Ritik Sharma

Services in Angular:

- Services are just a simple class with functionality which can be used among different components.
- It doesn't have html template, it contains just class file.
- To create a service in angular cli:

  ng g s service_name
- By default a service created by above command is tree-shakable means if a service is not used then it is removed from final bundle.
- Steps for a developer for service:
  1. Create a Service
  2. Provide a Service
  3. Inject a Service
  4. Use a Service
- How angular understand for service:
  1. Use a Service
  2. Inject a Service
  3. Provide a Service
  4. Create a Service
- **Step 1:** Create a Service:

  A Service is created with the above angular cli command.
- **Step 2:** Provide a Service:

  We can provide a service in two different ways:

  1. **Using provideIn:** provideIn is present in @Injectable decorator in service.
     If a service is provided with provideIn then it is tree-shakable.
     By default its value is set to 'root' which means it is singleton service means it will have only one object of the service.
     But we can set it to 'any' which it will create a single object for main and eagerly loaded modules but different objects for lazy loaded modules.
  2. **Using providers[ ]:** providers array is present in module or in a component.
     If a service is provided with providers array then it is not tree-shakable.
     We can pass directly service name as token in it in order to provide.

     providers: [Service_name]

     Which can be understand as:

     providers: [{ provide: Service_name, useClass: Service_class }]

     useClass is useful when we have a updated new Service but with same functionality signature and some new functionalities then we can useClass of updated Service but with same token.
     When we want to use multiple token for a existing object of a service, we can use useExisting option here:

     providers: [{ provide: Token, useExisting: Existing_token }]

     When we want some business logic to decide whether which service class object will be created, we have useFactory option:

     providers: [{ provide: Token, useFactory: Bussiness_Logic_function }]

One more option we have useValue when we want something globally accessed in the project.:

First create a token for that value to provide and use it:

```
const Token_name = new InjectionToken<type>("Detail about it");
```

Then provide with useValue:

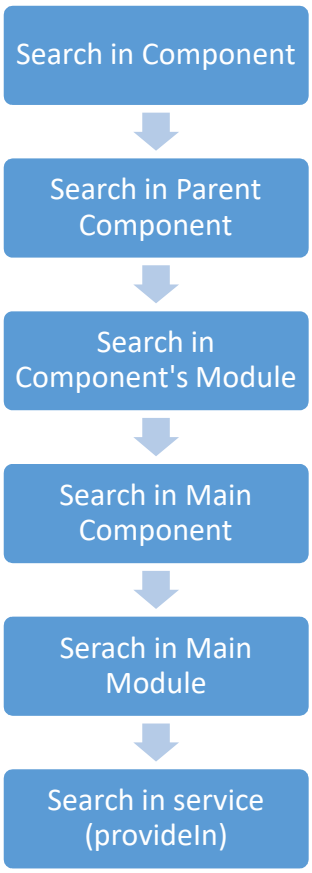
```
providers: [ { provide: Token, useValue: value_to_use} ]
```

3. **Inject a Service:** A service must be injected in a component to use it.

```
Constructor(private obj_name : Token)
```

4. **Use a Service:** Use a service with its object name to call its functions.

- How Angular will search for provide for a service in Project:



If angular will find service provided in any of these steps with not go down further it stop there.

And number of objects of service is directly equal to how many times it is provided in providers [ ] not in provideIn and it is being used in project.