English FAQ
简体中文 常见问题解答

C#LikeFree

v1.1 Update time 2023-02-27

C#Like is an Unity hot update script solution. You can easy to make your project into can be hot update.
You can debug code, you can interactive prefab data, you can inherit LikeBehaviour as inherit MonoBehaviour.
You don't need to register the type in not hot update script before use it in hot update script.

PLEASE NOTE that C#LikeFree can be downloaded from anywhere, is a FREE asset.
If you do not meet the conditions of Unity's personal license, you cannot use it for commercial purposes.
C#LikeFree is the lite version of the C#Like, it missing some features from full version.
For the user who owned the Unity's personal license, can use it easy to build a hot update project using C#.
For other user, can try to find out C#Like whether worth buying by on trial the C#LikeFree.
The file tree and usage guide of the C#LikeFree is same with C#Like, so the upgrade is very convenient.

Thanks to open source C#Light

As below are the samples which export to WebGL platform (Welcome export the C#LikeFree sample yourself to verify how to hot update script):
C#LikeFree Demo C#Like Demo

Main Features

features	C#Light	C#LikeFree	C#Like
delegate	support	support	support
lambda	support	support	support
object-oriented	interface inherit	adds support partial	adds support constructor(support this/base);destructor;class inherit;virtual function
math expression	+ - * / % += -= *= /= %= > >= < <= != == && || ! ++ --(support i++ but not support ++i) ?: is as	adds full support '++ --'	adds bit operation '& | ~ ^ &= |= ^= << >> <<= >>='
keyword	this	this typeof	this base sizeof typeof unsafe $ @ #pragma #warning #error
namespace using	Not support namespace.You must register the type in not hot update script before use it in hot update script.	Support namespace,**Don't need to register the type in not hot update script** before use it in hot update script,and 'using alias/command/static'.	adds 'using sentence'
exception	throw	same with C#Light	adds try-catch-finally
type	var void bool float double char string byte sbyte int uint short ushort long ulong null	same with C#Light	Adds support Nullable type and Nullable math expression and coalescing operator '?.' and '??'.
get/set Accessor	Only support automatic implement get/set accessor	same with C#Light	Adds custom implement get set accessor
loop	for foreach continue break if-else return while do-while	same with C#Light	Adds switch-case-default
debug	Can print error sentence	In debug mode, you can use **breakpoint and step-in** to debug your code by VisualStudio. In hot update mode,you can get the stack information (include file name,class name, function name,which line) while got error.	same with C#LikeFree, but greatly reduces the difference between debug mode and hot update mode because support more C# features
compile script	It may take several seconds or even more than ten seconds to compile at runtime, depending on the amount of your code, even if it has been Precompiled into token.	All compilation processes are completed in the editor and saved as binary file. **The loading time at runtime is almost negligible.** Although the compilation time is basically the same (it even takes more time to build cached data), The loading time gives user an excellent experience.	same with C#LikeFree
MonoBehaviour	not support	You can write your hot update script just same with in normal script. You can regard LikeBehaviour as MonoBehaviour in hot update script.	**adds support the coroutine.**
multi-threading	not support	support	adds lock syntax
code annotation	//	same with C#Light	// and /**/
macro and region	not support	same with C#Light	#if #elif #else #endif #region #endregion
enum	not support	same with C#Light	support
parameter modifier	not support	same with C#Light	**support 'ref out in params'**
function overloading	not support	same with C#Light	support
default parameters	not support	same with C#Light	support
CSV	not support	KissCSV	same with C#LikeFree
JSON	not support	KissJSON	same with C#LikeFree
Socket/WebSocket	not support	KissServerFramework is easy use in hot update script, corresponding C# server. This is a most simple and stupid IOCP server framework component include WebSocket/Socket/HTTP/MySQL. All your logic work in A single main thread, you don't need to worry about multi-threading problem. All the heavy work process by framework in background threads. Easy to use database even never hear about SQL. You won't use the SQL knowledge, just need define the struct of database table, and then can use that data and it will automatically synchronize data with client and database.	same with C#LikeFree

C#LikeFree Quick Guide

normal Unity code	C#LikeFree hot update code
We define a simple class, inherit MonoBehaviour, call some Unity periodic functions which most commonly used	Here are the same with the left side normal code, but using the hot update script. You can check it out the difference.

```
using UnityEngine;
using System;

namespace CSharpLike
{
    /// <summary>
    /// Example for HelloWorld.
    /// Show the most simple usage of the Unity periodic functions.
    /// And the usage of coroutine
    /// </summary>
    public class SampleHelloWorld : MonoBehaviour
    {
        void Awake()
        {
            gameObject.name = "I'm MonoBehaviour";
            Debug.Log("Awake");
        }
        void OnEnable()
        {
            Debug.Log("OnEnable");
        }
        IEnumerator Start()
        {
            Debug.Log("Start");
            Debug.LogError("Start:step0: " + DateTime.Now);
            yield return new WaitForSeconds(2f);
            Debug.LogError("Start:step1: " + DateTime.Now);
            yield return null;
            Debug.LogError("Start:step2: " + DateTime.Now);
            yield return StartCoroutine("SubCoroutine", "test", 123, 321f);
            Debug.LogError("Start:step3: " + DateTime.Now);
        }
        IEnumerator SubCoroutine(string str, int iValue, float fValue)
        {
            Debug.LogError("SubCoroutine:(" + str + "," + iValue + ","
            + fValue + ") " + DateTime.Now);
            //In full version, supported all kind of coroutine.
            yield return new WaitForSeconds(2f);
            Debug.LogError("SubCoroutine:end " + DateTime.Now);
        }
        void FixedUpdate()
        {
            Debug.Log("FixedUpdate:same with MonoBehaviour");
        }
        float angle = 0f;
        void Update()
        {
            //Debug.Log("LateUpdate");
            angle += Time.deltaTime * 50f;
            transform.localEulerAngles = new Vector3(0f, angle, 0f);
        }
        void LateUpdate()
        {
            Debug.Log("LateUpdate");
        }
        void OnGUI()
        {
            //Debug.Log("OnGUI");
            if (GUI.Button(new Rect(0, 0, 64, 64), "Back"))
            {
                //back to SampleCSharpLike and close self
                SampleCSharpLike.instance.gameObject.SetActive(true);
                HotUpdateManager.Hide("Sample/SampleHelloWorld");
            }
        }
        void OnDisable()
        {
            Debug.Log("OnDisable");
        }
        void OnDestroy()
        {
            Debug.Log("OnDestroy");
        }
    }
}
```

```
using UnityEngine;
using System;

namespace CSharpLike
{
    /// <summary>
    /// Example for HelloWorld.
    /// Show the most simple usage of the Unity periodic functions.
    /// (support all MonoBehaviour event,but we just show the some of them.)
    /// And simulate the coroutine
    /// </summary>
    public class SampleHelloWorld : LikeBehaviour
    {
        void Awake()
        {
            gameObject.name = "I'm HotUpdateBehaviour";
            Debug.Log("Awake:same with MonoBehaviour");
        }
        void OnEnable()
        {
            Debug.Log("OnEnable:same with MonoBehaviour");
        }
        int stepStart = 0;
        void Start()
        {
            if (stepStart == 0)
            {
                Debug.Log("Start:same with MonoBehaviour but can't use 'coroutine'");
                Debug.LogError("Test Coroutine: You can't use 'coroutine' in hot" +
                " update script in FREE version. (Supported in full version). " +
                "in FREE version. (Supported in full version). Strongly recommended"+
                " update to full version C#Like: " +
                "https://assetstore.unity.com/packages/slug/222256");
                // We provide the following solutions if you not update to full version:
                // coroutine: using MemberCallDelay to simulate coroutine
                // (Only simulate 'yield return new WaitForSeconds(float seconds);'
                // and 'yield return null;').
                // The coroutine is great invention, it make our code logic clear and tidy,
                // you should not give up it.
                Debug.LogError("Start:step0: " + DateTime.Now);
                stepStart++;
                MemberCallDelay("Start", 2f);
            }
            else if (stepStart == 1)
            {
                Debug.LogError("Start:step1: " + DateTime.Now);
                stepStart++;
                MemberCallDelay("Start");
            }
            else if (stepStart == 2)
            {
                Debug.LogError("Start:step2: " + DateTime.Now);
                stepStart++;
                SubCoroutine("test", 123, 321f);
            }
            else if (stepStart == 3)
            {
                Debug.LogError("Start:step3: " + DateTime.Now);
                stepStart = 0;
            }
        }
        int stepSubCoroutine = 0;
        void SubCoroutine(string str, int iValue, float fValue)
        {
            if (stepSubCoroutine == 0)
            {
                Debug.LogError("SubCoroutine:(" + str + "," + iValue + ","
                + fValue + ") " + DateTime.Now);
                stepSubCoroutine++;
                MemberCallDelay("SubCoroutine", str, iValue, fValue, 2f);
            }
            else if (stepSubCoroutine == 1)
            {
                Debug.LogError("SubCoroutine:end " + DateTime.Now);
                MemberCallDelay("Start");
                stepSubCoroutine = 0;
            }
        }
        void FixedUpdate()
        {
            Debug.Log("FixedUpdate:same with MonoBehaviour");
        }
        float angle = 0f;
        void Update()
        {
            //Debug.Log("LateUpdate:same with MonoBehaviour");
            angle += Time.deltaTime * 50f;
            transform.localEulerAngles = new Vector3(0f, angle, 0f);
        }
        void LateUpdate()
        {
            Debug.Log("LateUpdate:same with MonoBehaviour");
        }
        void OnGUI()
        {
            //Debug.Log("OnGUI:same with MonoBehaviour");
            if (GUI.Button(new Rect(0, 0, 64, 64), "Back"))
            {
                //back to SampleCSharpLike and close self
                SampleCSharpLike.instance.gameObject.SetActive(true);
                HotUpdateManager.Hide("Sample/SampleHelloWorld");
            }
        }
        void OnDisable()
        {
            Debug.Log("OnDisable:same with MonoBehaviour");
        }
        void OnDestroy()
        {
            Debug.Log("OnDestroy:same with MonoBehaviour");
        }
    }
}
```

C#LikeFree how to interactive data with Unity prefab (completely same with the full version)

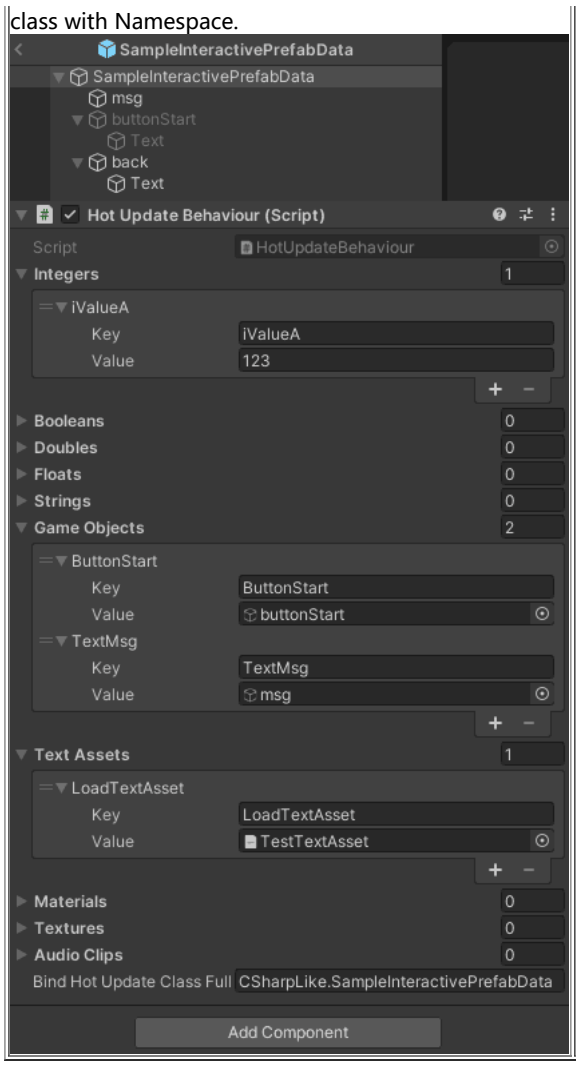
setting prefab in unity editor	interactiv in hot update script
As shown below, show how to get/set value and component which binding in prefab. BindHotUpdateClassFullName is the full name of the	As shown below, it's the binding hot update script.

```
using System;
using UnityEngine;
using UnityEngine.UI;

namespace CSharpLike
{
    /// <summary>
    /// Sample for interactive prefab data
    /// </summary>
    public class SampleInteractivePrefabData : LikeBehaviour
    {
        void Start()
        {
            //the instance of the HotUpdateBehaviour which inherit the MonoBehaviour.
            Debug.Log("HotUpdateBehaviour=" + behaviour);
            //same with the gameObject of MonoBehaviour
            Debug.Log("gameObject.name=" + gameObject.name);
            //same with the transform of MonoBehaviour
            Debug.Log("transform.localPosition=" + transform.localPosition);

            //We can get/set value(int/bool/float/double/string) which binding in prefab(key by string).
            //Here are the example for the integer value.
            //Get the integer value which binding in prefab.
            //For example,we blinding the key 'iValueA' with value 123 in prefab.
            //GetBoolean/GetDouble/GetFloat/GetString
            Debug.Log("integer value set in prefab:iValue=" + GetInt("iValueA"));
            //Modify the integer value, It will refresh the value in editor.
```

class with Namespace.




```
28.             //You also can modify it in editor when running.
29.             SetInt("iValueA", 321);//SetBoolean/SetDouble/SetFloat/SetString
30.             //Verify the integer value after modify
31.             Debug.Log("integer value after modify:iValue=" + GetInt("iValueA"));
32.
33.             //We can accept the Unity Component(GameObject/TextAsset/Material/Texture/AudioClip)
34.             //which binding in prefab(key by string)
35.             //Here are the example for get the 'GameObject' which bind at the gameObjects in HotUpdateBehaviour
36.             GameObject goStart = GetGameObject("ButtonStart");
37.             goStart.SetActive(true);
38.
39.             //Here are the example for get the UnityEngine.UI.Text in prefab.
40.             //You can get 'runtime' component which bind at the gameObjects in HotUpdateBehaviour.
41.             //You also can get any type which can get by GameObject.GetComponent<T>() in MonoBehaviour.
42.             //such as Image/Button/Slider/MeshRenderer/...
43.             Text text = GetComponent<Text>("TextMsg");
44.
45.             //Here are the example for get TextAsset 'resource' which bind in prefab.
46.             //Such as we bind some TextAsset resource to prefab, that can't drag into GameObject at editor.
47.             TextAsset ta = GetTextAsset("LoadTextAsset");//GetMaterial/GetTexture/GetAudioClip
48.             Debug.Log("Load TextAsset:" + ta.text);
49.             text.text = ta.text;
50.         }
51.         void OnClick()//response of 'ButtonStart' button
52.         {
53.             SetInt("iValueA", GetInt("iValueA") + 1);
54.             GetComponent<Text>("TextMsg").text = DateTime.Now.ToString();
55.         }
56.         void OnClickBack()//response of 'back' button
57.         {
58.             SampleCSharpLike.instance.gameObject.SetActive(true);//back to SampleCSharpLike
59.             HotUpdateManager.Hide("Sample/SampleInteractivePrefabData");//close self
60.         }
61.     }
62. }
```

C#LikeFree no need register before use (completely same with the full version)

C#Light	C#Like
You must register the type in not hot update script before use it in hot update script.	Just direct use in hot update script,no need register

```
01.  //You must register 'Vector2' in not hot update script like this
02.  RegisterType(typeof(UnityEngine.Vector2), "Vector2");
03.  //And then you can use 'Vector2' in hot update script
04.  Vector2 vector = new Vector2();
05.
06.  //UnityEngine.Vector2 vector = new UnityEngine.Vector2();
07.  //Full name is not support
08.  //Just can use one of class with the same name
09.  //but difference namespace in hot update script
10.  //RegisterType(typeof(UnityEngine.Random), "Random");
11.  //RegisterType(typeof(System.Random), "Random");
```

```
01.  //Just use like this or with full name
02.  Vector2 vector = new Vector2();
03.  UnityEngine.Vector2 vector2 = new UnityEngine.Vector2();
04.  //The class with the same name but difference namespace
05.  //are both can use with full name
06.  int random1 = UnityEngine.Random.Range(0, 100);
07.  int random2 = (new System.Random()).Next(100);
08.  //Or using alias
09.  using Random = UnityEngine.Random;
10.  int random3 = Random.Range(0, 100);
```

IL2CPP Stripping (completely same with the full version)

Type Stripping	Generic type
IL2CPP will strip the engine code if they are not be use. Unfortunately, our hot update script happens to be the strip one. You must set 'Managed Stripping Level' to 'Low' or 'Medium' (can't set to 'High'). Check which *.dll that you'll be use in the future in the folder '{Unity install folder}\Editor\Data\Managed\UnityEngine\'. And then the modify 'Assets\C#Like\link.xml', make sure your type wont be stripping by IL2CPP.	Because you can't JIT the generic type in IL2CPP,and must AOT. All the generic type must be used once in not hot update script before use in hot update script. We will automatic generate 'Assets\C#Like\Runtime\AheadOfTime\AheadOfTime.cs' while compile script. Theoretically all your generic types almost be generated when you finally release your product. Here is the file automatic generate in sample. Don't modify this file manually.

```
01.  <linker>
02.      <assembly fullname="System" preserve="all"/>
03.      <assembly fullname="mscorlib" preserve="all"/>
04.      <assembly fullname="Assembly-CSharp" preserve="all"/>
05.      <assembly fullname="UnityEngine" preserve="all"/>
06.      <assembly fullname="UnityEngine.CoreModule" preserve="all"/>
07.      <assembly fullname="UnityEngine.UIModule" preserve="all"/>
08.      <assembly fullname="UnityEngine.IMGUIModule" preserve="all"/>
09.      <assembly fullname="UnityEngine.InputModule" preserve="all"/>
10.  </linker>
```

```
01.  /*
02.   *           C#Like
03.   * Copyright © 2022 RongRong
04.   * It's automatic generate by CSharpLikeEditor,don't modify this file.
05.   */
06.
07.  namespace CSharpLike
08.  {
09.      namespace Internal
10.      {
11.          /// <summary>
12.          /// Ahead-of-
     time compile for generic type in ScriptingBackend with IL2CPP
13.          /// </summary>
14.          public class AheadOfTime
15.          {
16.              public AheadOfTime()
17.              {
18.                  new System.Collections.Generic.List<UnityEngine.Vector3>();
19.              }
20.          }
21.      }
22.  }
23.
```

C#Like usage guide (completely same with the full version)

File tree framework

```
01.  C#Like
02.      |
03.      |--Documentation
04.      |    |-C#LikeManual.pdf        //Manual of C#Like
05.      |
06.      |--Editor                      //C#LikeEditor, you can ignore it
07.      |
08.      |--HotUpdateScripts            //Default folder for hot update script.You can delete it while you final
09.      |    |                         //build your project to user, but remember backup it.
10.      |    |--Sample                 //Sample of the hot update script,you can delete this folder at any time.
11.      |    |   |--SampleInteractivePrefabData.cs            //Sample for interactive prefab data
12.      |    |   |--SampleHelloWorld.cs                       //Sample for HelloWorld and the usage of coroutine
13.      |    |   |--SampleC#                                  //Sample for C# function in hot update script
14.      |    |   |--AircraftBattle                            //An aircraft battle game sample make with hot update script
15.      |    |   |--NetObjects                                //Transfer object between client and game server
16.      |    |   |--SampleSocket.cs                           //Sample for how to use KissFramework by Socket/WebSocket
17.      |    |
18.      |    |--SampleFullVersion.zip  //Sample for the C#Like full version (work in full version only), you can find out the difference between lite version and full version
19.      |
20.      |--Resources
21.      |    |--Sample                 //The resources of the Sample,you can delete this folder at any time.
22.      |
23.      |--Runtime
24.      |    |--AheadOfTime            //Automatic generation by CSharpLikeEditor for IL2CPP,don't modify it
25.      |    |--Interaction            //The resources of the example,you can delete this folder at any time.
26.      |    |    |--CLS_Environment.cs              //The compare with C#Light/C#LikeFree/C#Like, and statement of the 'C#Light' author
```

```
    |    |    |--CSL_Tools.cs                    //custom Encrypt/Decrypt the final binary file
    |    |    |--HotUpdateBehaviour.cs           //Hot update script behaviour Which interactive with Unity prefab,
    |    |    |                                  //include 'Awake/OnEnable/Start/OnDisable/OnDestroy' behaviour only.
    |    |    |--HotUpdateManager.cs             //Hot update script manager
    |    |    |--LikeBehaviour.cs                //Hot update script base class 'MonoBehaviour'. If you want to interactive with Unity prefab,
    |    |    |                                  //your class must Inherit from LikeBehaviour.You can think of it as MonoBehaviour in hot update script.
    |    |    |--MyCustomConfig.cs               //Custom configuration
    |    |    |--MyHotUpdateManager.cs           //Custom hot update script manager
    |    |    |--HotUpdateBehaviourCustom        //You should custom your HotUpdateBehaviour base on HotUpdateBehaviour,
    |    |        |                             //so you can include more behaviours except 'Awake/OnEnable/Start/OnDisable/OnDestroy' behaviour
    |    |        |--HotUpdateBehaviourAll.cs            //Include 'ALL' behaviours in MonoBehaviour.It's a backup plan in case for
    |    |        |                                      //you forgot add your custom behaviour,we don't recommend use it.
    |    |        |--HotUpdateBehaviourApplication.cs    //Include 'OnApplicationFocus/OnApplicationPause/OnApplicationQuit' behaviours
    |    |        |--HotUpdateBehaviourCollision.cs      //Include 'OnCollisionEnter/OnCollisionExit/OnCollisionStay' behaviours
    |    |        |--HotUpdateBehaviourCollision2D.cs    //Include 'OnCollisionEnter2D/OnCollisionExit2D/OnCollisionStay2D' behaviours
    |    |        |--HotUpdateBehaviourCommon.cs         //Include 'FixedUpdate/LateUpdate/Update/OnGUI' behaviours
    |    |        |--HotUpdateBehaviourFixedUpdate.cs    //Include 'FixedUpdate' behaviour
    |    |        |--HotUpdateBehaviourLateUpdate.cs     //Include 'LateUpdate' behaviour
    |    |        |--HotUpdateBehaviourOnGUI.cs          //Include 'OnGUI' behaviour
    |    |        |--HotUpdateBehaviourTrigger.cs        //Include 'OnTriggerEnter/OnTriggerExit/OnTriggerStay' behaviours
    |    |        |--HotUpdateBehaviourTrigger2D.cs      //Include 'OnTriggerEnter2D/OnTriggerExit2D/OnTriggerStay2D' behaviours
    |    |        |--HotUpdateBehaviourUpdate.cs         //Include 'Update' behaviour
    |    |        |--HotUpdateBehaviourUpdateTrigger2D.cs //Include 'OnCollisionEnter2D/OnCollisionExit2D/OnTriggerStay2D/Update' behaviours
    |    |
    |    |--Internal                             //The internal core of the C#Like, you can ignore it.
    |    |--KissCSV                              //Read CSV file specially made for C#Like.
    |    |--KissJson                             //JSON serialize library specially made for C#Like.
    |    |--Sample                               //The not hot update script of the Sample,you can delete this folder at any time.
    |         |--SampleCSharpLike.cs             //The sample for show how to use C#Like.Include initialize C#Like
    |         |                                  //and load hot update script binary file and call prefab which using hot update script.
    |         |--SampleNotHotUpdateScript.cs     //The not hot update script use by sample
    |         |--MyAnchor.cs                     //The scene anchor use by sample
    |         |--MyRoot.cs                       //The scene root use by sample
    |
    |--Scenes
    |    |--SampleScene.unity                   //The scene of the example,you can delete this folder at any time.
    |
    |--Plugins
         |--WebGL                                //WebSocket lib use in WebGL.
```

Usage guide for the whole framework(C#LkieFree+KissServerFramework). If you just care about C#LkieFree, you can just need to read step 6.

```
1 Setup 'KissServerFramework' server
    1.1 Download source code of the server, and then using Visual Studio complie and public into a single EXE file(You can direct use the EXE file that I public if your don't want to complie yourself),put it to 'C:\KissServerFr
        C:\KissServerFramework
            |
            |--CSV
            |  |--Item.csv                    //CSV file for test only
            |
            |--KissServerFramework.exe        //Main EXE file
            |--KissServerFramework.json       //config JSON file
            |--kiss.sql                       //The SQL file use in step 3 only

    1.2 If your computer did not installed .NET5 runtime library, you should go to Microsoft web site download and install it. https://download.visualstudio.microsoft.com/download/pr/a0832b5a-6900-442b-af79-
6ffddddd6ba4/e2df0b25dd851ee0b38a86947dd0e42e/dotnet-runtime-5.0.17-win-x64.exe
    1.3 You can modify WebSocket port(default is 9000) and Socket port(default is 9001) and HTTP port(default is 9002) in 'KissServerFramework.json' IF you need to.
    1.4 Open the 9000-9002 TCP port IF your computer protected by Firewall.

2 Setup XAMPP
    2.1 Go to XAMPP web site download it. https://www.apachefriends.org/download.html
    2.2 Click 'Next' until XAMPP install done.(Default install to 'C:/xampp')
    2.3 Make sure both the MySQL and Apache is running, you can check it in 'XAMPP Control Panel'.
    2.4 Open the 80 TCP port IF your computer protected by Firewall.

3 Create Database 'kiss' and import from 'kiss.sql'
    3.1 Double-click to run 'files/CreateDatabase.bat' to create database 'kiss'. (Click 'Enter' whilewhile ask for password because no password as default). Modify the BAT file if your install folder is not 'C:/xampp'.
    3.2 Double-click to run 'files/ImportDatabase.bat' to import from 'kiss.sql'. (Click 'Enter' whilewhile ask for password because no password as default). Modify the BAT file if your install folder is not 'C:/xampp'.

4 (Optional step)HTTP upgrade to HTTPS, WS upgrade to WSS, change RSA certificate that use in Socket
    4.1 Prefare SSL certificate, You can go to Tencent( https://console.cloud.tencent.com/ssl ) or "Let's Encrypt"
( https://letsencrypt.org/ )apply for a free SSL certificate or buy one. I was apply the free one year SSL certificate from Tencent, and download the Apache version and Nginx version reserve for next step.
    4.2 Setup Nginx proxy
        4.2.1 Go to Nginx web site download it. http://nginx.org/en/download.html Normaly we download the stable version, we take 'nginx.Windows-1.22.1' for example.
        4.2.2 Unzip the 'nginx-1.22.1.zip' to 'C:\'.
        4.2.3 Copy 'files/QuitNginx.bat' 'files/RetartNginx.bat' 'files/StartNginx.bat' 'files/StopNginx.bat' to Nginx folder.
        4.2.4 "files/nginx.conf" replace "C:\nginx-1.22.1\conf\nginx.conf"
        4.2.5 Unzip the Nginx version SSL certificate to 'C:\nginx-1.22.1\conf\xxxx.com\'
            Final file tree:
            C:\nginx-1.22.1
                |
                |--conf
                |  |--xxxx.com             //Nginx version SSL certificate
                |  |   |--xxxx.com_bundle.crt
                |  |   |--xxxx.com.key
                |  |
                |  |--nginx.conf           //Nginx config file
                |
                |--contrib
                |--docs
                |--html
                |--logs
                |--temp
                |--nginx.exe
                |
                |--QuitNginx.bat          //Double-click it to quit Nginx
                |--RetartNginx.bat        //Double-click it to restart Nginx, use it after modify 'nginx.conf'
                |--StartNginx.bat         //Double-click it to start Nginx, use it in the first run Nginx
                |--StopNginx.bat          //Double-click it to stop Nginx

        4.2.6 Modify 'C:/nginx-1.22.1/conf/nginx.conf', change the 'xxxx.com' into your real domain. In my case I use 'csharplike.com' replace 'xxxx.com'
        4.2.7 You can change the WSS port and WS port and HTTPs port and HTTP port IF you need to. Open the 10000 and 10002 TCP port IF your computer protected by Firewall.
        4.2.8 Modify 'C:\KissServerFramework\KissServerFramework.json' because we use WSS and HTTPS now:
            "WebSocketURI": "wss://www.xxxx.com:10000" INSIDE the 'serverInfos' node
            "HttpURI": "https://www.xxxx.com:10002" INSIDE the 'serverInfos' node
        4.2.9 Finally remember execute 'StartNginx.bat' to start Nginx (Execute 'RetartNginx.bat' IF you had been run Nginx).

    4.3 Setup Apache SSL certificate
        4.2.1 Unzip the Apache version SSL certificate to 'C:\xampp\apache\conf\xxxx.com\'
        4.2.2 Modify 'C:\xampp\apache\conf\extra\httpd-ssl.conf'
            SSLCertificateFile "C:/xampp/apache/conf/xxxx.com/xxxx.com.crt"
            SSLCertificateKeyFile "C:/xampp/apache/conf/xxxx.com/xxxx.com.key"
            SSLCertificateChainFile "C:/xampp/apache/conf/xxxx.com/root_bundle.crt"
            Please replace the 'xxxx.com' to your real domain. In my case I use 'csharplike.com' replace 'xxxx.com'
        4.2.3 Open the 443 TCP port IF your computer protected by Firewall.

    4.4 Restart the Apache server make the config take effect. (Click Start button after click Stop button in 'XAMPP Control Panel')

    4.5 Our socket encrypt by RSA first and then encrypt by AES, you should modify the default RSA certificate to your unique.
        4.5.1 Generate RSA certificate
            'Menu/Window/C#Like' open the setting panel of C#Like, click 'Generate RSA' button, will generate 'Assets\C#Like\Editor\RSAPublicKey.txt' and 'Assets\C#Like\Editor\RSAPrivateKey.txt' two files.
        4.5.2 Modify the KissServerFramework config
            Modify 'socketServerRSAPrivateKey' value in 'C:\KissServerFramework\KissServerFramework.json' to the conten of 'Assets\C#Like\Editor\RSAPrivateKey.txt'
        4.5.3 Modify the C#Like config
            Modify 'socketRSAPublicKey' value in 'Assets\C#Like\HotUpdateScripts\Sample\SampleSocket.cs' to the conten of 'Assets\C#Like\Editor\RSAPublicKey.txt'

5 Double-click 'C:/KissServerFramework/KissServerFramework.exe' to start KissServerFramework
    5.1 If the EXE crash down, that may be not exist .NET runtime library, your should check the tips in 'Event Viewer' and download it from Microsoft web site.
    5.2 If the console just only show 'KissFramework version : 1.0.x.x' mean the port was used. You can click 'NetStat' button in 'XAMPP Control Panel' to check whick port was used.

6 Export C#Like or C#LikeFree to WebGL platform
    6.1 Create a empty 2D Unity project.
    6.2 Import C#Like or C#LikeFree plugin.
    6.3 Modify the HTTP connect domain and port in "Assets\C#Like\Runtime\Sample\SampleCSharpLike.cs"
        6.3.1 e.g. In HTTP:
            https://www.csharplike.com:10002/ReqGateway => http://127.0.0.1:9002/ReqGateway
        6.3.2 e.g. In HTTPS:
            https://www.csharplike.com:10002/ReqGateway => https://[Your domain]:[Your port]/ReqGateway
    6.4 'Menu/Window/C#Like' open the setting panel of C#Like. Click 'Rebuild Scripts' button, compile all '*.cs' files in 'Assets\C#Like\HotUpdateScripts' into binary file 'Assets\StreamingAssets\output.bytes'.
    6.5 (Optional step) : Delete the hold 'Assets\C#Like\HotUpdateScripts' folder after backup, to verify that our hot update script is really can be hot update!
    6.6 Click '/Assets/C#Like/Scenes/SampleScene.unity' to open the demo scene in Unity editor.
    6.7 'Menu/File/Build Settings...' open 'Build Settings' panel.
        6.7.1 Click 'Add Open Scenes' button, make 'C#Like/Scenes/SampleScene' to be the first scens in 'Scenes In Build'
        6.7.2 Choose 'WebGL' and then click 'Switch Platform' button, switch the current plaform to WebGL
        6.7.3 Click 'Player Setting...' button, and then modify as you need.
            6.7.3.1 "Resolution and Presentation"
                "Default Canvas Width" 600
                "Default Canvas Height" 960
                "Run In Background" "v"
            6.7.3.2 "Other Settings"
                "Api Compatibility Level" ".NET Standard 2.0"
                "Strip Engine Code" "v"
                "Managed Stripping Level" "Medium"
            6.7.3.3 "Publishing Settings"
                "Enable Exceptions" "Explicitly Thrown Exceptions Only"
                "WebAssembly Arithmetic Exceptions" "Throw"
                "Compression Format" "Gzip"
                "Data Caching" "v"
                "Decompression Fallback" "v"
```

```
        6.7.4 Click 'Build' button to export the final WebGL folder
            6.7.4.1 e.g. C#Like export to folder CSharpLikeDemo
            6.7.4.2 e.g. C#LikeFree export to folder CSharpLikeFreeDemo

7 Copy exported folder to 'C:\xampp\htdocs'
    7.1 e.g. Copy the 'CSharpLikeDemo' that exported in step '6.7.4.1' to 'C:\xampp\htdocs'
        7.1.1 e.g. in my case visit the demo by link : https://www.csharplike.com/CSharpLikeDemo/index.html
        7.1.2 e.g. in the local with no SSL certificate, you can visit the demo by link : http://127.0.0.1/CSharpLikeDemo/index.html
    7.2 e.g. Copy the 'CSharpLikeFreeDemo' that exported in step '6.7.4.2' to 'C:\xampp\htdocs'
        7.1.1 e.g. in my case visit the demo by link : https://www.csharplike.com/CSharpLikeFreeDemo/index.html
        7.1.2 e.g. in the local with no SSL certificate, you can visit the demo by link : http://127.0.0.1/CSharpLikeFreeDemo/index.html
```

Convenient KissJson

Can use both in normal script and hot update script, you can easy convert between JSON string and class/struct(include hot update script and normal script), and the JSONData is supper easy to use.

```
using CSharpLike.SubclassEx3;
using System.Collections.Generic;
using UnityEngine;

namespace CSharpLike
{
    public partial class SampleCSharp : LikeBehaviour
    {
        void TestKissJson()
        {
            Debug.LogError("Test TestKissJson:You can't use @ keyword to define JSON string and using Nullable type and using bit operation with JSONData in " +
            "hot update script in FREE version. (Supported in full version). Strongly recommended update to full version C#Like: " +
            "https://assetstore.unity.com/packages/slug/222256");
            // We provide the following solutions if you not update to full version:
            // @ keyword: direct use "" instead of @"".
            // bit operation: use function instead of bit operation.
            // Nullable: don't use Nullable type.

            //Build-In-Type::
            //byte/sbyte/char/bool/short/ushort/int/uint/long/ulong/double/float/string,
            //byte?/sbyte?/char?/bool?/short?/ushort?/int?/uint?/long?/ulong?/double?/float?,
            //List<Build-In-Type:>,
            //Dictionary<string, Build-In-Type:>

            //Direct assign value 'both' between Build-In-Type and JSONData.
            //You can direct assign Build-In-Type value to JSONData, Or direct assign JSONData value to Build-In-Type.

            //test Build-In-Type => JSONData
            JSONData iData = 2;
            Debug.Log("JSONData iData = 2;  test iData = " + iData);//output 2
            JSONData fData = 88.8f;
            Debug.Log("JSONData fData = 88.8f;  test fData = " + fData);//output 88.8
            List<string> listValue = new List<string>();
            listValue.Add("test list str1");
            listValue.Add("test list str2");
            JSONData listData = listValue;
            Debug.Log("JSONData listData = listValue;  test listData = " + listData);//output ["test list str1","test list str2"]
            Dictionary<string, int> dictValue = new Dictionary<string, int>();
            dictValue.Add("key1", 11);
            dictValue.Add("key2", 22);
            JSONData dictData = dictValue;
            Debug.Log("JSONData dictData = dictValue;  test dictData = " + dictData);//output {"key1":11,"key2":22}

            //test JSONData =>  Build-In-Type
            int iValue = iData;
            Debug.Log("int iValue = iData;  test iValue = " + iValue);//output 2
            float fValue = fData;
            Debug.Log("float fValue = fData;  test fValue = " + fValue);//output 88.8
            List<string> listValue2 = listData;
            string strTemp = "";
            foreach (var str in listValue2)
                strTemp += str + ",";
            Debug.Log("List<string> listValue2 = listData;  test listValue2 = " + strTemp);//output test list str1,test list str2,
            Dictionary<string, int> dictValue2 = dictData;
            strTemp = "";
            foreach (var item in dictValue2)
                strTemp += "(" + item.Key + "=" + item.Value + ")";
            Debug.Log("Dictionary<string, int> dictValue2 = dictData;  test dictValue2 = " + strTemp);//output (key1=11)(key2=22)

            //test JSON string => JSONData
            string strJson = "{\"str\":\"{test str\",\"i\":11,\"j\":2.3,\"k\":[3,null,{\"m\":true}],\"l\":{\"x\":1,\"y\":\"abc\"}}";
            JSONData data = KissJson.ToJSONData(strJson);
            //accept JSONData by ["key"] and [index]
            Debug.Log("JSON string => JSONData; test data[\"str\"] = " + data["str"]);//output {test str
            Debug.Log("JSON string => JSONData; test data[\"i\"] = " + data["i"]);//output 11
            Debug.Log("JSON string => JSONData; test data[\"j\"] = " + data["j"]);//output 2.3
            Debug.Log("JSON string => JSONData; test data[\"k\"] = " + data["k"]);//output [3,null,{"m":true}]
            Debug.Log("JSON string => JSONData; test data[\"l\"] = " + data["l"]);//output {"x":1,"y":"abc"}
            Debug.Log("JSON string => JSONData; test data[\"k\"][0] = " + data["k"][0]);//output 3
            Debug.Log("JSON string => JSONData; test data[\"l\"][\"y\"] = " + data["l"]["y"]);//output abc
            Debug.Log("JSON string => JSONData; test data[\"k\"][2][\"m\"] = " + data["k"][2]["m"]);//output true

            //test Math Expression between JSONData with Build-In-Type
            JSONData exprData1 = 2;
            JSONData exprData2 = 3;
            JSONData exprData3 = exprData1 * exprData2;//test +-*/
            Debug.Log("test Math Expression;  exprData3 = exprData1 * exprData2; exprData3 = " + exprData3);//output 6
            exprData3 = exprData1 - exprData2;
            Debug.Log("test Math Expression;  exprData3 = exprData1 - exprData2; exprData3 = " + exprData3);//output -1
            exprData3 *= exprData2;//exprData3=-1;exprData2=3
            Debug.Log("test Math Expression;  exprData3 *= exprData2; exprData3 = " + exprData3);//output -3

            iData = 2;
            if (iData > 1)//test compare operation
                Debug.Log("test Math Expression;  iData = 2, enter if (iData > 1)");//output
            else
                Debug.Log("test Math Expression;  iData = 2, not enter if (iData > 1)");

            //test JSONData => JSON string
            JSONData listData3 = JSONData.NewDictionary();//create a JSONData object
            listData3.Add("key1", 10);          //add data like Dictionary use function 'Add(key,value)'
            listData3["key2"] = "test string"; //add data like Dictionary use index set 'this[]'
            listData3["key3"] = JSONData.NewList(); //we add a list
            if (listData3.ContainsKey("key3"))  //make sure the key 'key3' exist if you don't know whether exist(just for example here, actually, we don't need to)
                listData3["key3"].Add(1);       //add some data to the list
            listData3["key3"].Insert(0,"string2"); //insert some data to the list,we don't check the key 'key3' exist because we just know it exist!
            listData3["key4"] = JSONData.NewDictionary(); //we add a JSONData to it
            listData3["key4"]["x"] = 1;
            listData3["key4"]["y"] = 2;
            listData3["key4"]["z"] = 3;
            Debug.Log("test JSONData => JSON string; strJson = " + listData3.ToJson());//output {"key1":10,"key2":"test string","key3":["string2",1],"key4":
{"x":1,"y":2,"z":3}}

            //test looping through the List or Dictionary in JSONData
            foreach (var item in listData3["key3"].Value as List<JSONData>)//foreach looping through the List
            {
                Debug.Log("test looping through the List in JSONData using foreach; listData3[\"key3\"].Value = " + item);//output string2/output 1
            }
            List<JSONData> tempList = listData3["key3"].Value as List<JSONData>;//for looping through the List
            for (int i=0; i< tempList.Count; i++)
                Debug.Log("test looping through the List in JSONData using for; listData3[\"key3\"].Value = " + tempList[i]);//output string2/output 1
            foreach (var item in listData3["key4"].Value as Dictionary<string, JSONData>)//foreach looping through the Dictionary
            {
                Debug.Log("test looping through the Dictionary in JSONData using foreach; listData3[\"key4\"], Key=" + item.Key + ",Value=" + item.Value);
                //output Key=x,Value=1/output Key=y,Value=2/output Key=z,Value=3
            }

            //test JSON string => class/struct
            strJson = "{\"str\":\"{test str\",\"i\":11,\"j\":1,\"k\":[3,1,7],\"datas\":{\"aa\":{\"id\":1,\"name\":\"aaa\",\"v2\":{\"x\":1,\"y\":2},\"info\":
[\"a\",\"xd\",\"dt\"],\"maps\":{\"x\":1,\"y\":2}},\"bb\":{\"id\":2,\"name\":\"bbb\",\"v2\":{\"x\":3,\"y\":4},\"info\":[\"x\",\"x3d\",\"ddt\"],\"maps\":
{\"x\":2,\"y\":3}}},\"data\":{\"id\":3,\"name\":\"ccc\",\"v2\":{\"x\":3,\"y\":1},\"info\":[\"ya\",\"xyd\",\"drt\"],\"maps\":{\"x\":3,\"y\":4}}}";
            TestJsonData testJsonDdata = (TestJsonData)KissJson.ToObject(typeof(TestJsonData), strJson);//JSON string => class/struct
            //test data after convert
            Debug.Log(testJsonDdata.str);//output Null
            Debug.Log(testJsonDdata.i);//output 11
            //"j":"Monday" or "j":"1" or "j":1 are both identified as 'DayOfWeek.Monday'
            //recommend use "j":1 because ToJson output as number
            Debug.Log(testJsonDdata.j);//output Monday
            Debug.Log((int)testJsonDdata.j);//output 1
            Debug.Log(testJsonDdata.z);//output 2
            Debug.Log(HotUpdateManager.ConvertEnumString(typeof(TestHotUpdateEnum),testJsonDdata.z));//output Evening
            foreach (var item in testJsonDdata.k)
                Debug.Log(item);//output 3/output 1/output 7
            foreach(var datas in testJsonDdata.datas)
            {
                Debug.Log(datas.Key);//output aa/output bb
```

```
                Debug.Log(datas.Value.v2);//output (1.0, 2.0)/output (3.0, 4.0)
            }

            //test class => JSON string
            strTemp = KissJson.ToJson(testJsonDdata);//class => JSON string
            Debug.Log(strTemp);//output {"i":11,"j":1,"k":[3,1,7],"datas":{"aa":{"id":1,"name":"aaa","v2":{"x":1,"y":2},"info":["a","xd","dt"],"maps":
{"x":1,"y":2}},"bb":{"id":2,"name":"bbb","v2":{"x":3,"y":4},"info":["x","x3d","ddt"],"maps":{"x":2,"y":3}}},"data":{"id":3,"name":"ccc","v2":{"x":3,"y":1},"info":
["ya","xyd","drt"],"maps":{"x":3,"y":4}}}
        }
    }
    namespace SubclassEx3
    {
        /// <summary>
        /// test class <=> JSON
        /// </summary>
        public class TestJsonDataSub
        {
            public string name;
            public Vector2 v2;//you can add other class/struct type(in hot update script or normal),such as Color/Rect/Vector3/...
            public List<string> info;//direct is List
            public Dictionary<string, int> maps;//can convert a Dictionary which key type is string
        }
        /// <summary>
        /// test JSON string <=> class/struct, be mark as KissJsonDontSerialize
        /// </summary>
        [KissJsonDontSerialize]//mark the class as KissJsonDontSerialize,will ignore while serialize JSON
        public class TestJsonDataSub2
        {
            public int id;
        }
        /// <summary>
        /// test class/struct <=> JSON
        /// </summary>
        public class TestJsonData
        {
            [KissJsonDontSerialize]
            public string str;//will ignore while serialize JSON because be mark as KissJsonDontSerialize
            public int i;
            public DayOfWeek j;//test enum of not hot update
            public List<int> k;
            public Dictionary<string, TestJsonDataSub> datas;//test Dictionary for class/struct
            public TestJsonDataSub data;//test single class/struct
            public TestJsonDataSub2 data2;//will ignore while serialize JSON because the class 'TestJsonDataSub2' mark as KissJsonDontSerialize
        }
    }
}
```

C# feature of C#LikeFree

class feature

Here are a interface IAnimal, and then class Mammals inherit IAnimal. This sample show the class feature.

```
using UnityEngine;
using CSharpLike.Subclass;

namespace CSharpLike
{
    public partial class ExampleCSharp : LikeBehaviour
    {
        /// <summary>
        /// test class feature.
        /// </summary>
        void TestClass()
        {
            Debug.LogError("Test class: Not supported 'constructor(this/base)/destructor/class inherit/virtual function' in FREE version. (Supported in full version). "+
            "Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
            // We provide the following solutions if you not update to full version:
            // constructor(this/base)/destructor: use normal function and call it by manual.
            // class inherit/virtual function: don't use it, C language can do everything even it have no class, you can do it too.
            // That cause your code not so object-oriented only, you still can have a class and can inherit some interfaces.

            //test class
            Mammals cow = new Mammals(4, 4, "cow");
            Debug.Log("cow:" + cow.GetInfo());
            //test interface
            IAnimal bull = new Mammals(0, 4, "bull");
            bull.female = false;
            Debug.Log("bull:female=" + bull.female + ", CanUseTool=" + bull.CanUseTool());
        }
    }
}
```

```
using UnityEngine;

namespace CSharpLike
{
    /// <summary>
    /// test for class in 'ExampleC#_Class.cs'
    /// </summary>
    namespace Subclass
    {
        public interface IAnimal
        {
            int feet { get; set; }
            bool female { get; set; }
            bool CanUseTool();
        }

        public class Mammals : IAnimal
        {
            public int breasts;
            public string name;

            public int feet { get; set; }
            public bool female { get; set; }

            public Mammals(int breasts, int feet, string name)
            {
                Debug.Log("Mammals(" + breasts + "," + feet + "," + name + ")");
                this.breasts = breasts;
            }

            public string GetInfo()
            {
                return "Mammals:" + name + " with " + feet + " feet and " + breasts + " breasts";
            }
            public bool CanUseTool()
            {
                return false;
            }
        }
    }
    /// <summary>
    /// test for namespace in 'ExampleC#_UsingAndNamespace.cs'
    /// </summary>
    namespace SubclassEx
    {
        /// <summary>
        /// this's test namespace with CSharpLike.Subclass.Mammals
        /// </summary>
        public class Mammals
        {
            public int breasts = 2;
            public int eyes = 2;
            public string TestNameSpace()
            {
                return "Mammals:breasts:" + breasts + ", eyes:" + eyes;
            }
        }
        public class Toys
        {
            public string name;
            public Toys(string name)
            {
                this.name = name;
            }
        }
    }
    /// <summary>
    /// test for namespace in 'ExampleC#_UsingAndNamespace.cs'
    /// </summary>
    namespace SubclassEx2
    {
```

```
        public class TestNamespace
        {
            public static string GetTestString()
            {
                return "Test string for namespace";
            }
        }
    }
}
```

Delegate and Lambda

We have 3 buttons('Test Delegate','Test Lambda','Test Bind') and a Text component in the interface.

```
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;

namespace CSharpLike
{
    public partial class ExampleCSharp : LikeBehaviour
    {
        void TestDelegateAndLambda()
        {
            Debug.LogError("Test delegate and lambda:");
            //test Delegate: We binding 'OnClickDelegate' as the response function of 'Test Delegate' button by code.
            HotUpdateManager.AddEventTrigger(GetGameObject("TestDelegate"), EventTriggerType.PointerClick, OnClickDelegate);
            //test Lambda: We binding Lambda expression as the response function of 'Test Lambda' button by code.
            HotUpdateManager.AddEventTrigger(GetGameObject("TestLambda"), EventTriggerType.PointerClick,
                (BaseEventData eventData) =>
                {
                    GetComponent<Text>("TestMessage").text = "OnClickLambda";
                    Debug.Log("On click lambda :" + eventData);
                });
        }
        /// <summary>
        /// Binding by delegate.
        /// The response function of 'Test Delegate' button
        /// </summary>
        void OnClickDelegate(BaseEventData eventData)
        {
            GetComponent<Text>("TestMessage").text = "OnClickDelegate";
            Debug.Log("OnClickDelegate:" + eventData);
        }
        /// <summary>
        /// Call by Button Component of 'ButtonTestBind' in prefab.
        /// This's the most simplest way bind OnClick event by Button.
        /// </summary>
        void OnClickBindButton()
        {
            GetComponent<Text>("TestMessage").text = "OnClickBindButton";
            Debug.Log("OnClickBindButton:");
        }
        /// <summary>
        /// Call by EventTrigger Component of 'ButtonTestBind' in prefab.
        /// It's not recommend because only one EventTriggerType in one LikeBehaviour,and can't change the method name.
        /// Recommend use HotUpdateManager.AddEventTrigger.
        /// </summary>
        void OnPointerEnter(BaseEventData eventData)
        {
            GetComponent<Text>("TestMessage").text = "OnPointerEnter";
            Debug.Log("OnPointerEnter:" + eventData);
        }
    }
}
```

Math Expression

```
using UnityEngine;

namespace CSharpLike
{
    public partial class ExampleCSharp : LikeBehaviour
    {
        /// <summary>
        /// test math expression:
        /// + - * / % += -= *= /= %= > >= < <= != == && || ! ++ -- is as & | ~ ^ &= |= ^= << >> <<= >>= ? ?? ?. ?:
        /// </summary>
        void TestMathExpression()
        {
            Debug.LogError("Test MathExpression:Not supported 'bit operation' and Nullable in FREE version. (Supported in full version). "+
            "Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
            // We provide the following solutions if you not update to full version:
            // bit operation: use function instead of bit operation.
            // Nullable: don't use Nullable type.

            int i = 1;
            int j = 2;
            int k = 100;
            //test + - * / %
            Debug.Log("i = 1, j = 2, test i + j = " + (i + j));//output 3
            Debug.Log("i = 1, j = 2, test i - j = " + (i - j));//output -1
            Debug.Log("i = 1, j = 2, test i * j = " + (i * j));//output 2
            Debug.Log("i = 1, j = 2, test i / j = " + (i / j));//output 0
            Debug.Log("i = 1, j = 2, test i % j = " + (i % j));//output 1

            //test += -= *= /= %=
            k += j;
            Debug.Log("j = 2, k = 100, test k += j then k = " + k);//output 102
            k = 100;
            k -= j;
            Debug.Log("j = 2, k = 100, test k -= j then k = " + k);//output 98
            k = 100;
            k *= j;
            Debug.Log("j = 2, k = 100, test k *= j then k = " + k);//output 200
            k = 100;
            k /= j;
            Debug.Log("j = 2, k = 100, test k /= j then k = " + k);//output 50
            k = 100;
            k %= j;
            Debug.Log("j = 2, k = 100, k %= j then k = " + k);//output 0

            //test > >= < <= != ==
            Debug.Log("i = 1, j = 2, test (i < j) = " + (i < j));//output True
            Debug.Log("i = 1, j = 2, test (i <= j) = " + (i <= j));//output True
            Debug.Log("i = 1, j = 2, test (i == j) = " + (i == j));//output False
            Debug.Log("i = 1, j = 2, test (i != j) = " + (i != j));//output True
            Debug.Log("i = 1, j = 2, test (i > j) = " + (i > j));//output False
            Debug.Log("i = 1, j = 2, test (i >= j) = " + (i >= j));//output False

            //test && || !
            bool b1 = true;
            bool b2 = false;
            Debug.Log("b1 = true, b2 = false, test (b1 && b2) = " + (b1 && b2));//output False
            Debug.Log("b1 = true, b2 = false, test (b1 || b2) = " + (b1 || b2));//output True
            Debug.Log("b1 = true, test (!b1) = " + (!b1));//output False

            //test ++ --(include prefix and suffix)
            k = 100;
            Debug.Log("k = 100, test (k++) = " + (k++));//output 100
            Debug.Log("finally k = " + k);////output 101
            k = 100;
            Debug.Log("k = 100, test (++k) = " + (++k));//output 101
            Debug.Log("finally k = " + k);////output 101
            k = 100;
            Debug.Log("k = 100, test (k--) = " + (k--));//output 100
            Debug.Log("finally k = " + k);////output 99
            k = 100;
            Debug.Log("k = 100, test (--k) = " + (--k));//output 99
            Debug.Log("finally k = " + k);////output 99

            // We are not support '++' '--' with index get/set '[]',
            // such as List or Dictionary or JSONData.
            //// List<int> lists = new List<int>();
            //// lists.Add(1);
            //// lists[0]++;//compiler error,suggest use 'lists[0] += 1;' instead
            //// ++lists[0];//runtime error,suggest use 'lists[0] += 1;' instead

            //test convert: is as
            object o = i;
            Debug.Log("int i = 1,object o = i, test (o is int) = " + (o is string));//output False
            Debug.Log("int i = 1,object o = i, test (o as string) = " + (o as string));//output null
        }
    }
}
```

loop syntax

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

namespace CSharpLike
{
    public partial class ExampleCSharp : LikeBehaviour
    {
        /// <summary>
        /// test loop syntax
        /// for foreach continue break if-else return while do-while switch-case-default.
        /// </summary>
        void TestLoop()
        {
            Debug.LogError("Test Loop:Not supported custom 'switch-case-default' in FREE version. (Supported in full version). "+
            "Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
            // We provide the following solutions if you not update to full version:
            // switch-case-default: direct use "if-else" (Look ugly and a little lower performance if too much branches).

            int x = 1;
            int y = 3;
            int z = 100;

            //test if-else
            if (x < y)
                Debug.Log("test 'if-else': enter 'if'");
            else if (x < z)
                Debug.Log("test 'if-else': enter 'else if'");
            else
                Debug.Log("test 'if-else': enter 'else'");

            List<Vector3> lists = new List<Vector3>();// this generic type should be AOT
            lists.Add(Vector3.zero);
            lists.Add(Vector3.one);
            //test for
            for (int i = 0, j = 2; i < lists.Count; i++, j += i)
            {
                if (i == 2)
                {
                    Debug.Log("test 'continue':");
                    continue;
                }
                Debug.Log("test 'for': lists[" + i + "] = " + lists[i] + ",j = " + j);
            }

            //test foreach
            foreach(var item in lists)
            {
                Debug.Log("test 'foreach': item = " + item);
                if (item.x == 2)
                {
                    Debug.Log("test 'break':");
                    break;
                }
            }

            //test while
            whilewhile(x < y)
            {
                x++;
                Debug.Log("test 'while': x = " + x + ", y = " + y);
            }

            //test do-while
            x = 1;
            do
            {
                x++;
                Debug.Log("test 'do-while': x = " + x + ", y = " + y);
            } whilewhile (x < y) ;

            //test switch (simulate)
            //in FREE version:
            if (x == 0)
                Debug.Log("test 'switch': enter 0");
            else if (x == 1)
                Debug.Log("test 'switch': enter 1");
            else if (x == 2 || x == 3)
                Debug.Log("test 'switch': enter 2 or 3");
            else
                Debug.Log("test 'switch': enter default");
            ////in full version:
            //switch(x)
            //{
            //    case 0:
            //        Debug.Log("test 'switch': enter 0");
            //        break;
            //    case 1:
            //        Debug.Log("test 'switch': enter 1");
            //        break;
            //    case 2:
            //    case 3:
            //        Debug.Log("test 'switch': enter 2 or 3");
            //        break;
            //    default:
            //        Debug.Log("test 'switch': enter default");
            //        break;
            //}
        }
    }
}
```

custom get/set accessor

```
using UnityEngine;

namespace CSharpLike
{
    public partial class ExampleCSharp : LikeBehaviour
    {
        void TestGetSetAccessor()
        {
            Debug.LogError("Test TestGetSetAccessor: Not supported 'custom implement get/set accessor' in FREE version. (Supported in full version). "+
            "Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
            // We provide the following solutions if you not update to full version:
            // custom implement get/set accessor: direct use function instead.

            //test auto implement get set accessor
            Debug.Log("before set value testGetSetAutoImp = " + testGetSetAutoImp);//output False
            testGetSetAutoImp = true;
            Debug.Log("after set value: testGetSetAutoImp = " + testGetSetAutoImp);//output True
        }

        public bool testGetSetAutoImp { get;set; }
    }
}
```

multi-threading

```
using System;
using System.Collections.Generic;
using System.Threading;
using System.Threading.Tasks;
using UnityEngine;
using UnityEngine.UI;
using Random = UnityEngine.Random;

namespace CSharpLike
{
    public partial class SampleCSharp : LikeBehaviour
    {
        void TestThread()
        {
            if (Application.platform != RuntimePlatform.WebGLPlayer)
            {
                Debug.LogError("Test Thread: Not supported 'lock syntax' in FREE version. (Supported in full version). "+
                "Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
                // We provide the following solutions if you not update to full version:
                // lock syntax: Do it in not hot update script. Or don't use lock (May cause multi-threading problem).
```

```
                //easy use for thread with no param
                HotUpdateManager.CreateThread(TestThreadRunLoop);

                //Task.Run
                Task.Run(() =>
                {
                    Debug.LogError("Task.Run as lambda start " + DateTime.Now);
                    Thread.Sleep(2000);
                    Debug.LogError("Task.Run as lambda end " + DateTime.Now);
                });
                Task.Run(TestTaskRun);
            }
            else
                Debug.LogError("Test Thread:WEBGL can't use thread.");
        }
        void TestTaskRun()
        {
            Debug.LogError("Task.Run as delegate start " + DateTime.Now);
            Thread.Sleep(4000);
            Debug.LogError("Task.Run as delegate end " + DateTime.Now);
        }
        /// <summary>
        /// Call by Button Component of 'ButtonTestThread' in prefab.
        /// </summary>
        void OnClickTestThread()
        {
            GetComponent<Text>("TestMessage").text = "OnClickTestThread";
            //easy use for thread with param
            JSONData jsonData = JSONData.NewDictionary();
            jsonData["id"] = Random.Range(1, 1000);
            jsonData["data"] = JSONData.NewList();
            jsonData["data"].Add("dump1");
            jsonData["data"].Add("dump2");
            List<int> testParams = new List<int>();
            testParams.Add(123);
            jsonData.SetObjectExtern("TestExternMsg", testParams);//Set extern params to thread
            HotUpdateManager.CreateThread(DoSomeWorkInThread,//This's the function which call in thread
                jsonData,//This's the param which call in thread and callback to main thread.
                behaviour,//the current behaviour.let it null if you don't care the callback
                OnDoSomeWorkInThreadDone);  //This's the function which callback in main thread.
                                            //let it null if you don't care the callback

            //let 'TestThreadRunLoop' print log by change the value
            countInThread = Random.Range(1, 1000);
        }
        bool bExist = false;
        void OnDestroy()
        {
            bExist = true;//notify thread to stop while main thread about to exist
        }
        int countInThread = 0;
        volatile int countLoopThread = 0;//test 'volatile' keyword
        /// <summary>
        /// work in thread with param
        /// </summary>
        void TestThreadRunLoop()
        {
            Debug.LogError("TestThreadRunLoop start");
            whilewhile (!bExist)//until this component destroy
            {
                if (countInThread > 0)
                {
                    Debug.LogError("TestThreadRunLoop:countInThread=" + countInThread);
                    countInThread = 0;
                }
                if ((++countLoopThread) >= 10000)
                    countLoopThread = 0;
                Thread.Sleep(200);//sleep 0.2 second in the loop
            }
            Debug.LogError("TestThreadRunLoop end");
        }
        /// <summary>
        /// work in thread with no param
        /// </summary>
        void DoSomeWorkInThread(object obj)
        {
            //We transfer params by 'JSONData'.
            //You can storage your data in the JSONData
            JSONData jsonData = obj as JSONData;

            List<int> testParams = jsonData.GetObjectExtern("TestExternMsg") as List<int>;//test extern param
            Debug.Log("DoSomeWorkInThread testParams.Count=" + testParams.Count);
            //do some work (you can't accept the component of unity in this thread, such as 'PlayerPrefs', just same as in native C#.)
            Debug.Log("DoSomeWorkInThread(" + jsonData + ") start at " + DateTime.Now);
            long count = 0;
            int id = jsonData["id"];
            for (int i = 0; i < 10000; i++)
            {
                id += jsonData.Count;   //Don't direct use 'jsonData["id"] += jsonData.Count;',
                                        //because JSONData is not efficient in hot update script in this block which loop 10000 times.
                Interlocked.Increment(ref count);
            }
            jsonData["id"] = id;
            Debug.Log("DoSomeWorkInThread(" + jsonData + ") end at " + DateTime.Now);

            //callback to main thread for show the result
            HotUpdateManager.OnThreadDone(jsonData);//mark work done and send to main thread
        }
        /// <summary>
        /// refresh UI in main thread when thread done
        /// </summary>
        void OnDoSomeWorkInThreadDone(object obj)
        {
            JSONData jsonData = obj as JSONData;
            GetComponent<Text>("TestMessage").text = "OnDoSomeWorkInThreadDone:"+ jsonData["id"];
            Debug.Log("OnDoSomeWorkInThreadDone:" + jsonData["id"]);
        }
    }
}
```

macro and region

```
using UnityEngine;

namespace CSharpLike
{
    public partial class SampleCSharp : LikeBehaviour
    {
        /// <summary>
        /// Support unity build-in define symbols and user custom define symbols
        /// ("Edit"->"Project Settings"->"Player"->"Scripting Define Symbols").
        /// just same with unity,except 'UNITY_EDITOR/UNITY_EDITOR_WIN/UNITY_EDITOR_OSX/UNITY_EDITOR_LINUX'
        /// only work both in 'debug mode' and in editor.
        /// </summary>
        void TestMacroAndRegion()
        {
            Debug.LogError("Test TestMacroAndRegion: Not supported 'macro' and 'region' in FREE version. (Supported in full version). " +
            "Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
            // We provide the following solutions if you not update to full version:
            // macro: define a variable in hot update script.
            // region: don't use region (Your code look not so tidy in editor only).

            // test macro (simulate)
            // in FREE version
            if (Application.platform == RuntimePlatform.WebGLPlayer)
                Debug.Log("Test macro (simulate): is UNITY_WEBGL");
            else
                Debug.Log("Test macro (simulate): is not UNITY_WEBGL");
            //// in full version
//#if UNITY_WEBGL
//            Debug.Log("Test macro: is UNITY_WEBGL");
//#else
//            Debug.Log("Test macro: is not UNITY_WEBGL");
//#endif
        }
    }
}
```

enum

```
using System;
using UnityEngine;

namespace CSharpLike
```

```
{
    public partial class ExampleCSharp : LikeBehaviour
    {
        Debug.LogError("Test Enum: You can use enum but can't DEFINE enum in hot update script in FREE version. (Supported in full version). "+
        "Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
        // We provide the following solutions if you not update to full version:
        // enum: You can direct use integer number instead of define enum in hot update script.
        // In FREE version, you still can use enum which defined in not hot update script.

        Debug.Log("Test use the enum of the normal script: DayOfWeek = " + DayOfWeek.Saturday);//output Saturday
    }
}
```

parameter modifier

```
using UnityEngine;

namespace CSharpLike
{
    public partial class ExampleCSharp : LikeBehaviour
    {
        /// <summary>
        /// test modifier for params of function, include 'ref' 'out' 'in' 'params' keyword.
        /// </summary>
        void TestModifier()
        {
            Debug.LogError("Test Modifier:Not supported params modifier ('ref'/'out'/'in'/'params') in FREE version. (Supported in full version). "+
            "Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
            // In FREE version, you can't use 'ref' 'out' 'in' 'params' keyword in hot update,
            // but you still can call the function in not hot update script with those keywords.
            // We provide the following solutions if you not update to full version:

            // test ref
            Vector3 current = Vector3.zero;
            Vector3 target = Vector3.one;
            Vector3 currentVelocity = Vector3.zero;
            Debug.Log("TestModifier ref:before:current=" + current + ",target=" + target + ",currentVelocity=" + currentVelocity);

            //in FREE version:
            SampleHowToUseModifier.currentVelocity = currentVelocity;
            current = SampleHowToUseModifier.SmoothDamp(current, target, 0.5f);
            currentVelocity = SampleHowToUseModifier.currentVelocity;
            ////in full version:
            //current = Vector3.SmoothDamp(current, target, ref currentVelocity, 0.5f);

            Debug.Log("Test ref:after:current=" + current + ",target=" + target + ",currentVelocity=" + currentVelocity);

            // test out
            Dictionary<string, int> dicts = new Dictionary<string, int>();
            dicts.Add("test", 1);
            int iValue;
            //in FREE version:
            if (SampleHowToUseModifier.TryGetValue(dicts, "test"))
            {
                iValue = SampleHowToUseModifier.value;
                Debug.Log("TestModifier out:iValue=" + iValue);
            }
            ////in full version:
            //if (dicts.TryGetValue("test", out iValue))
            //    Debug.Log("TestModifier out:iValue=" + iValue);

            // test in
            iValue = 100;
            //in FREE version:
            SampleHowToUseModifier.TestModifierIn("test", iValue);//You can ignore 'in' keyword 'SampleHowToUseModifier.ModifierIn("test", iValue);'
            ////in full version:
            //SampleHowToUseModifier.ModifierIn("test", in iValue);//You can ignore 'in' keyword 'SampleHowToUseModifier.ModifierIn("test", iValue);'

            // test params
            //in FREE version:
            SampleHowToUseModifier.TestModifierParams("free version");
            SampleHowToUseModifier.TestModifierParams("free version", "test");
            SampleHowToUseModifier.TestModifierParams("free version", "test", 1);
            SampleHowToUseModifier.TestModifierParams("free version", "test", 1, 0.5f);
            ////in full version:
            //SampleHowToUseModifier.ModifierParams("free version");
            //SampleHowToUseModifier.ModifierParams("free version", "test");
            //SampleHowToUseModifier.ModifierParams("free version", "test", 1);
            //SampleHowToUseModifier.ModifierParams("free version", "test", 1, 0.5f);
        }
    }
}
```

not hot update script:

```
using UnityEngine;

namespace CSharpLike
{
    public class SampleHowToUseModifier
    {
        #region Modifier ref
        public static Vector3 currentVelocity = Vector3.zero;
        public static Vector3 SmoothDamp(Vector3 current, Vector3 target, float smoothTime)
        {
            return Vector3.SmoothDamp(current, target, ref currentVelocity, smoothTime);
        }
        #endregion
        #region Modifier out
        public static int value = 0;
        public static bool TryGetValue(Dictionary<string, int> dics, string key)
        {
            return dics.TryGetValue(key, out value);
        }
        #endregion
        #region Modifier in
        public static void TestModifierIn(string str, int iValue)
        {
            ModifierIn(str,in iValue);
        }
        public static void ModifierIn(string str, in int iValue)
        {
            Debug.Log("ModifierIn:"+str + iValue);
        }
        #endregion
        #region Modifier params
        public static void TestModifierParams(string str)
        {
            ModifierParams(str);
        }
        public static void TestModifierParams(string str, object v1)
        {
            ModifierParams(str, v1);
        }
        public static void TestModifierParams(string str, object v1, object v2)
        {
            ModifierParams(str, v1, v2);
        }
        public static void TestModifierParams(string str, object v1, object v2, object v3)
        {
            ModifierParams(str, v1, v2, v3);
        }
        public static void ModifierParams(string str, params object[] values)
        {
            string strTemp = "ModifierParams:" + str;
            foreach (var value in values)
                strTemp += "," + value;
            Debug.Log(strTemp);
        }
        #endregion
    }
}
```

default parameters and function overloading

```
using UnityEngine;

namespace CSharpLike
{
    public partial class ExampleCSharp : LikeBehaviour
    {
        /// <summary>
        /// test function overloading and function param default value
```

```
        /// </summary>
        void TestOverloadingAndDefaultValue()
        {
            Debug.LogError("Test OverloadingAndDefaultValue: Not supported function overloading and function param default value in FREE version. "+
            "(Supported in full version). Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
            // We provide the following solutions if you not update to full version:
            // function overloading: use different function name.
            // param default value: remove that default value.
        }
    }
}
```

keyword

```
using UnityEngine;//test using command
using System;
using Random = UnityEngine.Random;//test using alias
using System.Text;
using System.IO;

namespace CSharpLike
{
    public partial class ExampleCSharp : LikeBehaviour
    {
        /// <summary>
        /// test not classify keyword
        /// </summary>
        void TestKeyword()
        {
            Debug.LogError("Test Keyword:Not supported '$ sizeof unsafe #pragma #warning #error' in FREE version. (Supported in full version). "+
            "Strongly recommended update to full version C#Like: https://assetstore.unity.com/packages/slug/222256");
            // We provide the following solutions if you not update to full version:
            // '$""' keyword: using string.Format (Your code look not so tidy and easy in editor only).
            // '@""' keyword: direct use "" instead.
            // sizeof and unsafe: direct use integer number.
            // #pragma #warning #error: don't use it.

            //build-in data type,
            //Int32/UInt32/Int64/UInt64/Int16/UInt16/Char/Single/Double/Decimal/Boolean/SByte/Byte/String.
            //equal
            //int/uint/long/ulong/short/ushort/char/float/double/decimal/bool/sbyte/byte/string.
            Int32 i = 123;//equal to 'int i = 123;'
            Debug.Log("test Int32 i = " + i);//output 123
        }
    }
}
```

Contact me: csharplike@qq.com