1. How do you create instance of an injector for dependency injection in Angular 2?

* Injector is created by the frame work we don’t create one.

1. With DI application becomes more flexible for changes. Support this statement.

* Yes, without using DI if the class we depend on changes the structure we are supposed to change our class too. And this makes our code difficult to maintain and become inflexible and hard.

1. **@Injectable()** is a mandatory decorator used for a Service class. (YES/NO)

* No, it is not we do that if the service depends on another service. But this is always recommended way to do it.

1. What is a service provider?

* It provides instance of a service that the component needs it at run time. Basically a service is a class or object.

1. Dependency instance in Angular 2 is an application wide singleton. (YES/NO)

* No, it is singleton within the scope of the injector. Not with application.

1. Registering provider in the root module is the only way to scope a service to become application wide.

* No, if we register it in any module it is by default application logic. @NgModule this makes it application wide. @Component this makes it component wide.

1. What are dependency injection tokens? What is OpaqueToken?

* Tokens are the key to a map. So, when we provide component the class component used as token by default. And when we don’t have class token we use Opaque token.
* Token is just used to register it and to get it.

1. How can you make a dependency optional?

* We can use @optional decorator inside the constructor.

1. Apart from services, we also need dependency injection in our components and directives. But we never used **@injectable()** for our component. Why?

* Because on the component we have @Component decorator. And this is subtype of the @Injectable   
  other subtypes are @Pipe @Directive