

FastDI is very easy to use. There are three main principles:

- 1) All objects you want to bind with each other must be installed in Binder.
- 2) The reference for the object you want to use for injection must be installed in an Awake() method or in the class Constructor (for regular classes).

For example:

```
private void Awake(){
    Binder.Install(this);
}

private void OnDestroy(){
    Binder.Remove(this);
}
```

- 3) Unity can't guarantee the initialization order for different MonoBehaviour classes. Do any reference to injected object should be in Start() method or latter in time in MonoBehaivour lifecycle.

Bad practice:

```
private void Awake(){
    Binder.Install(this);

    // At this point, initialization of _otherInjected is not guaranteed!
    // You should never refer to the injected object in Awake()!
    _otherInjected.DoSomething();
}
```

Good practice:

```
private void Awake(){
    Binder.Install(this);
}

private void Start(){
    // At this point initialization of _otherInjected is garanted and
    // works stable
    _otherInjected.DoSomething();
}
```

Contexts

For better performance or in the case of business logic, FastDI support different contexts.

`Binder.Install(object instance, BinderContext context = BinderContext.Global)`

`BinderContext` is an enum with contexts: `Global`, `Scene`. You can add your own, for example, `UI`, `GameScene`...

One object can be registered in different contexts:

```
private void Awake(){
    Binder.Install(this);
    Binder.Install(this, BinderContext.Scene);
}

private void OnDestroy(){
    Binder.Remove(this);
    Binder.Remove(this, BinderContext.Scene);
}
```

More effective contexts can be used when you have thousands of different objects that refer to one global service.

Interfaces

FastDI can be used for interface injection. Example:

`[Inject] private IServiceThree _serviceThree;`

```
namespace Example
{
    // Regular class with interface
    public class ServiceThree : IServiceThree, IDisposable
    {
        public void DoSomething()
        {
            Debug.Log("Inject regular class by interface - Works");
        }

        public ServiceThree()
        {
            Binder.Install(this);
        }

        public void Dispose()
        {
            Binder.Remove(this);
        }
    }
}
```