Assignment 15:

(1) What is dependency Injection in Angular? and how does it help in Application development.

Ans:- Dependency Injection is a mechanism to creating and delievering some parts of an app to another that require them.

Helps :-

- promote reusability of Services
- improves maintanbility & scalability of Application
- Simplifies testing By allowing easy Mocking of Services.

(2) How do you create & register a service in Angular? Explain the role of @Injectable() & provided in?

Ans:- Create a service :-

ng generate service my-Service

Role of @Injectable() :-

- mark the class as available for Dependency Injection.
- needed for service that inject other services.

Role of provided in :-

- Registers the service in Root injector.
- it provides the service throughout Application

③ What is the difference between providing a service in Root, a module, or a component? How does it effect the service scope & instance.

Ans: providedIn: Root:

-) it Registers the service app-wide
-) Angular creates one instance across the entire app.
-) scope (Global)

Providing in a module:

- Service is Availble only to that module & its components.
- scope (Module-level)

④ Explain the concept of Hierarchial Injectors in Angular?

⇒ Services can be scoped at different levels (like root, Module, component) & Angular will search up the injector tree to resolve dependency.

How it works:

When Component requests a Service, Angular
1. checks if the component has its own injector providing the service.
2. if not found, it checks the parent Component's Injectors & its keep going until it found/ tree the service or throw error

(5) What is the purpose of @Inject() decorator in Angular DI? When would you use it instead of constructor injection alone?

Ans:- The @inject decorator tells Angular exactly what dependency to inject when it cannot determine it Automatically.

It is used when injecting:

- interfaces
- strings or primitives
- Custom InjectionToken

(6) What is the difference between @optional() and @self() in Angular dependency injection? Provide use cases.

Ans:- @optional() :- tells Angular if the dependency is not found, inject (null) instead of throwing error.

@self() :- tells Angular to look in current injector not Parent injectors, throws error if not found there.

Ex:- constructor (@optional() private logger?: LoggerService){}

Ex:- constructor (@self() private control: NgControl){}

⑦ What Are the difference between using useclass, useValue, useFactory, and useExisting in Service providers? provide examples?

useClass :- provide class to create new instance.

Ex:- { provide: myService, useclass: myService }

useValue :- to provide a fixed value (obj, string, number, etc--)

Ex:- { provide: 'API_URL' useValue: 'https://example.com' }

useFactory :- provide a value by Calling a custom function:

Ex:- { provide: myService, useFactory: ()=> new myService() { } }

useExisting :- Reuse a Existing Service under another token.

Ex:- { provide: Logger, useExisting: myService }

⑧ How you Override a service at the component level in Angular?

you can override a service provided into root injector by adding it to a component providers array:-

Implications:-

When you need custom behaviour in a specific part of the app.

Overriding at component level creates a localized service instance, allowing different behavior without affecting the rest of app.

(9) What is the role of Angular injectors class, & how can it be used to resolve dependency Manually at runtime?

Ans The injector class in Angular is responsible for resolving & providing dependencies at runtime. it allows you to get instances of services instead of using constructor injector.

Ex:-
```
constructor( private injector : injector) {}

ngonInit () {
    Const myService = this.injector.get(myService);
    myservice.doSomething();
}
```

usecases :-

-) When you need to dynamically inject a service.

-) when the dependency is optional or conditional.