Access Modifiers

What Are Access Modifiers?

All types and type members in C# have an accessibility level and access modifiers are for controlling where they can be used from.

Caller's location	public	protected internal	protected	internal	private protected	private
Within the class	✔	✔	✔	✔	✔	✔
Derived class (same assembly)	✔	✔	✔	✔	✔	☐
Non-derived class (same assembly)	✔	✔	☐	✔	☐	☐
Derived class (diff. assembly)	✔	✔	✔	☐	☐	☐
Non-derived class (diff. assembly)	✔	☐	☐	☐	☐	☐

> https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers

---

What Access Modifiers Should I Use?

UdonSharp 1.x supports all access modifiers for the source C# scripts to allow for better design of larger frameworks, so we can use them all just like we would with native C#.

> Rule of thumb: Keep the access modifiers as strict as possible.

Common Mistakes

Creators who may be new to C# development standards often expose everything in their classes by leaving fields and methods on ***public***, which in case of prefab distribution can lead to confusion.

Example 1 - Fields | Public vs Private + [SerializeField]

Simply setting the access modifier of a field to ***public*** is the easiest way of making it accessible in the Unity Editor, but it also exposes it in other scripts, which may be unwanted behavior.

We can change the access modifier of the field to ***private*** and add an attribute, ***[SerializeField]***. This will result in identical behavior on the inspector, but limits the scope of the field to that class only.

```
public bool publicField;


[SerializeField]
private bool privateSerializedField;
```

Revision #1
Created 2022-08-04 16:02:35 UTC by Varneon
Updated 2023-11-07 07:36:01 UTC by Varneon