.

Single Page Applications are web applications that load a single HTML page and only a part of the page instead of the entire page gets updated with every click of the mouse. The page does not reload or transfer control to another page during the process. This ensures high performance and loading pages faster.

SPA in Angular are implemented using Routing.

9) What are decorators? Give one example

**Reference URLs:**

<https://codecraft.tv/courses/angular/es6-typescript/decorators/>

<https://medium.com/@madhavmahesh/list-of-all-decorators-available-in-angular-71bdf4ad6976>

Decorators are a new feature of TypeScript and used throughout the Angular code, but they are nothing to be scared of.

With decorators we can configure and customize our classes at design time.

They are just functions that can be used to add meta-data, properties or functions to the thing they are attached to.

A collection of useful decorators, for use in your projects or just to read and learn, can be found here:

<https://github.com/jayphelps/core-decorators.js>

**Example of Decorator:**

@NgModule:

Defines a module that contains components, directives, pipes, and providers.

**Usage:**

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

@NgModule({

declarations:[Component1, Component2],

imports: [Module1, Module2],

exports: [MyModule],

providers: [Service1, Service2],

bootstrap: [AppComponent]})

class MyModule {}

10) What are npm and node\_module folders?

**Reference URL:** <https://docs.npmjs.com/about-npm>

npm is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well.

npm consists of three distinct components:

* the website
* the Command Line Interface (CLI)
* the registry

Use the [website](https://npmjs.com/) to discover packages, set up profiles, and manage other aspects of your npm experience. For example, you can set up [organizations](https://www.npmjs.com/features) to manage access to public or private packages.

The [CLI](https://docs.npmjs.com/cli/npm) runs from a terminal, and is how most developers interact with npm.

The [registry](https://docs.npmjs.com/misc/registry) is a large public database of JavaScript software and the meta-information surrounding it.

**node\_modules folder**

This folder is generated when we run "npm install" command. This folder contains third-party libraries and files. All these files are bundled in our project together.

**Note**  
We don't need this folder while deploying our application somewhere.

11) What are annotations?

**Reference URL:**

<https://stackoverflow.com/questions/37317705/what-is-the-difference-between-annotation-and-decorator>

Traceur gives us annotations. TypeScript gives us decorators. But Angular supports both.

Annotations create an "annotations" array whereas Decorators are functions that receive the decorated object and can make any changes to it they like.

**As angular use TypeScript instead of atScript so it is using decorators.** There are basically **four kind of decorators** are there which are

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

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

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

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

For more in depth you can refer:

* <https://toddmotto.com/angular-decorators>
* <http://nicholasjohnson.com/blog/annotations-vs-decorators/>