Awake與Start的差別:

當Unity腳本被載入時，Awake與Start兩個函式都會被自動執行。

其中Awake函式會先執行。

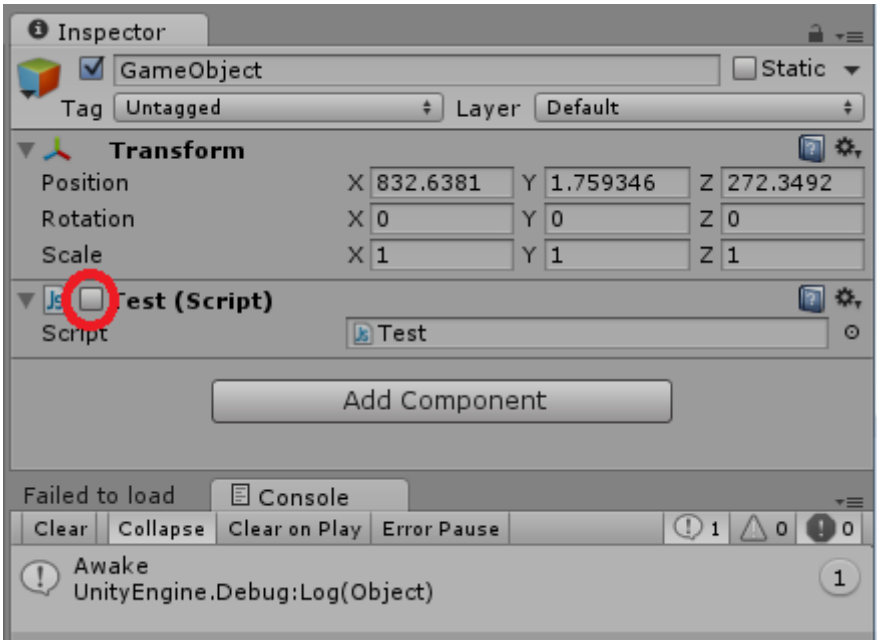
Awake函式有個特別的地方，那就是即便是腳本並沒有被致能(enable)，Awake函式依然會被執行。

試試看這個腳本:

2

```
#pragma strict
function Awake () {
Debug.Log("Awake");
}
function Start () {
Debug.Log("Start");
}
```

很明顯即便是沒有enable 腳本，Awake函式依然執行了。



Awake的特性讓他很適合用來設定任何腳本間的參考(也就是說在腳本中參考到別的腳本元件)，以及程式的初始化。

Start函式是在Awake之後呼叫，但會在第一個Update呼叫前呼叫。

換句話說，Start()函式呼叫後，會緊接著呼叫第一個Update函式。

所謂的"緊接著"的意思是針對該腳本而言，

事實上當start()執行完後，可能會先執行其他腳本的程式碼，然後再回來執行Update函式。

當然，Start()函式只有在腳本被enable時才會執行。

這可以讓你用來處理當腳本enble時要處理的事情，譬如說在真的需要時才執行某些部分的初始化。

要注意的是，Awake與Start在被貼到物件的生命週期中，只會被呼叫一次，而且不能藉由disable/enable腳本來重新呼叫。

較複雜的執行順序:

當在腳本A中參考到腳本B時，腳本B的Awake函式會被立刻呼叫並且執行，

但是Start函式並不會。Start函式會處在一種待執行的狀態，等它前面的該執行的程式執行完後才會執行。
舉個例子來說，假設場景中只有A腳本。
在A腳本的Start中參考到B腳本，此時會立刻執行B腳本的Awake函式。
B腳本的Awake函式執行完後回來繼續執行A腳本的Start。
A腳本的Start執行完後才會執行B腳本的Start。
要注意的是，如果你把B腳本的一些變數或是函式的初始化放到Start中，
但是在A腳本參考B腳本後立刻讀取或寫入這些變數或函式就可能會導致錯誤。
因為B腳本的Start函式根本還沒執行，所以這些變數完全處在未初始化的階段!
因此產生了錯誤。
下面兩個腳本用來驗證這些順序，將腳本A拖到場景中的空物件即可：
腳本First.js：

```
#pragma strict
function Awake () {
Debug.Log("First Awake -start");
Debug.Log("First Awake -end");
}

function Start () {
Debug.Log("First Start-start");
Debug.Log("before test");
test();
Debug.Log("after test");
Debug.Log("First Start-end");
}

function Update () {
Debug.Log("First Update-start");

Debug.Log("First Update-end");
}

function test(){

Debug.Log("test Start -1");
var x:GameObject=new GameObject("x");
x.AddComponent("second");
Debug.Log("test Start -2");

}
```

腳本second.js：

```
#pragma strict

function Awake () {
Debug.Log("Second Awake");
}
function Start () {
Debug.Log("Second Start");
}

```

```
10        function Update () {
11        Debug.Log("Second Update");
12        }
```

輸出: