1.

对象，代表序列化数据的集合，表示某个资源的具体实例。它可以是Unity引擎使用的任何类型的资源，例如网格，Sprite，音频剪辑或动画剪辑。所有的对象（Object）都是UnityEngine.Object基类的子类。游戏对象是Unity最重要的对象。

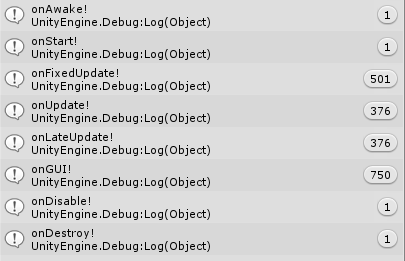
资源（Asset）是硬盘中的文件，存储在Unity工程的Assets文件夹内。例如，纹理（Texture），材质（Material）和FBX文件等，它们都是资源。

一个资源文件可能会包括多个Object。有些资源作为模板，可实例化成游戏中具体的对象。

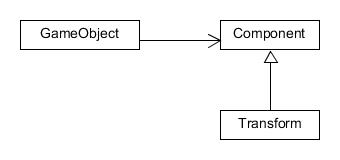
2.

初始化之后执行onAwake和onStart，仅执行一次。然后是update和ONGUI的循环，每一桢都被调用一次。按终止后销毁对象，仅执行一次。代码如下：

using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class NewBehaviourScript : MonoBehaviour {  
  
    void Awake()  
    {  
        Debug.Log("onAwake!");  
    }  
    void Start()  
    {  
        Debug.Log("onStart!");  
    }  
    void Update()  
    {  
        Debug.Log("onUpdate!");  
    }  
    void OnGUI()  
    {  
        Debug.Log("onGUI!");  
    }  
    void FixedUpdate()  
    {  
        Debug.Log("onFixedUpdate!");  
    }  
    void LateUpdate()  
    {  
        Debug.Log("onLateUpdate!");  
    }  
    void Reset()  
    {  
        Debug.Log("onReset!");  
    }  
    void OnDisable()  
    {  
        Debug.Log("onDisable!");  
    }  
    void OnDestroy()  
    {  
        Debug.Log("onDestroy!");  
    }  
}



3. transform是每个GameObject都有的设置位置,大小，角度的组件(Component)



4.

查找对象：

1）通过对象名称（Find方法）

static GameObject **Find** (string **name**)

2）通过标签获取

**FindWithTag方法：**

static GameObject **FindWithTag** (string **tag**)

**FindGameObjectsWithTag方法：**

static GameObject[] **FindGameObjectsWithTag** (**string tag**)

3）通过类型获取

**FindObjectOfType方法：**

static Object FindObjectOfType(Type type)

**FindObjectsOfType方法：**

static Object FindObjectsOfType(Type type)

添加子对象：

GameObject.CreatePrimitive(PrimitiveType type)

遍历对象树：

foreach (Transform child in transform){

child.position += Vector3.up //或者修改child的其它属性

}

清除所有子对象

foreach (Transform child in transform){

Destroy(child.gameObject);

}

5.

预设的好处：

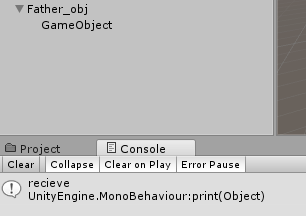
预设，即把制作好了游戏对象（场景中的任意一个gameobject绑定好了相应的组件），将它制作成一个组件模版，用于批量的套用工作，例如说场景中本质上“重复”的东西。有这类需求的时候将会大大节省时间。

与对象克隆的关系：

默认生成的prefab其实和模版是一模一样的，就像是克隆体，但生成的位置和角度以及生成后的一些属性是允许发生变化的。而克隆的话是会随克隆本体的变化而变化。

6.组合模式

组合模式，又称之为“部分-整体”模式，属于对象结构型模式。组合模式可以让客户端可以统一的对待单个对象和组合对象。cast信息有点类似C#中的事件委托机制。



Father\_obj:

void Start () {  
        this.BroadcastMessage("f");  
    }

Gameobject:

void f(){  
        print ("recieve");  
    }