

vInspector Manual

v1.1.2

Attributes

- Button attribute

- ButtonSize attribute

- ButtonSpace attribute

- Foldout attribute

- Tab attribute

- Variants attribute

- Unity's attributes

- Hidelf, ShowIf, EnableIf, DisableIf attributes

Dictionaries

- Resettable variables

- Cleaner header

- Static inspector

- Disabling features

If you have any questions or feedback - please contact us
kubacho.lab@gmail.com

Attributes

Attributes allow you to create and group UI elements in inspector without writing custom editors

Add this line to your script to use attributes:

```
using VInspector;
```

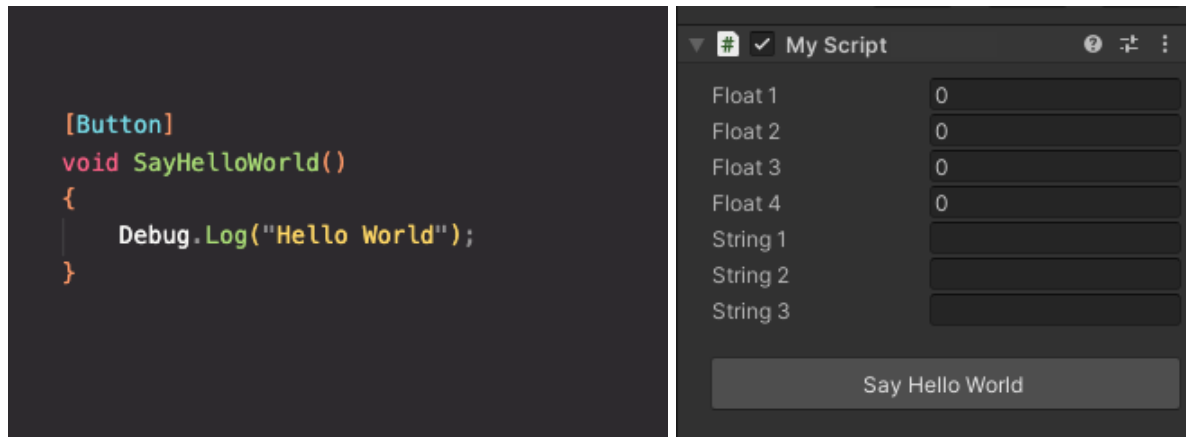
If you want attributes to retain their state after recompilation (e.g. foldouts staying folded or expanded), add this variable to your script:

```
public VInspectorData vInspectorData;
```

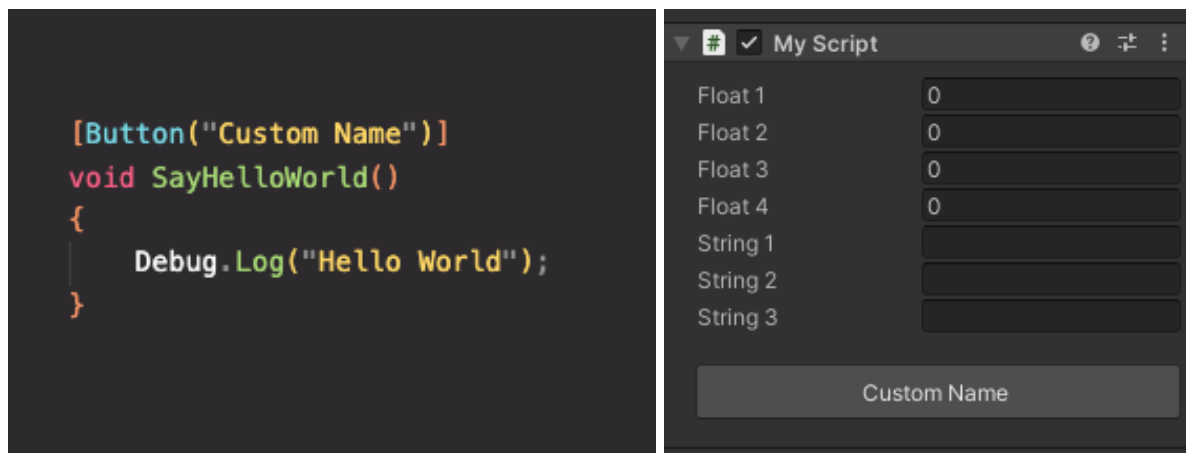
Button attribute

Creates a button at the bottom of inspector

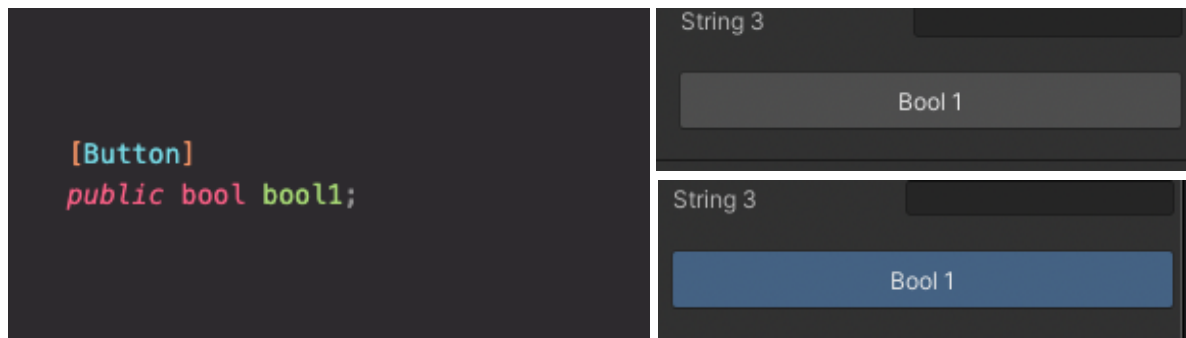
Add it before the function you want the button to invoke:



You can assign a custom name to the button:

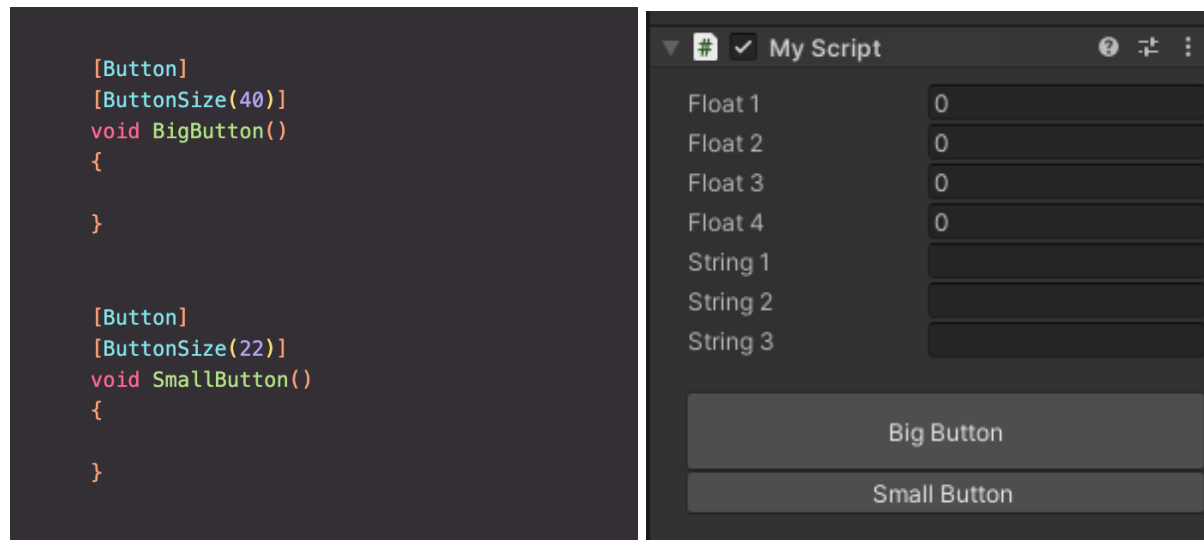


Buttons can be used to toggle bools (button appears pressed when bool is true):



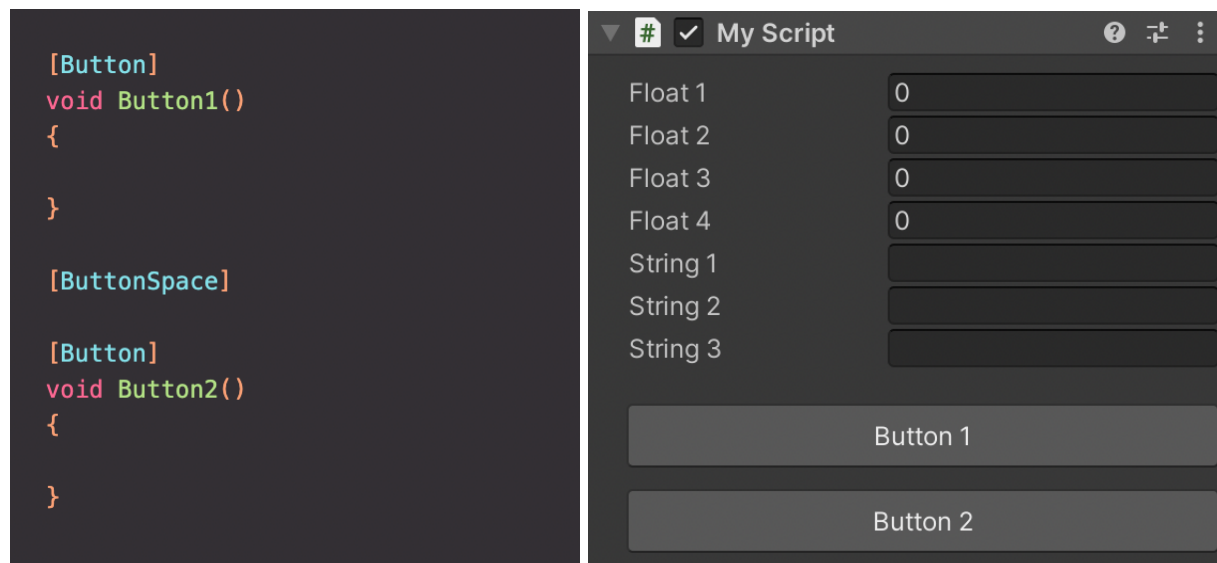
ButtonSize attribute

Use it to change button size:



ButtonSpace attribute

Use it to add space between buttons:



You can change amount of space by passing it as argument:



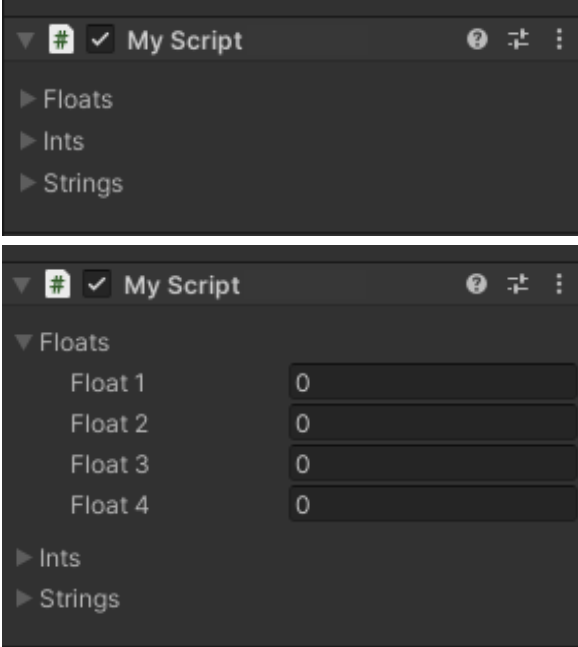
Foldout attribute

Use it to group variables into foldouts:

```
[Foldout("Floats")]
public float float1;
public float float2;
public float float3;
public float float4;

[Foldout("Ints")]
public int int1;
public int int2;
public int int3;

[Foldout("Strings")]
public string string1;
public string string2;
public string string3;
```



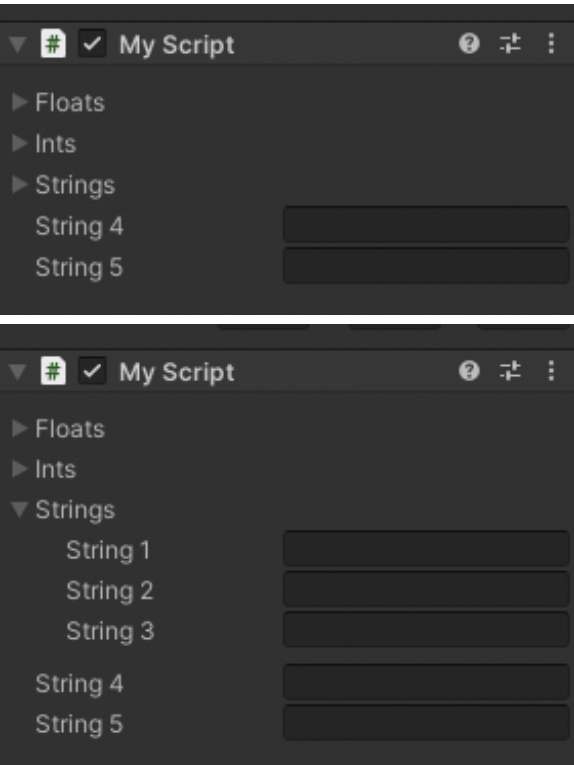
Use EndFoldout attribute to prevent grouping variables into foldout:

```
[Foldout("Floats")]
public float float1;
public float float2;
public float float3;
public float float4;

[Foldout("Ints")]
public int int1;
public int int2;
public int int3;

[Foldout("Strings")]
public string string1;
public string string2;
public string string3;
[EndFoldout]

public string string4;
public string string5;
```



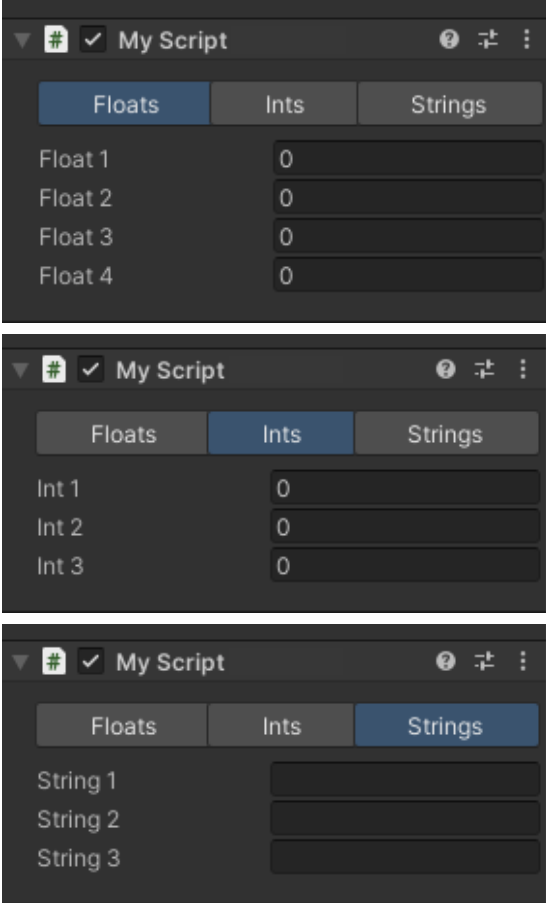
Tab attribute

Use it to group variables or buttons into tabs:

```
[Tab("Floats")]
public float float1;
public float float2;
public float float3;
public float float4;

[Tab("Ints")]
public int int1;
public int int2;
public int int3;

[Tab("Strings")]
public string string1;
public string string2;
public string string3;
```

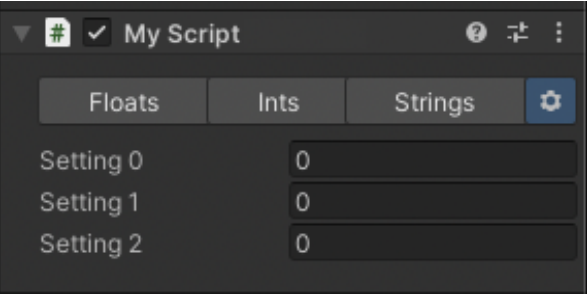


The image shows three screenshots of a script editor window titled "My Script". Each screenshot displays a different tab selected from a set of three tabs: "Floats", "Ints", and "Strings".

- Floats tab:** Shows four float variables: Float 1, Float 2, Float 3, and Float 4, each with a value of 0.
- Ints tab:** Shows three integer variables: Int 1, Int 2, and Int 3, each with a value of 0.
- Strings tab:** Shows three string variables: String 1, String 2, and String 3, each with an empty text field.

If the last tab is called Settings, it will appear as an icon instead of text:

```
[Tab("Settings")]
public int setting0;
public int setting1;
public int setting2;
```



The image shows a screenshot of a script editor window titled "My Script". The "Settings" tab is selected, which is represented by a gear icon instead of text. The tab is part of a set of three tabs: "Floats", "Ints", and "Settings". The "Settings" tab contains three integer variables: Setting 0, Setting 1, and Setting 2, each with a value of 0.

Like with foldouts, you can prevent grouping into tabs by using EndTab attribute

Variants attribute

Use it to create a dropdown for setting strings to predefined variants:



Unity's attributes

All of unity's built-in attributes are supported, feel free to mix them with `VInspector`

If you're not familiar with them - be sure to check them out, [here's a good tutorial](#)

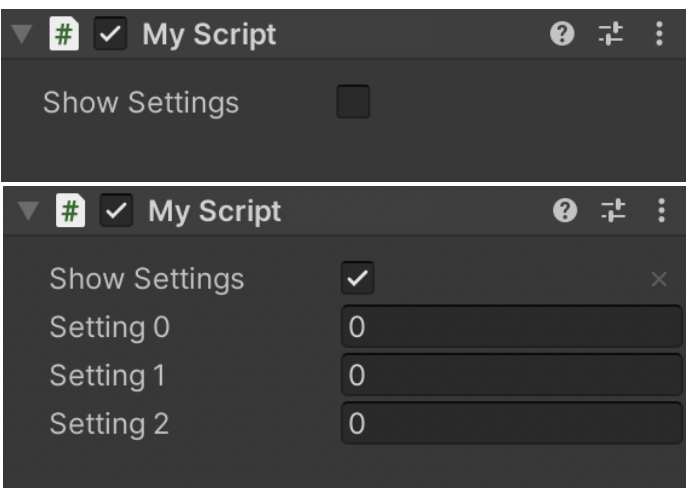
Hidelf, ShowIf, EnableIf, DisableIf attributes

These attributes allow you to hide or disable variables or buttons based on value of another variable

You can use bool as condition by passing its name as argument:

```
public bool showSettings;

[ShowIf("showSettings")]
public float setting0;
public float setting1;
public float setting2;
[EndIf]
```



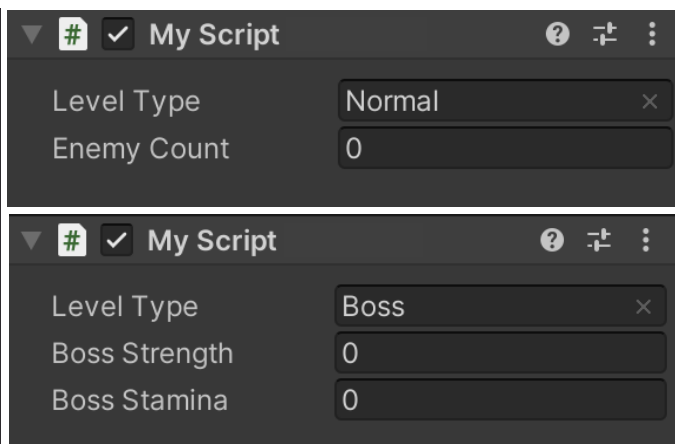
Or use any variable as condition by passing its name and desired value:

```
public string levelType;

[ShowIf("levelType", "Normal")]
public float enemyCount;

[ShowIf("levelType", "Boss")]
public float bossStrength;
public float bossStamina;

[EndIf]
```

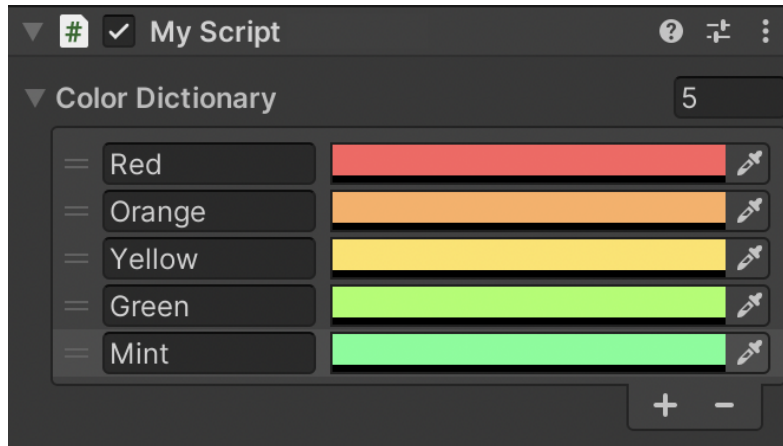


Use [EndIf] to prevent next variables getting hidden/disabled by the condition

Dictionaries

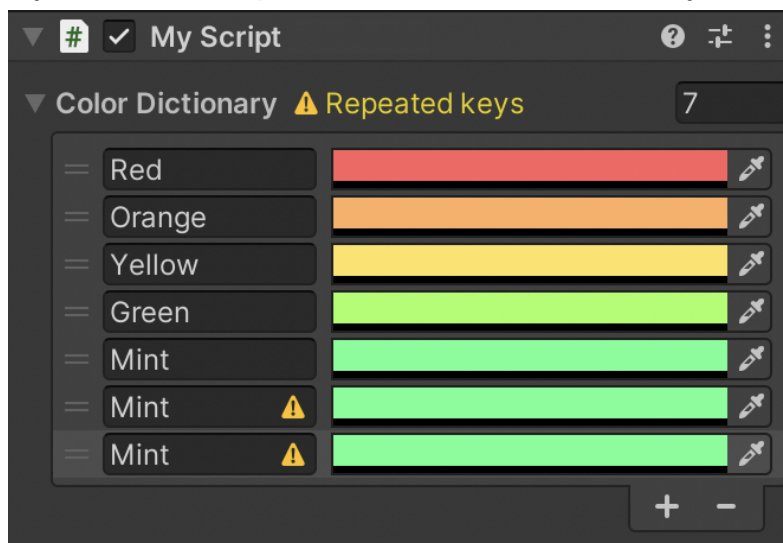
Inspector provides a serializable and editable dictionary - SerializedDictionary:

```
public SerializedDictionary<string, Color> colorDictionary;
```



It inherits from the usual dictionary class, so you can use it like any other dictionary

If you have multiple entries with the same key - warnings will be displayed:

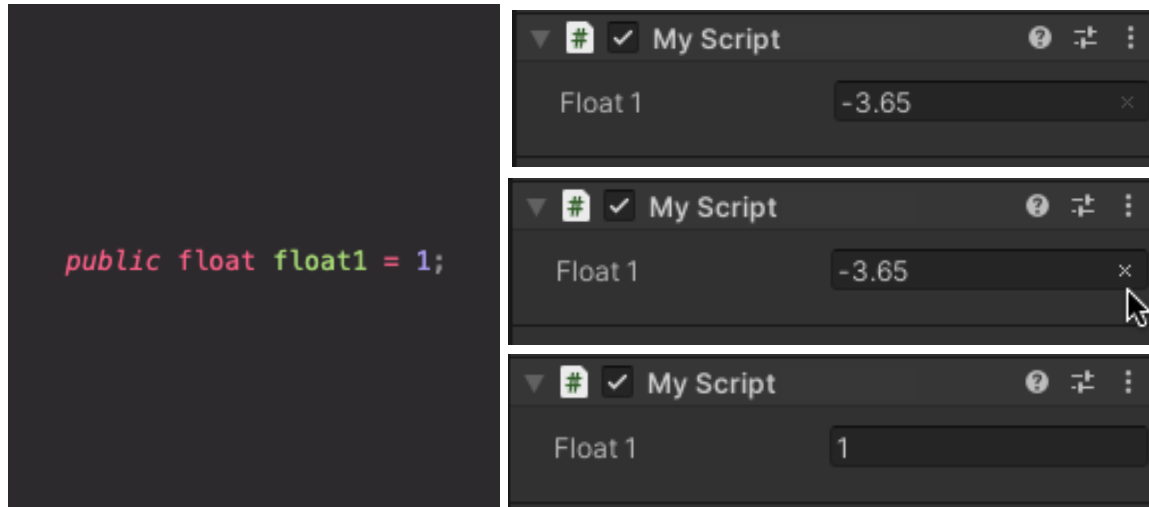


Repeated keys only appear in UI - no need to worry about them when working with the dictionary from code

Size of key and value sections can be adjusted by dragging the divider between them

Resettable variables

Variables can be reset to default value by clicking on the cross button:



If the script is attached to a prefab instance, the value on the original prefab is considered default, otherwise default is the value you defined in script

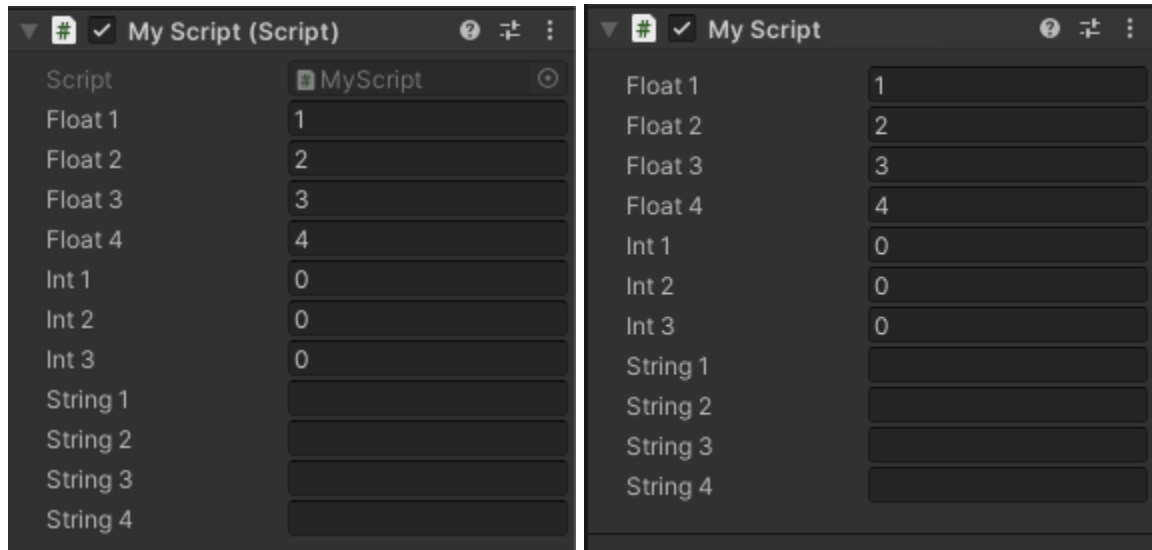
This feature works out of the box, no code needed

Except if you want it work with unity's Range attribute, use RangeResettable attribute instead:

```
[RangeResettable(0, 1)]  
public float float1 = .1f;
```

Cleaner header

vInspector hides the greyed-out Script field and “(Script)” text at the top:

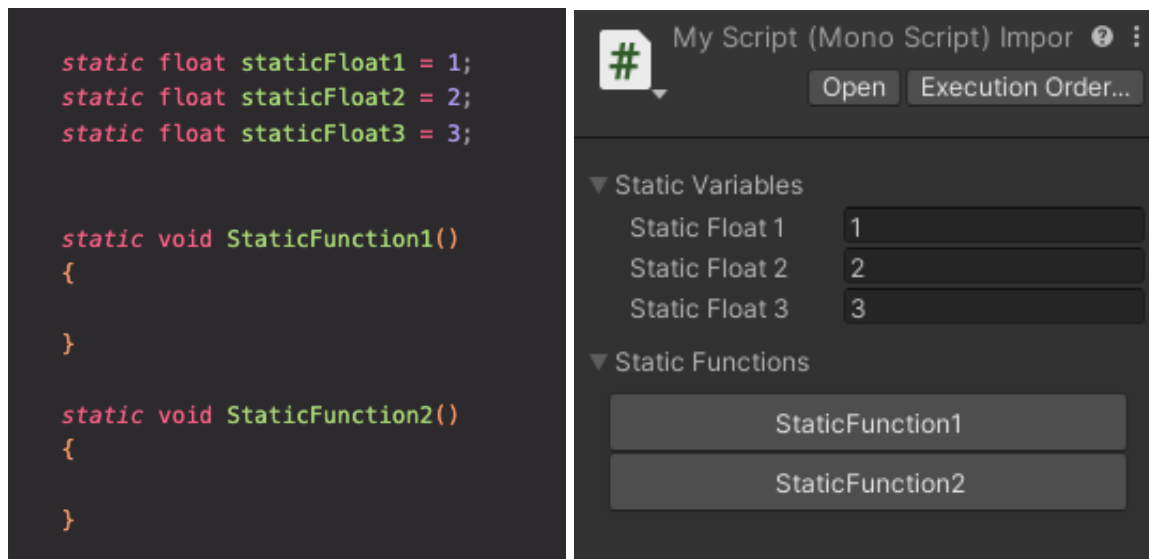


Also it allows you to open the script by double-clicking the script name or to show the script in project browser by alt-clicking the name

If you don't want script field to be hidden, you can unhide it in Tools/vInspector menu

Static inspector

Static Inspector allows you to see static variables and invoke static functions from script asset inspector:



Useful when you want to run some code without creating a GameObject and attaching a script to it

This feature works out of the box, no code needed

Disabling features

If you want to disable some features of vInspector, you can do that in Tools/vInspector menu:

