Attributes

What are attributes?

Attributes are "***markers***" or "***tags***" for associating metadata with code in a declarative way. They can be used to tell the Unity Editor, C# compiler, or even our own scripts how to treat certain ***classes***, ***fields***, ***methods***, and so on.

> https://docs.unity3d.com/Manual/Attributes.html

Attributes are declared right before their target and are always wrapped in brackets:

```
using JetBrains.Annotations;
using System.Runtime.CompilerServices;
using UdonSharp;
using UnityEngine;
using UnityEngine.Serialization;
using Varneon.VUdon.Editors;
using Varneon.VUdon.Noclip.Abstract;
using Varneon.VUdon.Noclip.Enums;
using VRC.SDKBase;
using VRC.Udon.Common;

[assembly: InternalsVisibleTo("Varneon.VUdon.Noclip.Editor")]

namespace Varneon.VUdon.Noclip
{
    [SelectionBase]
    [DefaultExecutionOrder(-1000000000)]
    [AddComponentMenu("VUdon/Noclip")]
    [DisallowMultipleComponent]
    [HelpURL("https://github.com/Varneon/VUdon-Noclip/wiki/Settings")]
    [UdonBehaviourSyncMode(BehaviourSyncMode.None)]
    public partial class Noclip : UdonSharpBehaviour
    {
        [FoldoutHeader("Options", "Options that can be edited before build and in-game")]
        [SerializeField]
        [Tooltip("Method for triggering the noclip mode")]
        private NoclipTriggerMethod noclipTriggerMethod = NoclipTriggerMethod.DoubleJump;

        [SerializeField]
        [FieldLabel("Toggle Threshold (s)")]
        [Tooltip("Time in which jump has to be double tapped in order to toggle noclip")]
        [FieldRange(0.1f, 1f)]
        private float toggleThreshold = 0.25f;

        [SerializeField]
        [FieldLabel("Speed (m/s)")]
        [Tooltip("Maximum speed in m/s")]
        [Min(1f)]
        [FormerlySerializedAs("velocity")]
        private float speed = 15f;

        [PublicAPI("Sets noclip enabled")]
        public void _SetNoclipEnabled(bool enabled)
        {
            SetNoclipEnabled(enabled);
        }

#if UNITY_EDITOR && !COMPILER_UDONSHARP
        [UsedImplicitly]
        [UnityEditor.Callbacks.PostProcessScene(-1)]
        private static void InitializeOnBuild() { }
#endif
    }
}
```

What attributes should I know about?

There are several quality of life attributes that everyone should know and use, and here are most of them:

Field Attributes

These attributes can be used on fields, primarily to alter their appearance in the inspector.

[Range]	Restrict a float or int variable to a specific range
[Min]	Restrict a float or int variable to a specific minimum value
[Header]	Add a space and a header above a field in inspector
[TextArea]	Make string field height-flexible and scrollable
[ColorUsage]	Configure Color field to support HDR and/or alpha
[GradientUsage]	Configure Gradient field's color space and HDR
[Space]	Add a space above a field in inspector
[SerializeField]	Force Unity to serialize a private field
[HideInInspector]	Hide a variable from the inspector
[Tooltip]	Display a text in inspector when hovering over a field
[NonSerialized]	Prevent variable from being serialized (also hides from inspector)
[NonReorderable]	Disable default reorderability in new array and list fields ***(Unity 2020.2+)***
[FormerlySerializedAs]	Preserve original serialized value of a field when renaming it

Class Attributes

These attributes can be used on classes.

[UdonBehaviourSyncMode]	Enforce a synchronization mode of an UdonSharpBehaviour
[DefaultExecutionOrder]	Specify the execution order of update loops in relation to other UdonSharpBehaviours
[RequireComponent]	Add a component automatically to the same object and prevent its removal
[DisallowMultipleComponent]	Prevent multiple instances of the component from being added to the same object
[AddComponent]	Specify the path to this component in the "Add Component" menu

[ExcludeFromPreset]	Prevent creation of presets from instances of the class
[SelectionBase]	Mark the GameObject as a selection base object for Scene View picking
[Icon]	Specify an icon for a MonoBehaviour or ScriptableObject ***(Unity 2021.3+)***

Method Attributes

These attributes can be used on methods.

[ContextMenu]	Add a command to the Component's context menu

Common Attributes

[Obsolete]	Mark an element to be no longer in use
[PublicAPI]	Mark publicly available API which should not be removed and treated as used
[UsedImplicitly]	Indicates that the marked symbol is used implicitly (e.g. via reflection, in external library)
[NotNull]	Indicates that the value of the marked element can never be null
[CanBeNull]	Indicates that the value of the marked element could be null sometimes

Other Attributes

These attributes don't fall into the categories above, but are extremely useful and commonly used.

[InternalsVisibleTo]

This attribute allows you to make your Runtime assembly's internal members visible to the Editor assembly.

> Example of this attribute being used: Udonity's AssemblyInfo.cs

1. Create a new C# file called `AssemblyInfo.cs` into your Runtime folder
2. Add the following content inside the file:

```
using System.Runtime.CompilerServices;
[assembly: InternalsVisibleTo("YOUR_EDITOR_ASSEMBLY_NAME_HERE")]
```

[CreateAssetMenu]

Mark a ScriptableObject-derived type to be automatically listed in the Assets/Create submenu.

Learn more about ScriptableObjects here.

```
// menuName: Path to the menu item
// fileName: Default name of the new file
// order: Priority of the menu item (100 is often reasonble)
[CreateAssetMenu(menuName = "VUdon - Vehicles/Data Presets/Car Spec Sheet", fileName =
"NewCarSpecSheet.asset", order = 100)]
public class CarSpecSheet : ScriptableObject { }
```

[InspectorName]

Use this attribute on enum value declarations to change the display name shown in the Inspector.

```
public enum ColorDisplayMode
{
    [InspectorName("RGB 0-255")]
    RGB255,

    [InspectorName("RGB 0-1.0")]
    RGB1,

    HSV
}
```

Revision #2
Created 3 November 2023 08:40:07 by Varneon
Updated 7 November 2023 08:05:15 by Varneon