Dependency injection

# How it works

Dependency injection decouple service initialization from client code by requiring only interface of service

It means that instead of initializing service class in client that requires service object we move creation of service to some injector and now client will only as for service object which satisfy interface

DI separate code structure in 4 pieces

1. Client
2. Service
3. Interface
4. Injector
5. **Client** is our component or code that requires functionality provided by service, Client should depend only on abstraction of service not on concrete implementation, helps in testing.
6. **Service** is concrete implementation on which client depends on it should implement abstraction (interface)
7. **Interface** is a contract between client and service, it defines properties or methods which service should implement to satisfy client requirements. Client uses interface to interact with service without knowing it’s specific implementation details
8. **Injector** is responsible for creating service instance and injecting it inside client. It insures that client will receives appropriate instance of service based on interface required by the client

# Explain as simple as possible

Dependency injection moves creation of object with some functionality (service) not to object that requires this changes (client) but to some helper (injector) and send it to client through some abstraction (interface)

# How could apply

I could apply DI when need different implementations of the same client, it means that I could change between different services if they implement the same interface

Also I could use DI for easier testing of components by mocking service by interface

# Theoretical example

Let’s suppose that I have a service (client) that requires some other logger \*service, in dependency of client I should be able to switch between logging services.

To do that I would make an \*interface that will contain method log(params: any).

This interface will implement all logging services and client will ask in it’s constructor

We already have a simple DI implementation but we are missing our \*Injector

In that case our injector could be our main file where we will initialize our Logging service and will send it as param when initializing our \*client service.