Custom Console for Unity - Documentation

This package provides a runtime console for Unity, allowing developers to execute marked methods via a command-line interface. Follow the steps below to integrate and use the console in your project.

Setup Instructions

1. Implement the ICodeEcho Interface

Add the ICodeEcho interface to any class that contains methods you want to expose to the console. This is an empty marker interface and requires no implementation.

```
using UnityEngine;


public class ExampleClass : MonoBehaviour, ICodeEcho
{
   // Methods will go here
}
```

---

2. Mark Methods with [CodeEchoMark]

Decorate the methods you want to include in the console with the [CodeEchoMark] attribute. This marks them as executable via the console.

Default Command Name:

The command name defaults to the method name.

```
[CodeEchoMark]
public void SimpleMethod()
{
   Debug.Log("SimpleMethod executed");
}
```

Custom Command Name:

You can provide a custom command name by passing it as a parameter to the attribute.

```
[CodeEchoMark("ComplexFunc")]
public void Complex_function_name_very_long()
```

```
{
    Debug.Log("ComplexFunc executed");
}
```

---

3. Add the EchoConsoleWrapper Prefab

Drag and drop the EchoConsoleWrapper prefab into any scene where you wish to have access to the debug console.

---

4. Bind Input Keys

Under the EchoConsoleWrapper prefab, locate the EchoConsole script component and bind your desired keys for the following actions:

- **Activate Console**: Opens the console (default: ```).
- **Compile Command**: Executes the entered command (default: Enter).

Usage Instructions

Activating the Console

At runtime, press the key bound to **Activate Console** (default: ```) to open the console UI.

Executing Commands

1. Type the command name of the method you want to execute.
2. As you type, matching commands will appear in a suggestion panel. Click on a suggestion to auto-populate the input field.
3. Press the key bound to **Compile Command** (default: Enter) to execute the command.

Method Constraints

- **Parameters**: Methods can accept no parameters or **one parameter** of the following types:
    - int
    - float
    - string
- Methods with unsupported parameter types will not be recognized by the console and throw an error.

Important Notes

1. **Testing Only**:
    - Ensure you remove the EchoConsoleWrapper prefab from your scenes before shipping your project to avoid exposing debug functionality to end users.
2. **Best Practices**:
    - Use descriptive command names for better clarity.
    - Avoid marking methods that may unintentionally disrupt gameplay or core functionality.

Enjoy using the Custom Console for Unity! If you have any questions or issues, feel free to reach out for support.