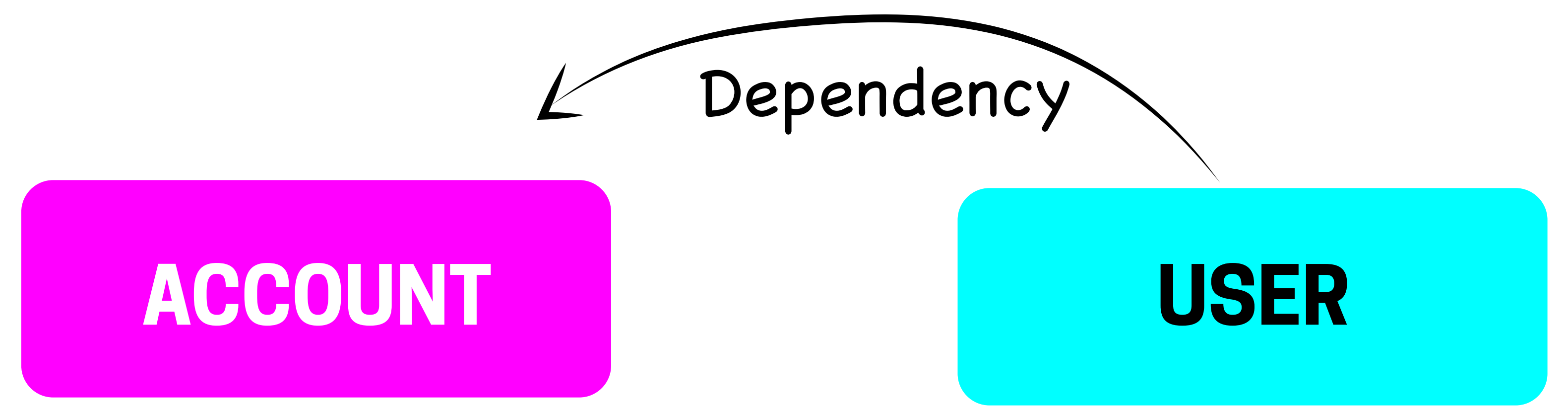
Dependency

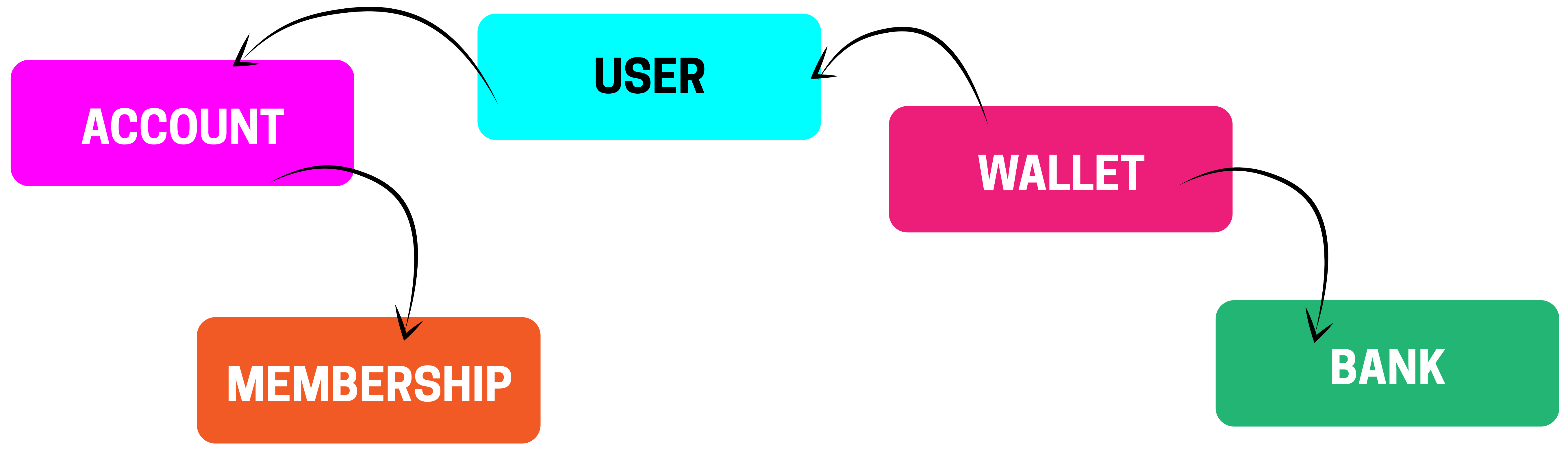


```
const account = new Account ( )

const user = new User ( account )
```

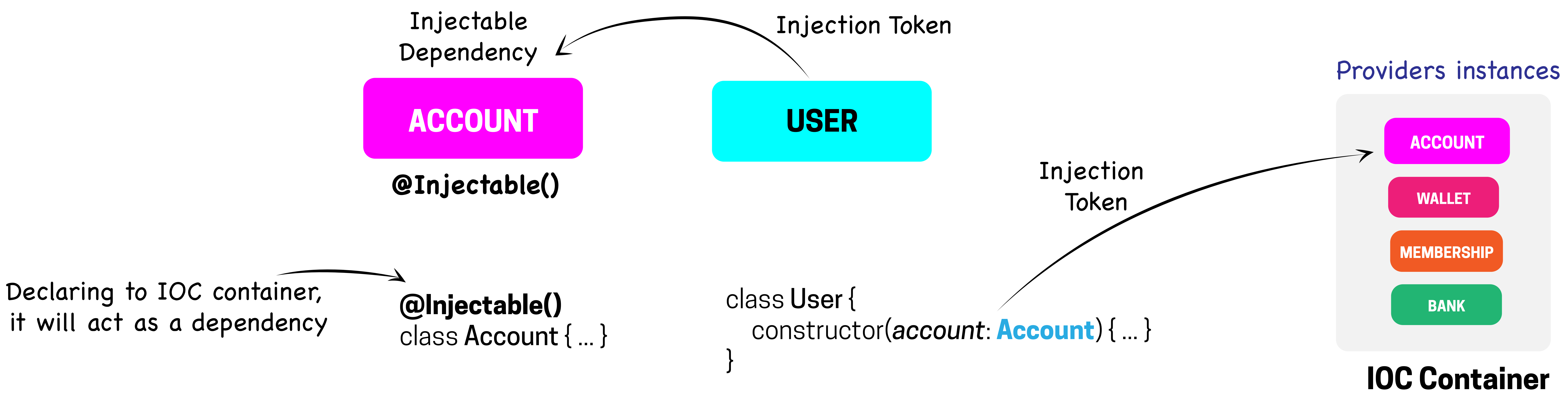
Imperative Approach

We are taking **control**
to manage **dependencies**



Dependencies Hierarchy

Dependency Injection



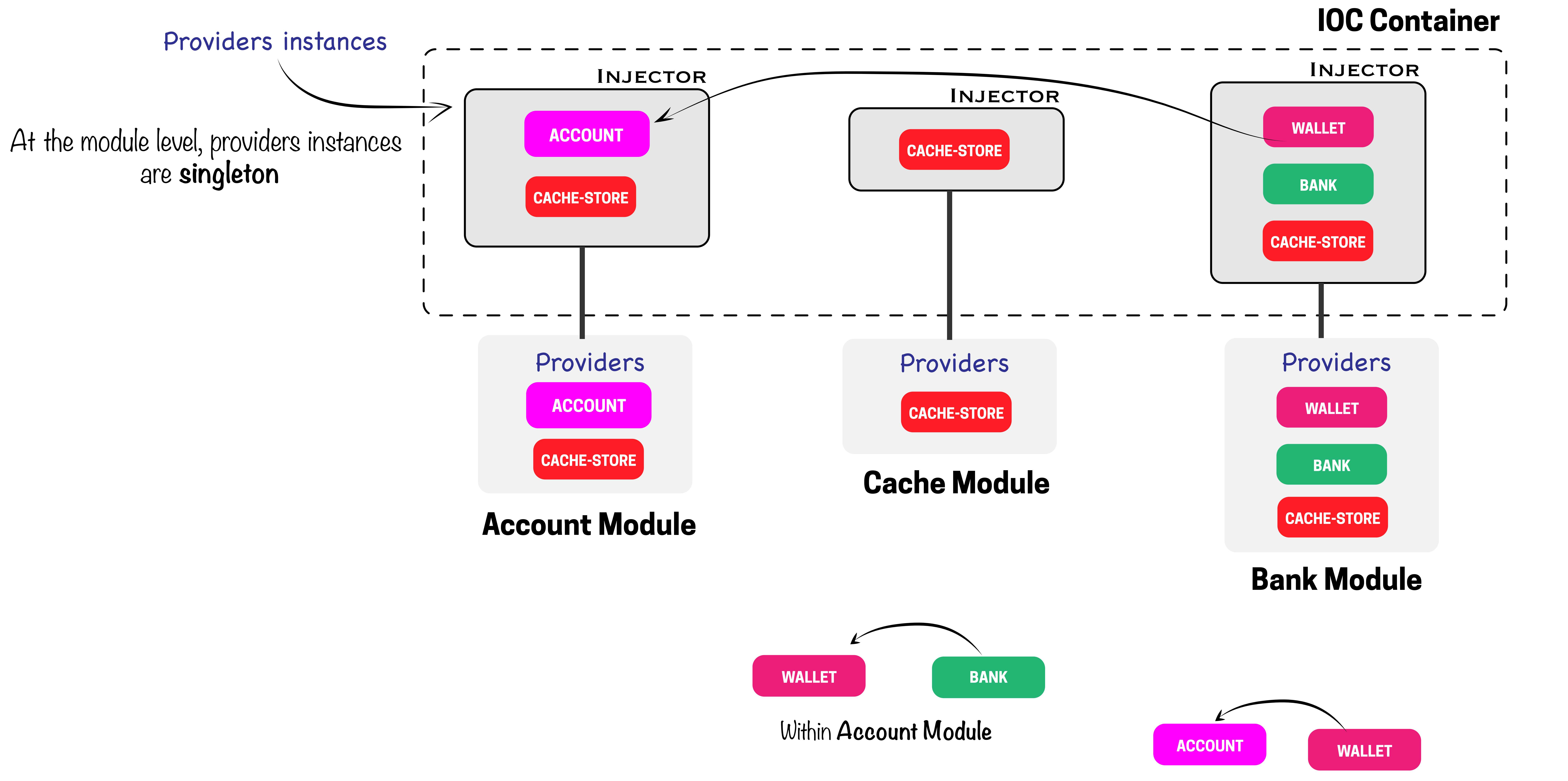
```
@Injectable()
class Account { ... }
```

```
class User {
  constructor(account: Account) { ... }
}
```

IOC Approach

We are delegating **control**
to **Nestjs runtime** for **instantiation** & **managing dependencies**

Providers Scope



We need to import the ***Account Module***,
then only ***Account instance*** is available to ***Bank Module***

Provider Types

Class Based

STANDARD

Class

Use case: Services

Non-Class Based

VALUE

Number, Boolean, String,
Object, Array,
Function

Use case: Configs, Database name, url

Factory

FACTORY

Factory Function,
Async Factory Function

Use case: Dynamic or Conditional
values, class instance

Injection Types

The **Dependency** that we need, we can ask with an **Injection token** either in the **Constructor** or in **Property** definiation.

Constructor Injection

Class provider

```
constructor( account: Account ) { ... }

constructor( @Inject(Account) account ) { ... }
```

Non-Class (value provider)

```
constructor( @Inject(ENV) env: String ) { ... }
```

Property Injection

Class provider

```
class User {

 @Inject(Account)
 account: Account
 .....

}
```

Non-Class (value provider)

```
@Inject(ENV)
env: String
```

Computer Baba

AJIT

Youtube Channel https://www.youtube.com/c/ComputerBabaOfficial

Twitter https://twitter.com/akacomputerbaba

Discord Server https://discord.gg/9V4VTDM