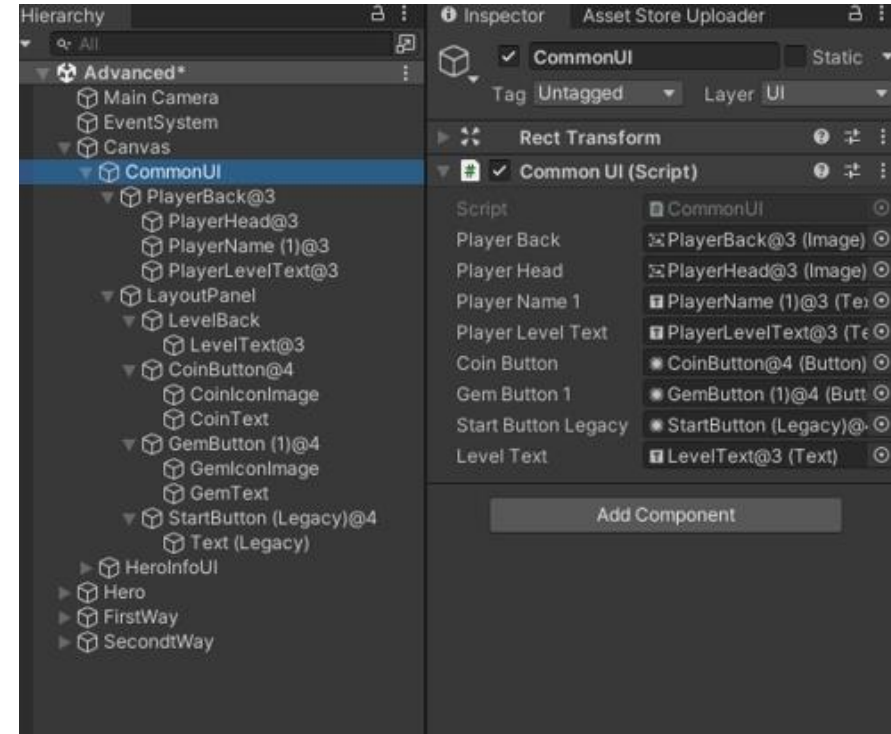
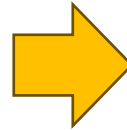
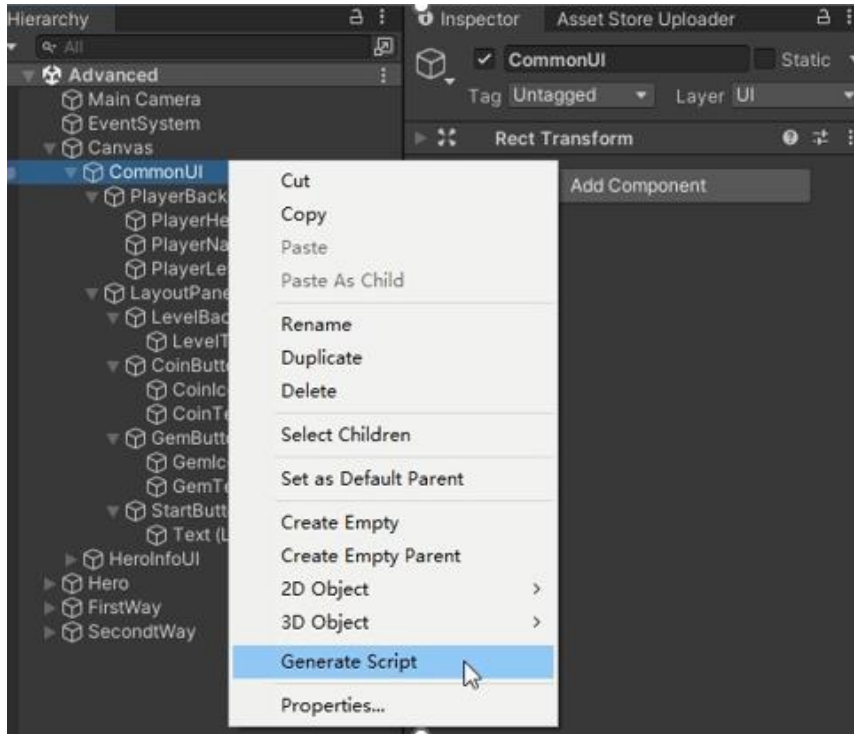


Coding Assistant

One-click(Recommended, Use@)

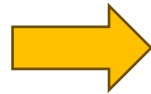
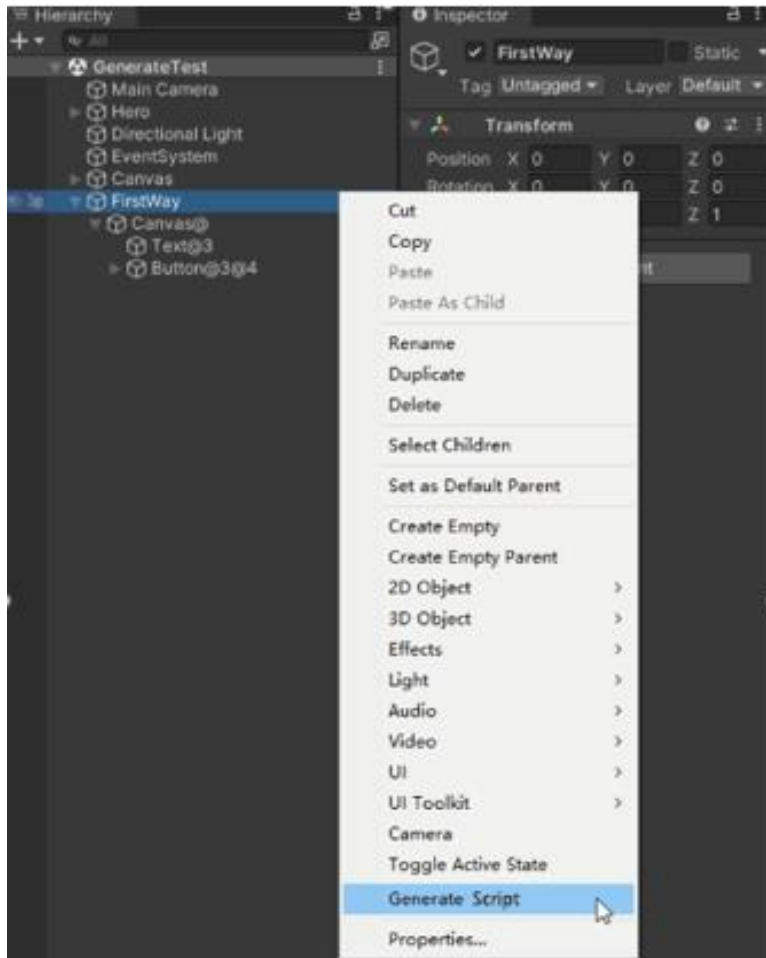


Just click the [Generate Script](#) and wait a while

- **Section 1:** Script generation

1. The **first** way (Use @)
2. The **second** way (Use GUI)
3. The **mixed** way (GUI & @)

1. The first way (Use @)



1. Right-click on your GameObject
2. Locate the **Generate Script** menu from the context menu and click:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[Unity Script 1 0 references]
public class FirstWay : MonoBehaviour
{
    [SerializeField] protected UnityEngine.GameObject canvas;
    [SerializeField] protected UnityEngine.UI.Text text;
    [SerializeField] protected UnityEngine.UI.Image button_Image;
    [SerializeField] protected UnityEngine.UI.Button button_Button;

    //Start is called before the first frame update
    [Unity Message 1 0 references]
    protected void Start()
    {
        button_Button.onClick.AddListener(OnButtonClick);
    }

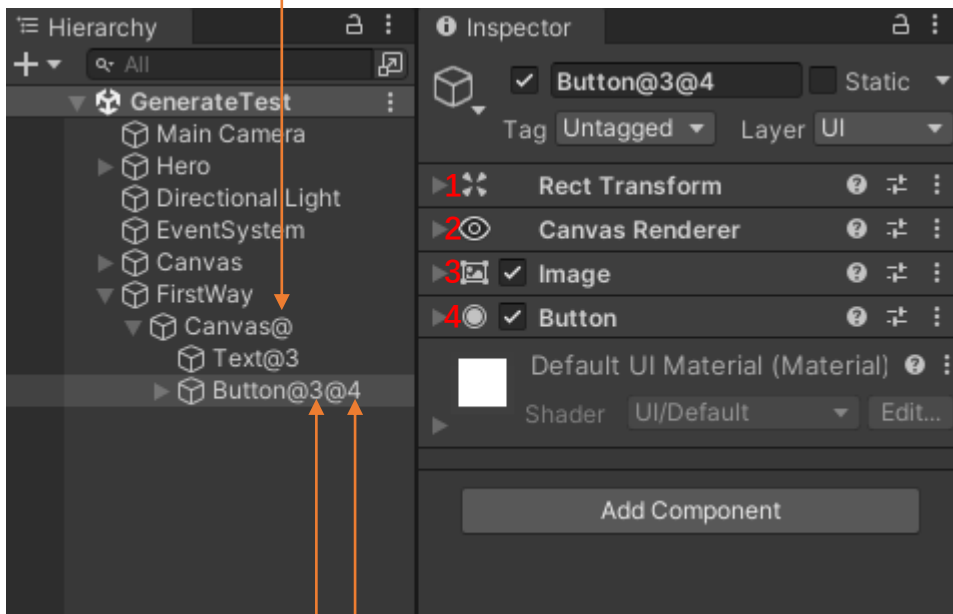
    [Unity Message 1 0 references]
    protected void OnDestroy()
    {
        button_Button.onClick.RemoveListener(OnButtonClick);
    }

    [2 references]
    protected void OnButtonClick() { }
}
```

Automatically recognize "**@(Field Maker)**" and generate scripts

The instruction of `@(Field Maker)`

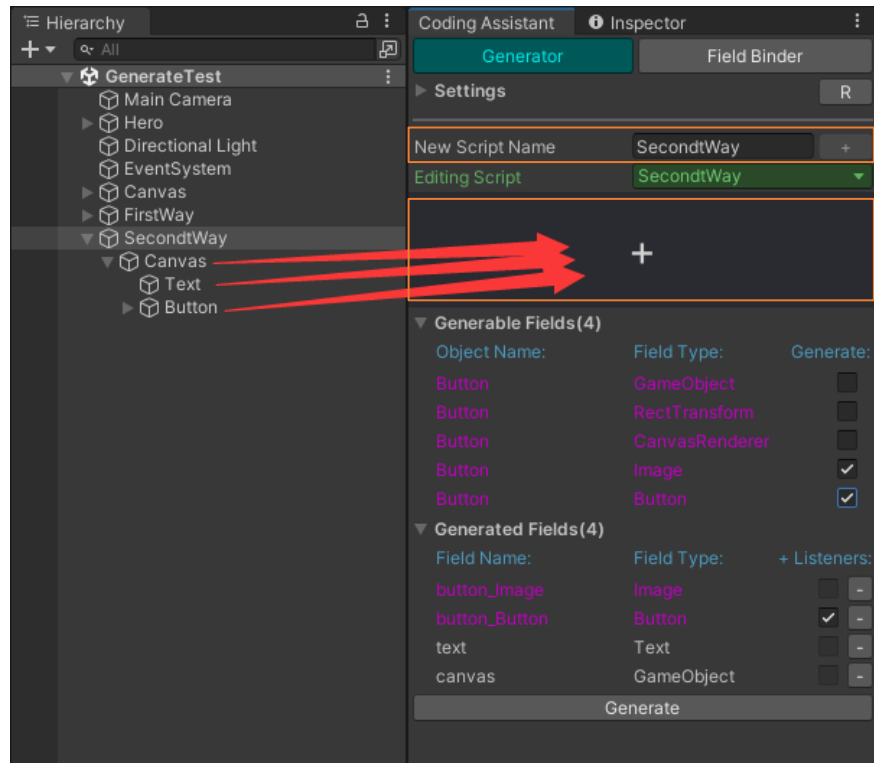
If there is no number after the "@" symbol, it defaults to representing the GameObject



The number after the "@" represents the index of the component on the object

1. The GameObject `Canvas@` in the left image will generate the following fields:
...
[SerializeField] protected GameObject canvas;
...
2. The GameObject `Button@3@4` in the left image will generate the following fields:
...
[SerializeField] protected Image button_Image;
[SerializeField] protected Button button_Button;
...
3. Therefore, if you want to generate any component you desire in the script, simply use "@" followed by the index number of the component(eg: @Index)

2. The **second** way (Use GUI), Open(Tools => We Can Do => Coding Assistant, select **Generator** tap)



1. Select GameObject **SecondtWay** and click the “+” at the end of the New Script Name line to create a new script

2. Drag GameObject into **GameObject Reading Area** and select the component you want to generate from the **Generable Fields** list

3. Finally, click the Generate button to generate the script

The instruction of Generator GUI

The image shows a screenshot of the 'Generator' GUI within a 'Coding Assistant' window. The interface is divided into several sections: 'Settings', 'New Script Name', 'Editing Script', 'Generable Fields', and 'Generated Fields'. Annotations with arrows point to specific parts of the GUI, explaining their functions.

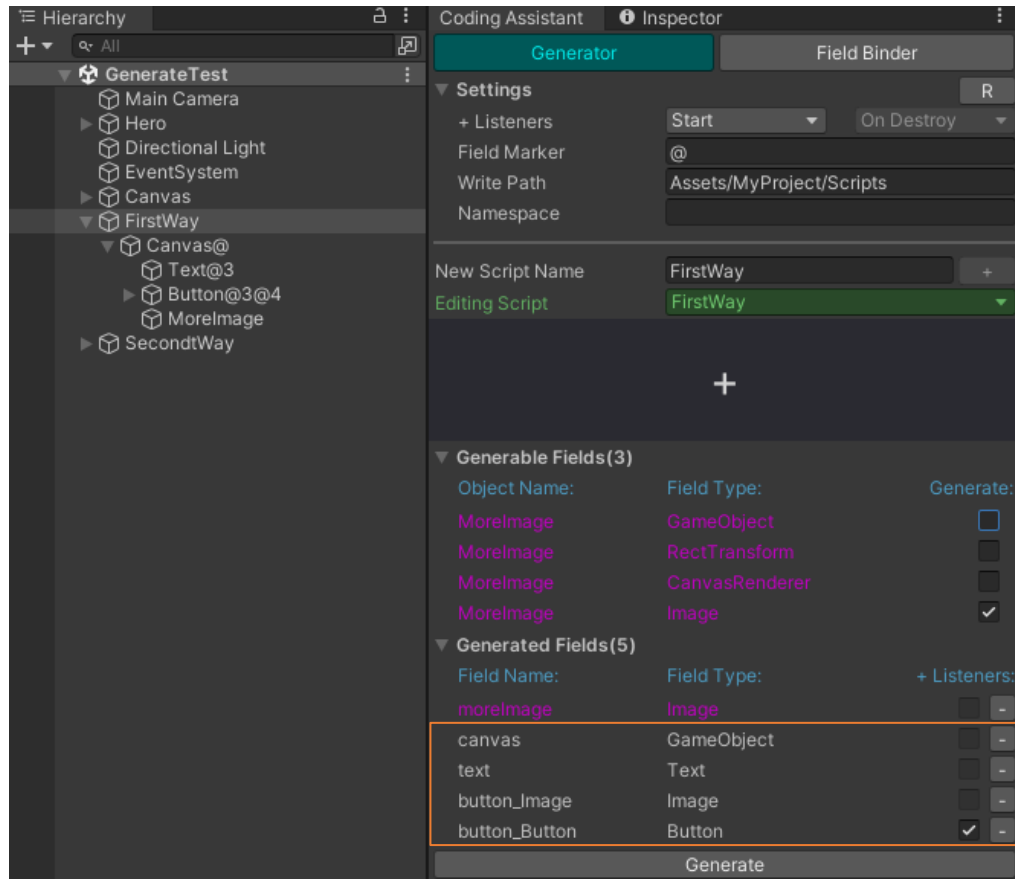
Annotations:

- Where do you bind event listener functions like Button's OnClick**: Points to the '+ Listeners' dropdown in the Settings section.
- The Field Maker as mentioned above**: Points to the 'Field Marker' dropdown in the Settings section.
- The namespace and the location of script generation**: Points to the 'Write Path' and 'Namespace' fields in the Settings section.
- GameObject Reading Area:**
Drag your GameObject here to read all the components and list them under "Generable Fields" as shown in the left image: Points to the large empty box with a '+' icon.
- Whether to generate fields for this component**: Points to the 'Generate' checkbox for a field in the 'Generable Fields' list.
- Remove the field generation for this component**: Points to the 'Generate' checkbox for a field in the 'Generable Fields' list.
- Whether to generate events for this component(just for: Button,Toggle..) and their associated listener functions, eg(default: true):**
...
button.onClick.AddListener(OnButtonClick);
...
button.onClick.RemoveListener(OnButtonClick);
...
protected void OnButtonClick() { }
... Points to the '+ Listeners' checkboxes in the 'Generated Fields' list.

GUI Details:

- Settings:** Includes '+ Listeners' (dropdown), 'Field Marker' (dropdown), 'Write Path' (text field), and 'Namespace' (text field).
- New Script Name:** Text field containing 'MainCamera'.
- Editing Script:** Dropdown menu showing 'SecondtWay'.
- Generable Fields(1):** A table with two columns: 'Object Name' and 'Field Type'. It lists 'SecondtWay' with 'GameObject' and 'Transform' field types.
- Generated Fields(5):** A table with two columns: 'Field Name' and 'Field Type'. It lists 'secondtWay' (Transform), 'canvas' (GameObject), 'button_Image' (Image), 'button_Button' (Button), and 'text' (Text).
- Buttons:** 'Generate' button at the bottom right, and a '+' button in the 'Generable Fields' section.

3. The **mixed** way (GUI & @), Open(Tools => We Can Do => Coding Assistant)



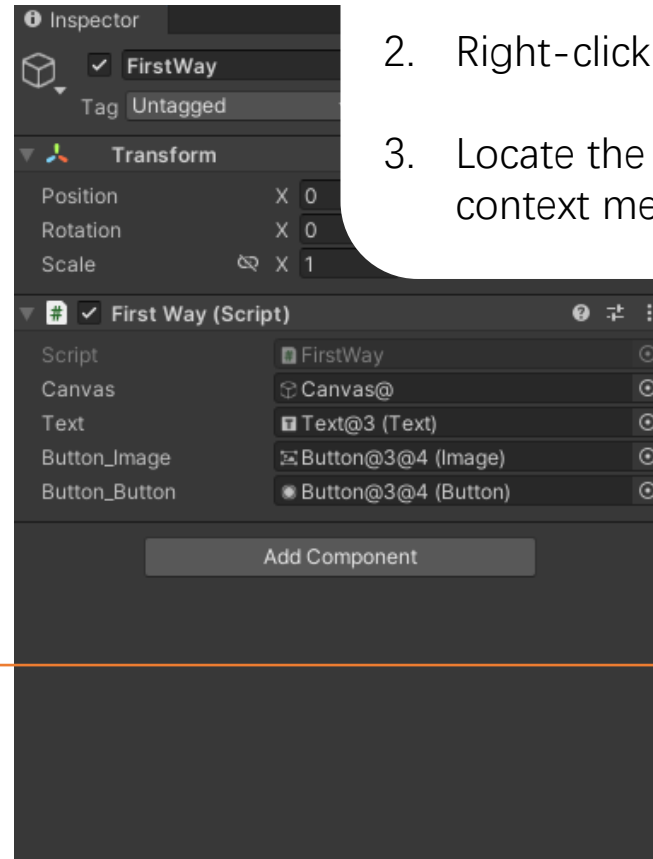
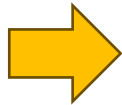
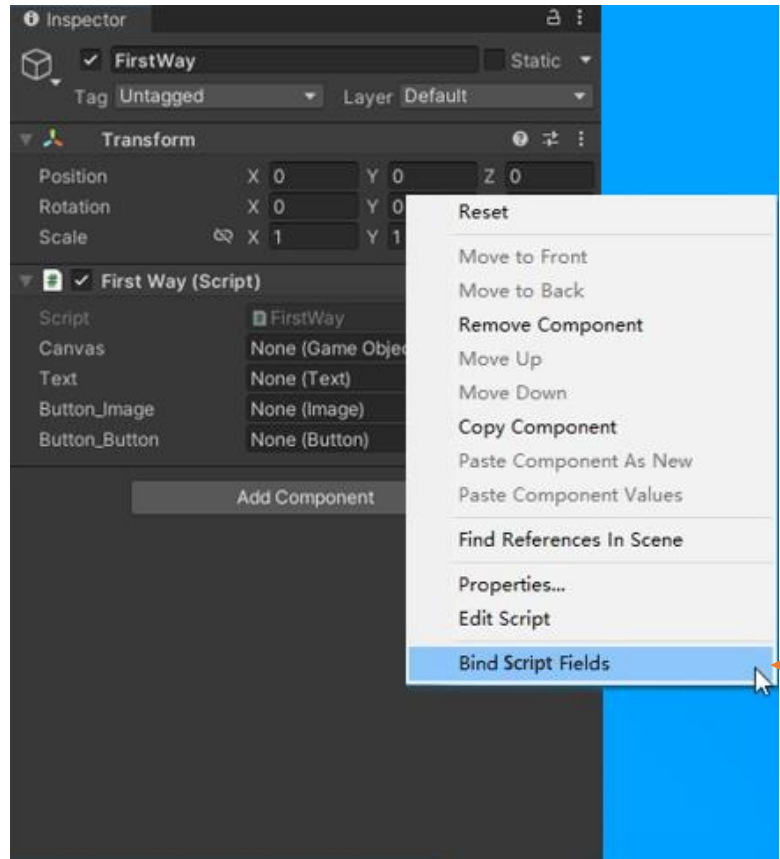
1. As the **second** way above, if you select GameObject **FirstWay** to create script, components tagged with @ will be automatically identified and listed below the **Generated Fields** section

2. Now, all you have to do is manually drag the unlisted GameObjects into **GameObject Reading Area** and select the component you want to generate, like **MoreImage** on the left

Section 2: Script field binding

1. The first way
2. The second way (Use GUI)

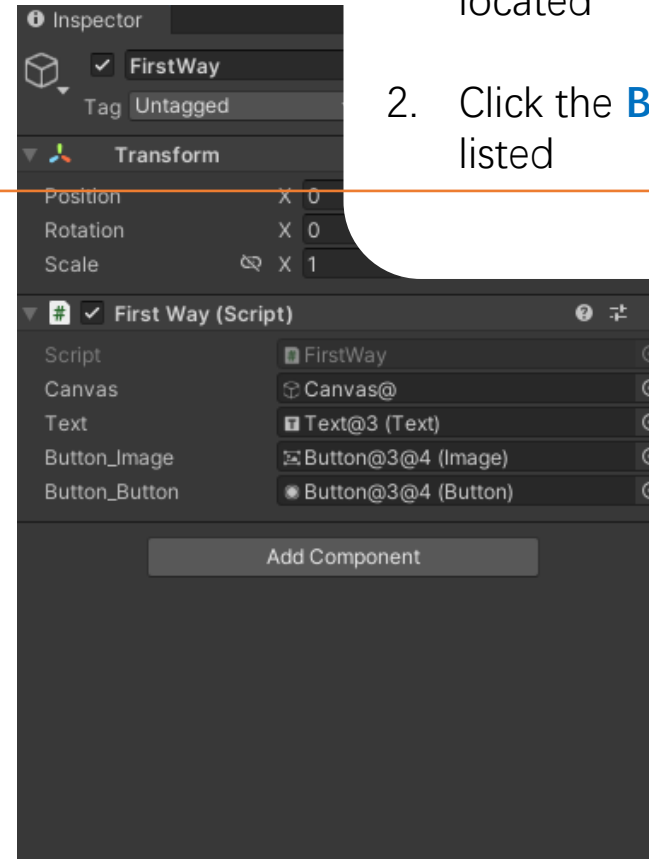
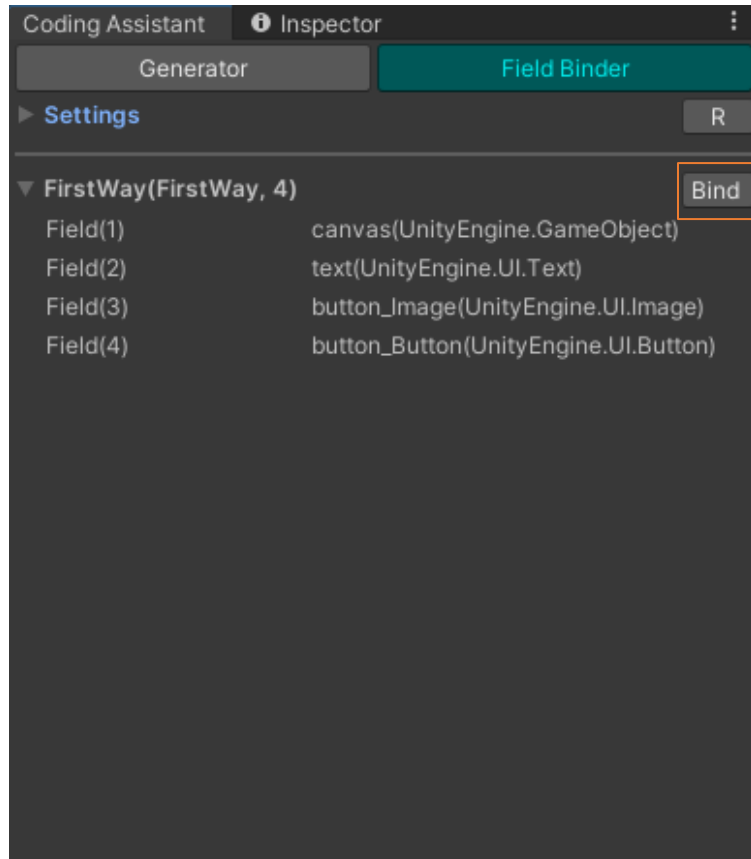
1. The **first** way



1. After the script is generated, hang it on the corresponding GameObject
2. Right-click on the script to open content menu
3. Locate the **Bind Script Fields** menu from the context menu and click

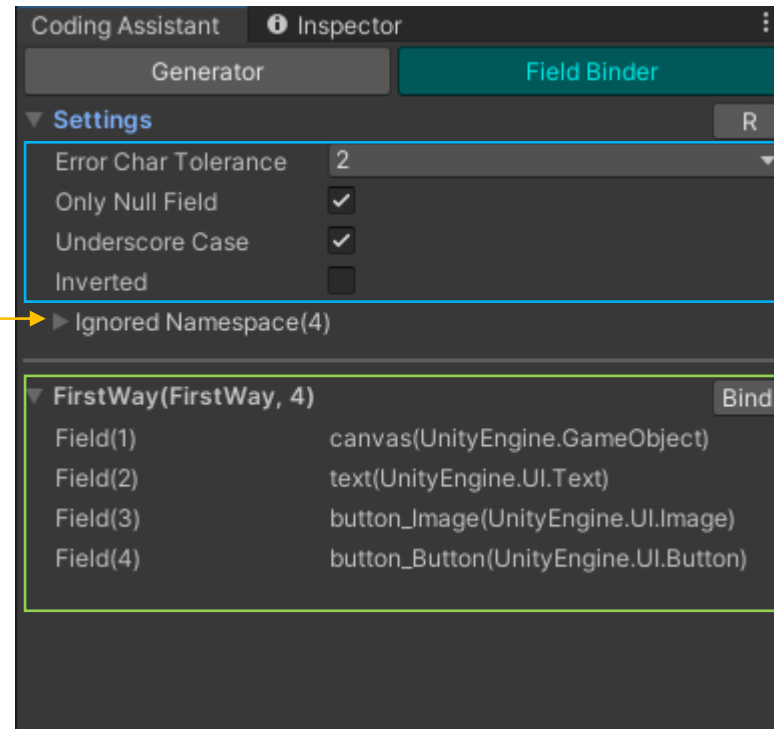
2. The **second way** (Use GUI), Open(Tools => We Can Do => Coding Assistant, select **Field Binder** tap)

1. Select the GameObject where your **script** is located
2. Click the **Bind** button next to all the scripts listed



The instruction of Field Binder GUI

Ignore the binding of the script under the namespace



Field binding Settings

Lists all scripts and fields on the GameObject that can be bound

Thank you