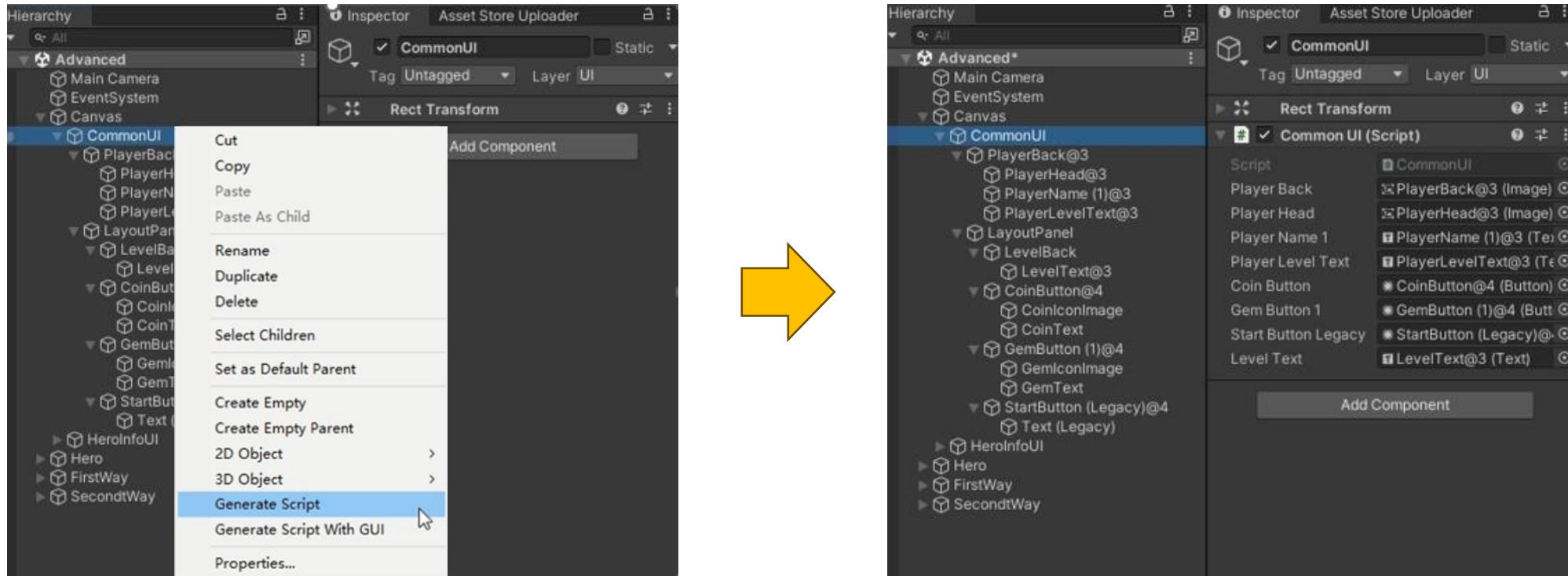


Coding Assistant

One-click(The Fastest Way, Use@)

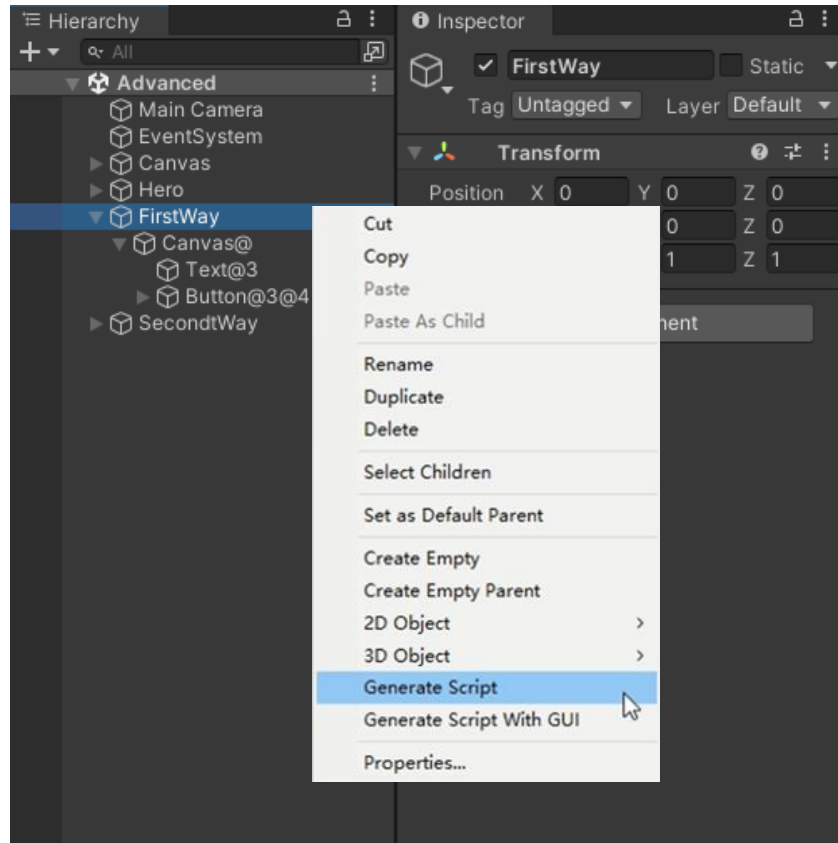


Just click the [Generate Script](#) and wait a while

- **Section 1:** Script Generation

1. The Generate Script(fastest) way (Use @).
2. The Generate Script With GUI(powerful) way (GUI & @).

1. The **Generate Script(fastest)** 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;

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
    protected void Start()
    {
        button_Button.onClick.AddListener(OnButtonClick);
    }

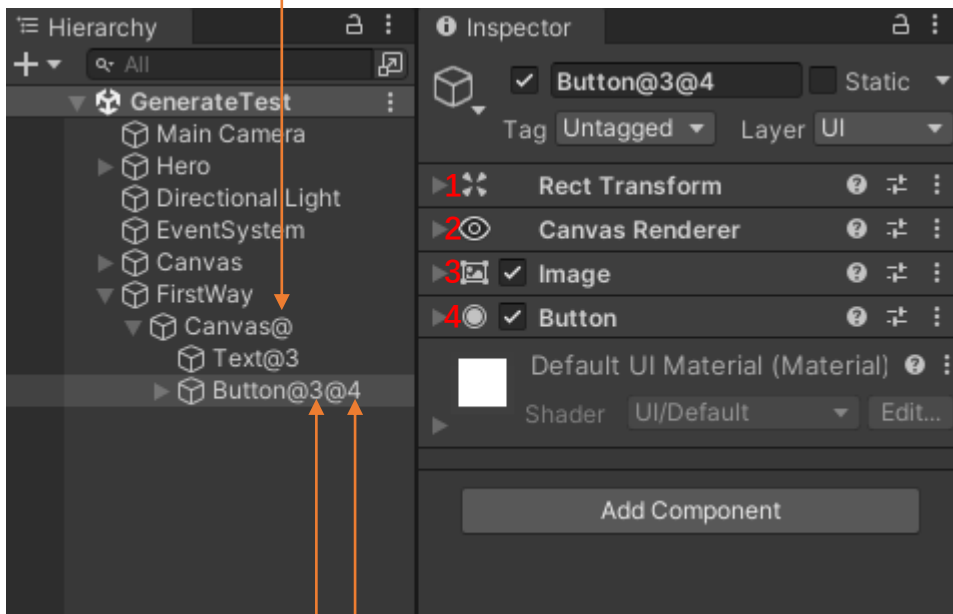
    protected void OnDestroy()
    {
        button_Button.onClick.RemoveListener(OnButtonClick);
    }

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

Automatically recognize “**@(Field Maker)**”, Generate the script and hang it on the GameObject

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;
...

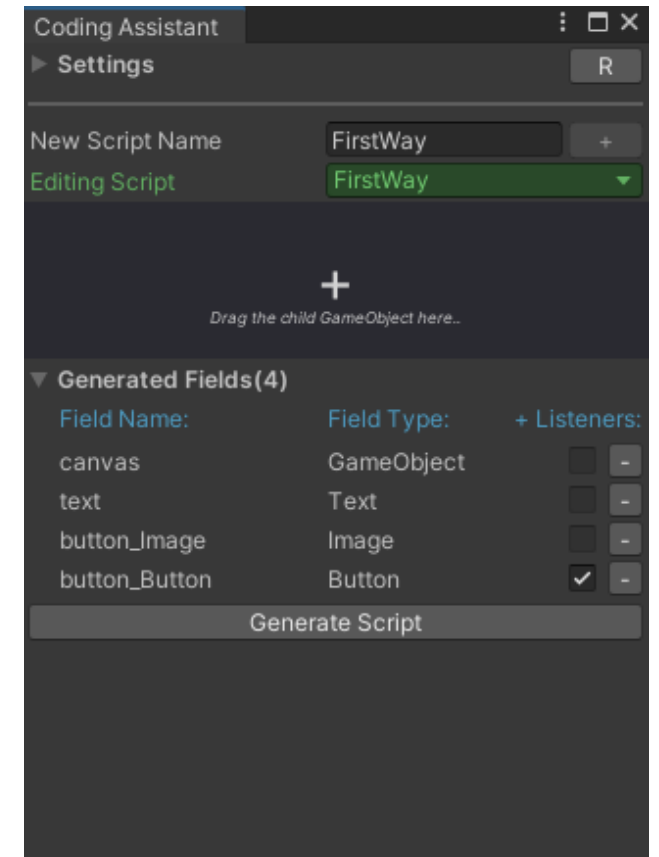
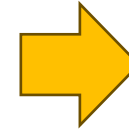
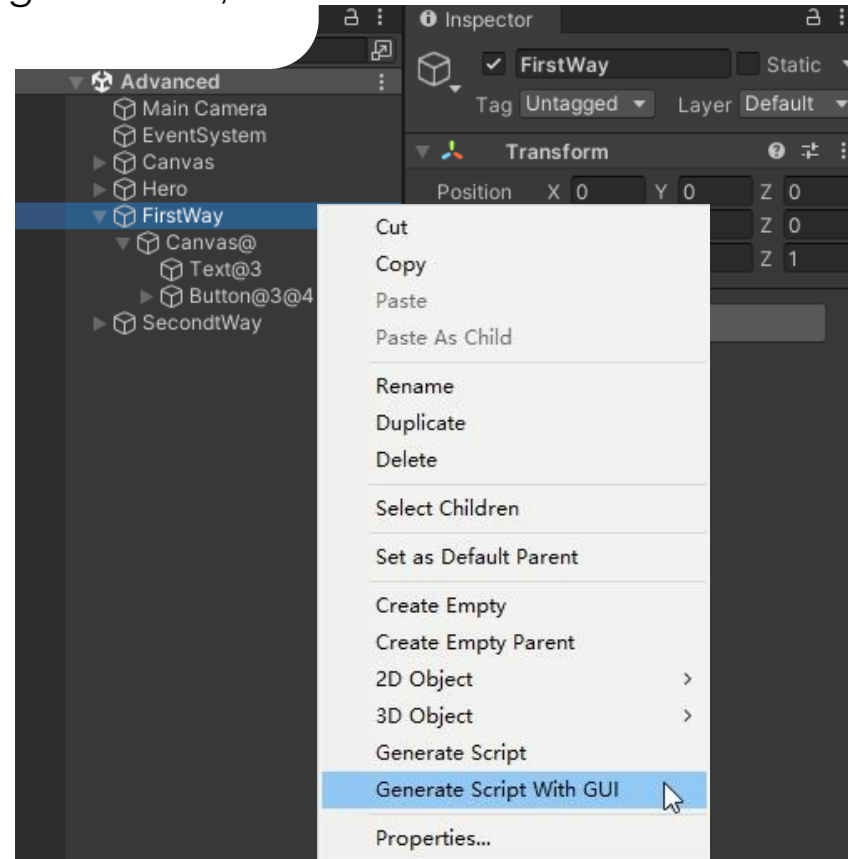
2. The GameObject `Button@3@4` in the left image will generate the following fields:

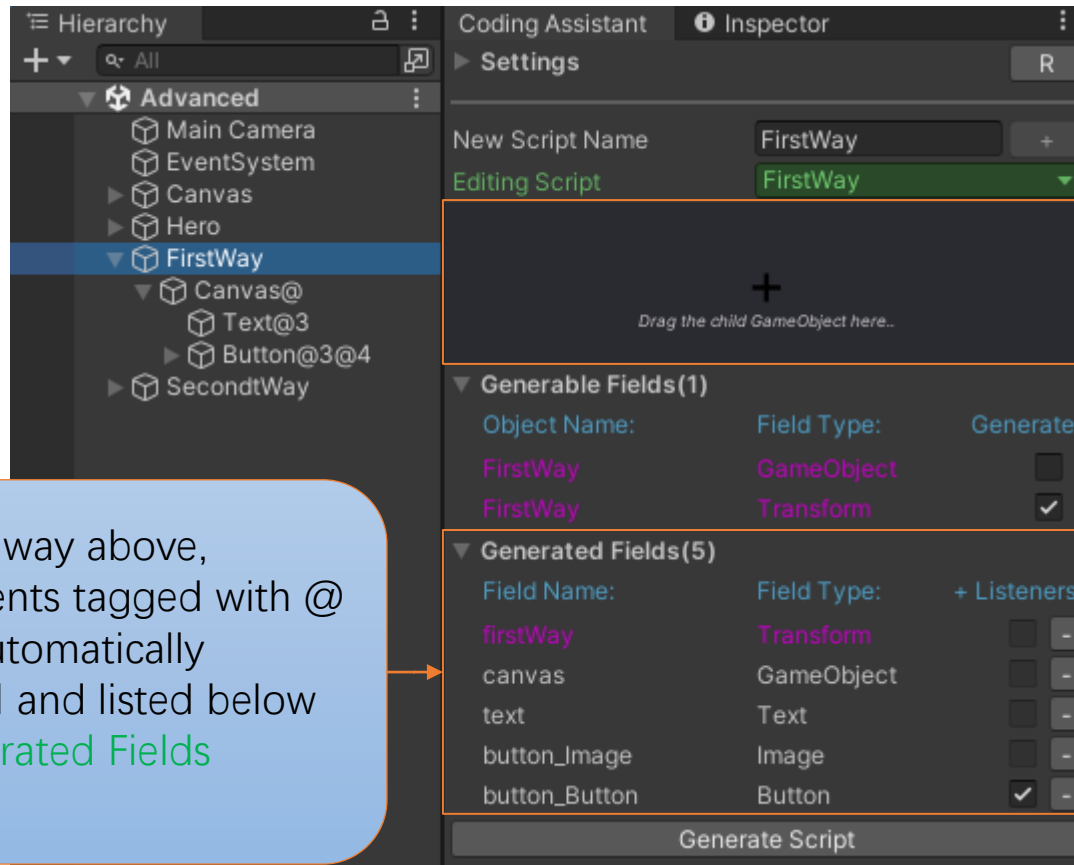
...
[SerializeField] protected Image button_Image;
[SerializeField] protected Button button_Button;
...

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 **Generate Script With GUI**(powerful) way (**GUI & @**) .

1. Same as step above, but click the **Generate Script With GUI**.
2. You can then re-edit, **add** or **remove** fields (components) to be generated, and even more.





1. As the way above, components tagged with @ will be automatically identified and listed below the **Generated Fields** section.

2. Now, drag GameObject into **Read Area** and select the component you want to generate from the **Generable Fields** list, like the transform **FirstWay** on the left.

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

The instruction of Generator GUI

The image shows the Coding Assistant GUI with several annotations explaining its features:

- Where do you bind event listener functions like Button's OnClick**: Points to the **+ Listeners** dropdown menu.
- The Field Maker as mentioned above**: Points to the **Field Marker** input field.
- The namespace and the location of script generation**: Points to the **Write Path** and **Namespace** input fields.
- 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 **Drag the child GameObject here...** area.
- Whether to generate fields for this component**: Points to the **Generate:** checkbox for the first component.
- Remove the field generation for this component**: Points to the **Generate:** checkbox for the first component.
- Whether to generate events for this component(just for: Button,Toggle..) and their associated listener functions, eg(default: true):**: Points to the **+ Listeners:** checkboxes for the components.

The GUI interface includes the following sections:

- Settings**:
 - Generator Setting** (selected) and **Field Binder Setting**.
 - + Listeners**: Start, On Destroy.
 - Field Marker**: @.
 - Write Path**: Assets/MyProject/Scripts.
 - Namespace**: (empty).
- New Script Name**: SecondtWay.
- Editing Script**: SecondtWay.
- Drag the child GameObject here...**: (empty area).
- Generable Fields(1)**:

Object Name:	Field Type:	Generate:
SecondtWay	GameObject	<input type="checkbox"/>
SecondtWay	Transform	<input checked="" type="checkbox"/>
- Generated Fields(5)**:

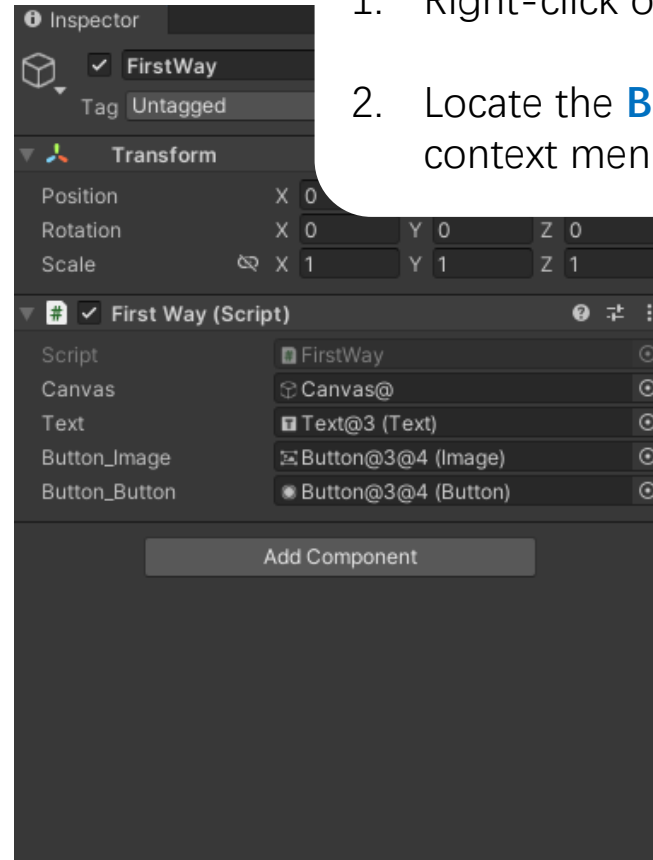
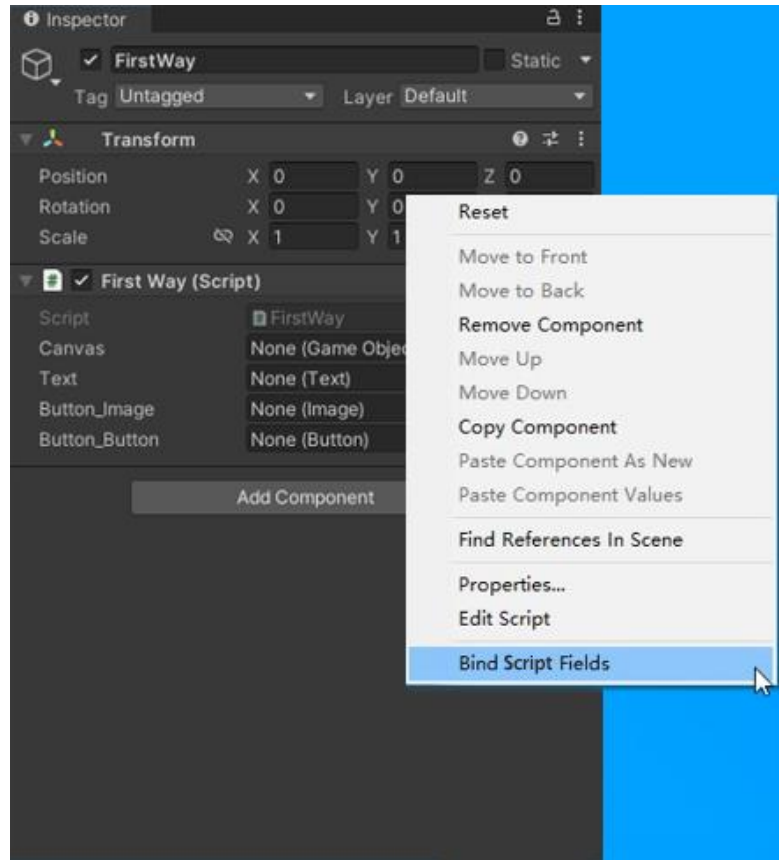
Field Name:	Field Type:	+ Listeners:
secondtWay	Transform	<input checked="" type="checkbox"/>
canvas	GameObject	<input type="checkbox"/>
text	Text	<input type="checkbox"/>
button_Button	Button	<input checked="" type="checkbox"/>
button_Image	Image	<input type="checkbox"/>
- Generate Script**: (button)

Generated Code Snippet:

```
...  
button.onClick.AddListener(OnButtonClick);  
...  
button.onClick.RemoveListener(OnButtonClick);  
...  
protected void OnButtonClick() { }  
...
```


Section 2: Script Fields Binding

1. Bind the missing reference fields (for components).



1. Right-click on the script to open content menu.
2. Locate the **Bind Script Fields** menu from the context menu and click.

The instruction of **Field Binder Setting**

The screenshot shows the 'Coding Assistant' window with the 'Field Binder Setting' tab selected. The settings include 'Error Char Tolerance' set to 2, 'Only Null Field' checked, 'Underscore Case' checked, and 'Inverted' unchecked. Below these is the 'Ignored Namespace(4)' section. At the bottom, there is a 'New Script Name' input field with a '+' button, a warning message 'Enter a script name and click the + button to get started!', and a 'Generate Script' button. The area below the button is the script editing area.

Ignore the binding of the script under the namespace

Field binding Settings

Script editing area.

Thank You