

Unity Scripting

Fundamentals

1. Unity Scripting
2. Scripting Languages in Unity
3. Creating Script Assets
4. Default Script Structure
5. 명령어 (Command)
6. 변수 (Variable)
7. DataType
8. Naming Rule
9. Attaching Scripts to Objects
10. public vs. private
11. Debug Utility Function
12. 함수 (Function, Method)
13. 제어문
14. 스크립트 간의 호출

Unity Scripting

- Scripting in Unity is the programming side of game development
- Unity primarily uses the C# language
- C# is very similar to Java, and is ideal for game development because it is very object-oriented

After all, everything we want to interact with is a GameObject

Much easier to write code, if we can think in terms of objects

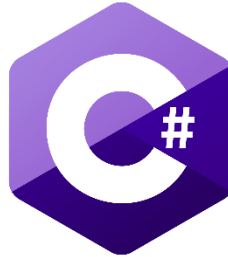
- Unity Scripting is primarily interacting with GameObject components
- Scripts are really just custom components

When you create a script, you are creating your own component. You can give the component behavior, properties, fields, and values

- You add scripts to GameObject just like any other component

Supported Languages in Unity

- 3 Languages are supported

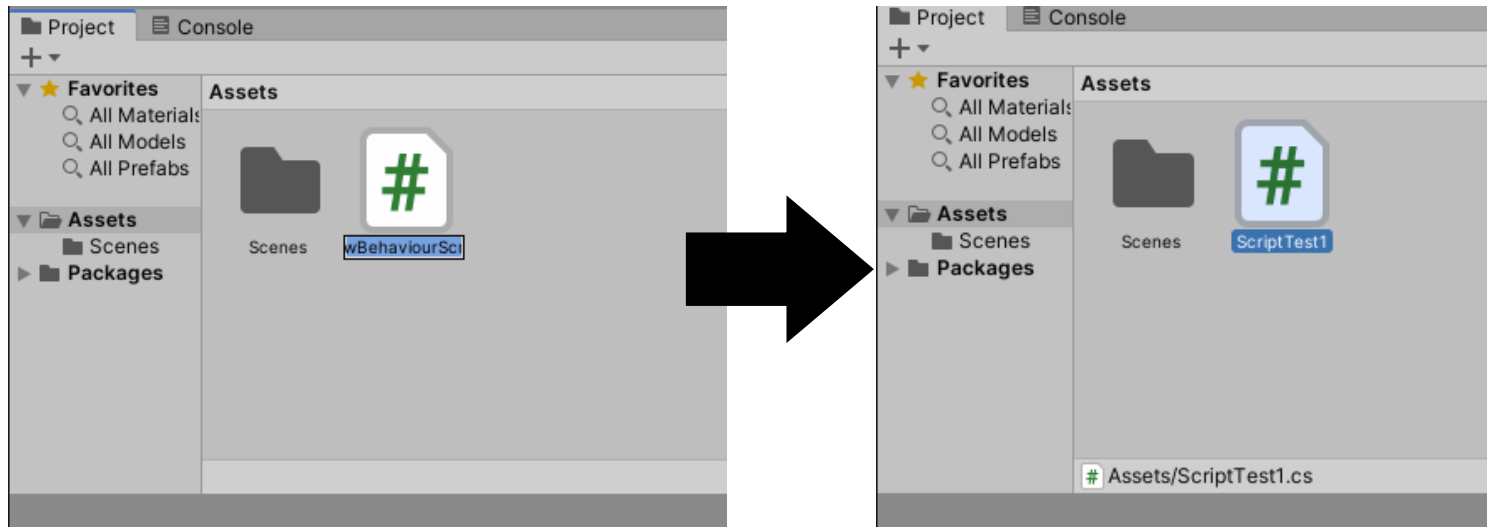
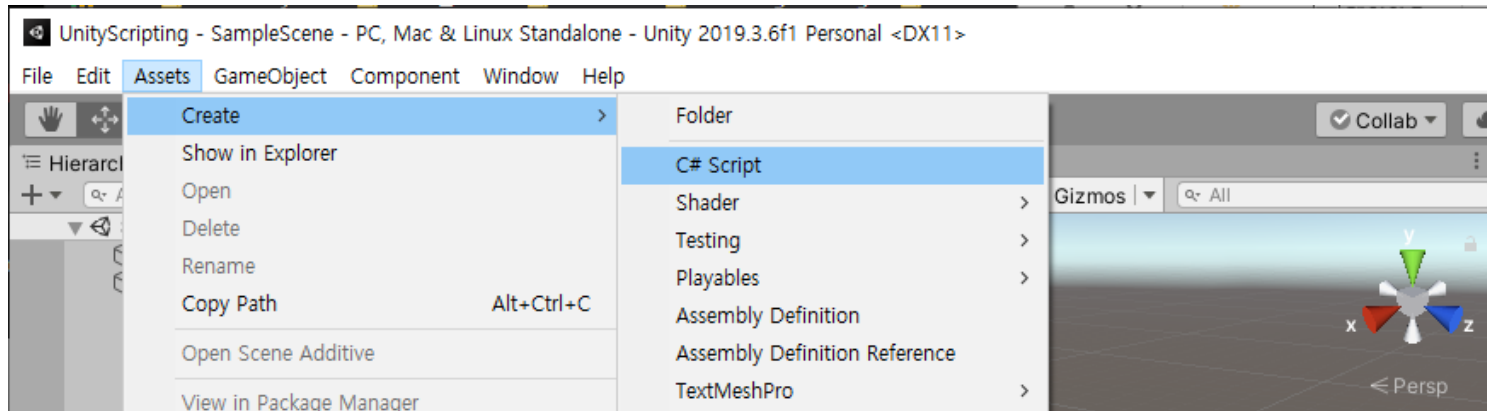


- Unity documentation provides examples for each type.
- Javascript is often used in online Unity examples, but C# is now becoming **first choice** in current Unity tutorials .
- Module using C# to align more with other modules in programs and commercial activities.

Creating Script Assets

- Create a new script
 - From the **Assets** Menu, choose **Create**
 - Choose the language you want from the menu list (click **C# Script**)
 - The default name newly created scripts is **NewBehaviorScript**
 - **Rename the script** to give a name that conveys the scripts functionality

Creating Script Assets



Default Script Structure

- Javascript Example

```
#pragma strict  
  
function Start () {  
}  
  
function Update () {  
}
```

Default Script Structure

- C# Example

```
using UnityEngine;
using System.Collections;

public class NewBehaviourScript : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }

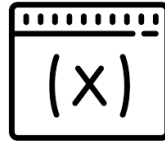
}
```


명령어

- Command
 - 스크립트에 사용된 명령
 - 세미콜론(;)으로 끝남

```
// use this for initialization  
void Start() {  
    int speed = 5;  
}
```

변수



- 변수 (Variable)
 - 정보를 담는 상자 (메모리 공간)
 - “변하는 수” : 프로그램 수행 과정에서 값이 수시로 변경될 수 있음

(1) 변수 선언

```
DataType variableName;
```

```
int speed;  
string nameOfBox;  
bool isFound;
```

- DataType : 저장하려는 정보의 종류를 명시



Simple Types in C#

- Data types and Variables in C#

`int, float, bool, char, string ..`

- With MonoBehaviour, use all components as types in Unity

```
public class MyFirstScript : MonoBehaviour {  
    GameObject myObj;  
    Camera myMainCamera;  
    ShpereCollider mySphereCollider;  
    MyFirstScript myScript;  
}
```



Type Declaration in C#

- Declaring data type

```
datatype variableName;
```

```
int anInteger;
```

```
// a float type stores decimal values
```

```
float aFloatValue;
```

```
String aName;
```

– 변수 명명 규칙(Naming Rule)

- Keyword 사용 금지 (예, transform)
- 알파벳으로 시작 (숫자로 시작하면 안됨)
- 카멜표기법(camelCase)을 권장
 - 변수 이름은 소문자로 시작, 클래스 이름은 대문자로 시작
 - 변수 이름을 보고 직관적으로 이해할 수 있도록
 - 2개 이상의 단어를 결합시킬 때는 단어 사이에 대문자로 구분
 - In general, don't abbreviate names of things. Spell them out, even if they're long:

Code	Commentary
<code>destinationSelection</code>	Good.
<code>destSel</code>	Not clear.
<code>setBackgroundColor:</code>	Good.
<code>setBkgdColor:</code>	Not clear.



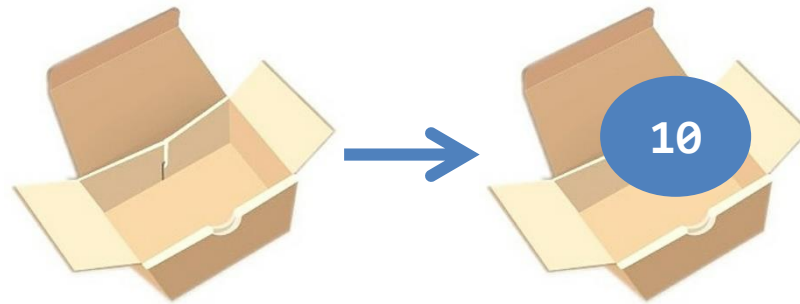
(2) 변수에 값 지정(할당)

– 대입연산자(=) 사용

```
int speed;  
string nameOfBox;  
bool isFound;  
  
void Start() {  
    speed = 100;  
    nameOfBox = "chocolate box";  
    isFound = true;  
}
```

(예)
int x; // 변수 선언
x = 10; // 값 지정

변수 = 값 (또는 수식) ;
↖ ↗



초기상태

대입 연산이 수행되어
값이 지정된 상태

Variable Assignment in C#

- Assigning the value into variables

```
anInteger = 20;  
// a float type stores decimal values  
aFloatValue = 20.0f;  
aName = "David Smith";  
// String concatenation  
"My name is" + aName;  
// or use System.Concat(string, string) method
```

– 변수 선언과 초기값 지정

```
int speed = 200;  
string nameOfBox = "candy box" ;  
bool isFound = false;
```

– 변수 값 수정

```
int speed = 200;  
string nameOfBox = "candy box";  
bool isFound = false;  
  
void Start() {  
    speed = 300;  
    nameOfBox = "chip box";  
    isFound = true;  
  
    ...  
  
    speed = speed + 200;  
}
```


Attaching Scripts to Objects

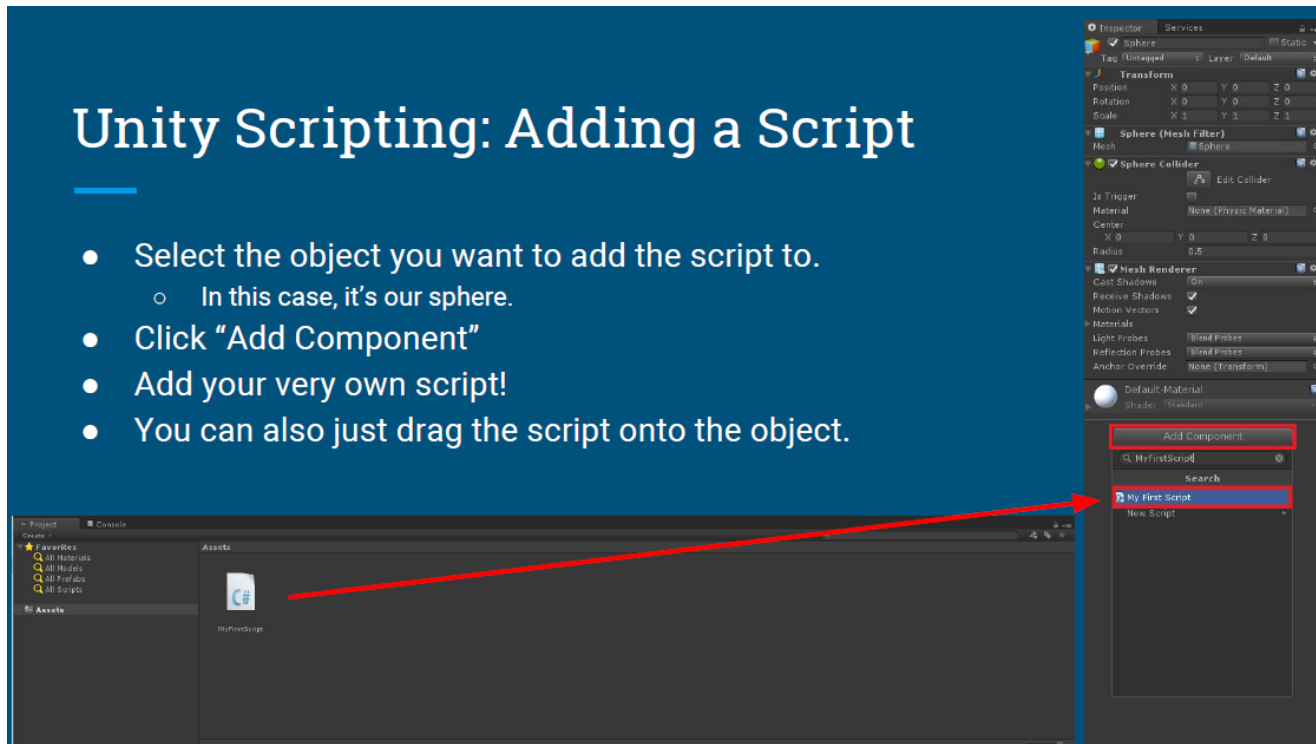
- Attaching the script
 - Create the new script
 - Name the script accordingly
 - Drag the script from the Assets Window onto the object you want the script to affect.

OR select Add Component from the Inspector Window and navigate to the Scripts Panel.
 - Script behaviors can be switched on or off via the Component Panel checkbox.



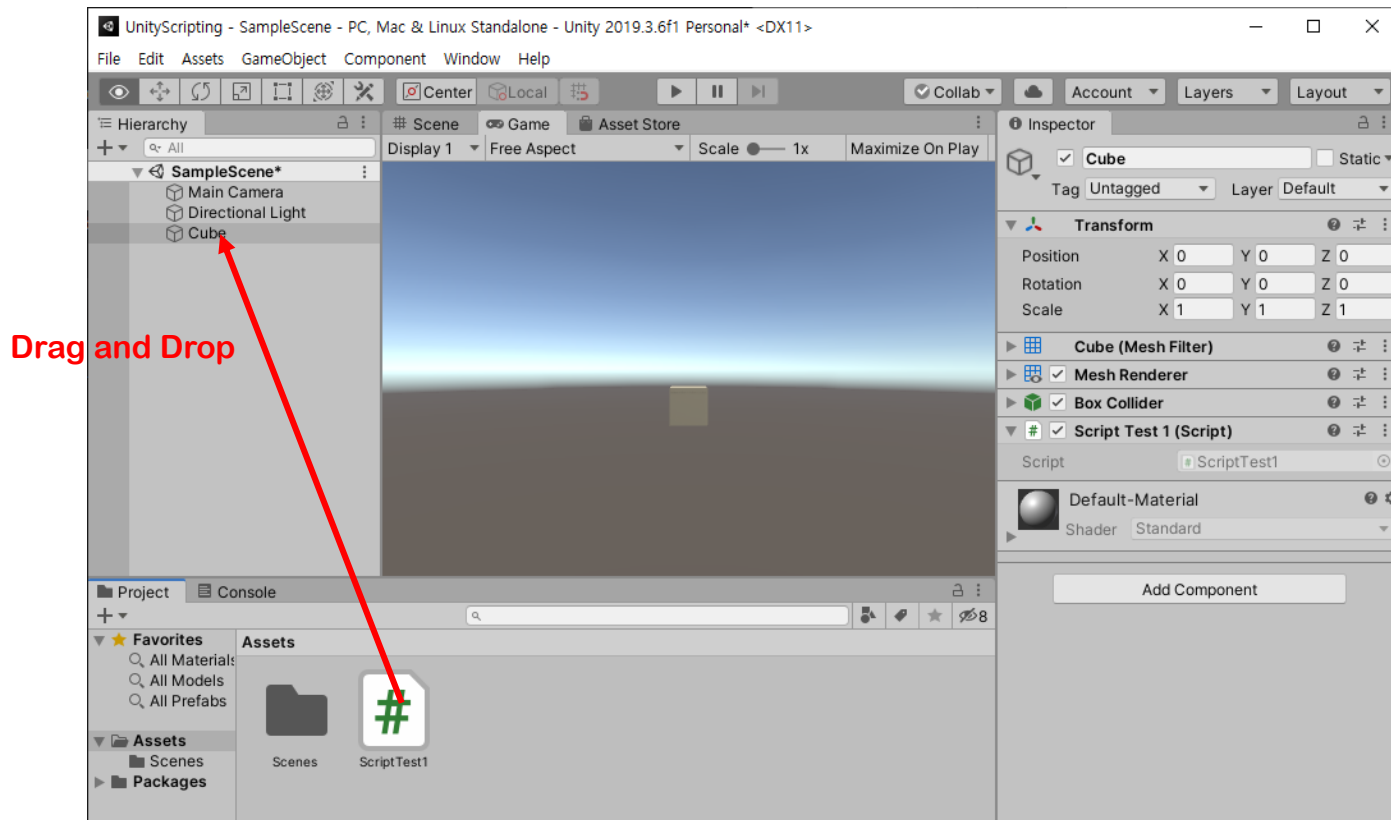
Unity Scripting: Adding a Script

- Select the object you want to add the script to.
 - In this case, it's our sphere.
- Click “Add Component”
- Add your very own script!
- You can also just drag the script onto the object.

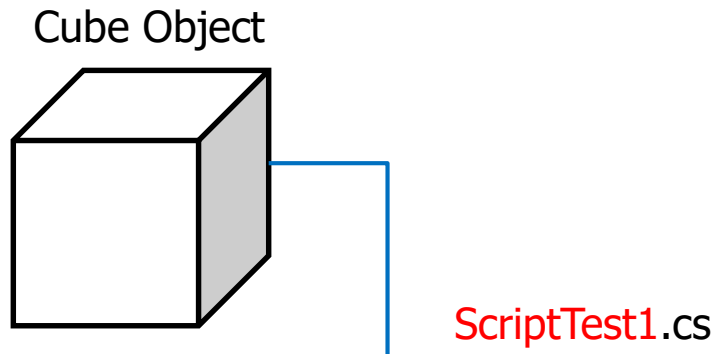


[출처] University of California San Diego

- Attaching the Script to Object
 - 스트립트를 (연결하고자 하는) 객체로 드래그앤드롭



- Changing an Object's Properties via Scripts



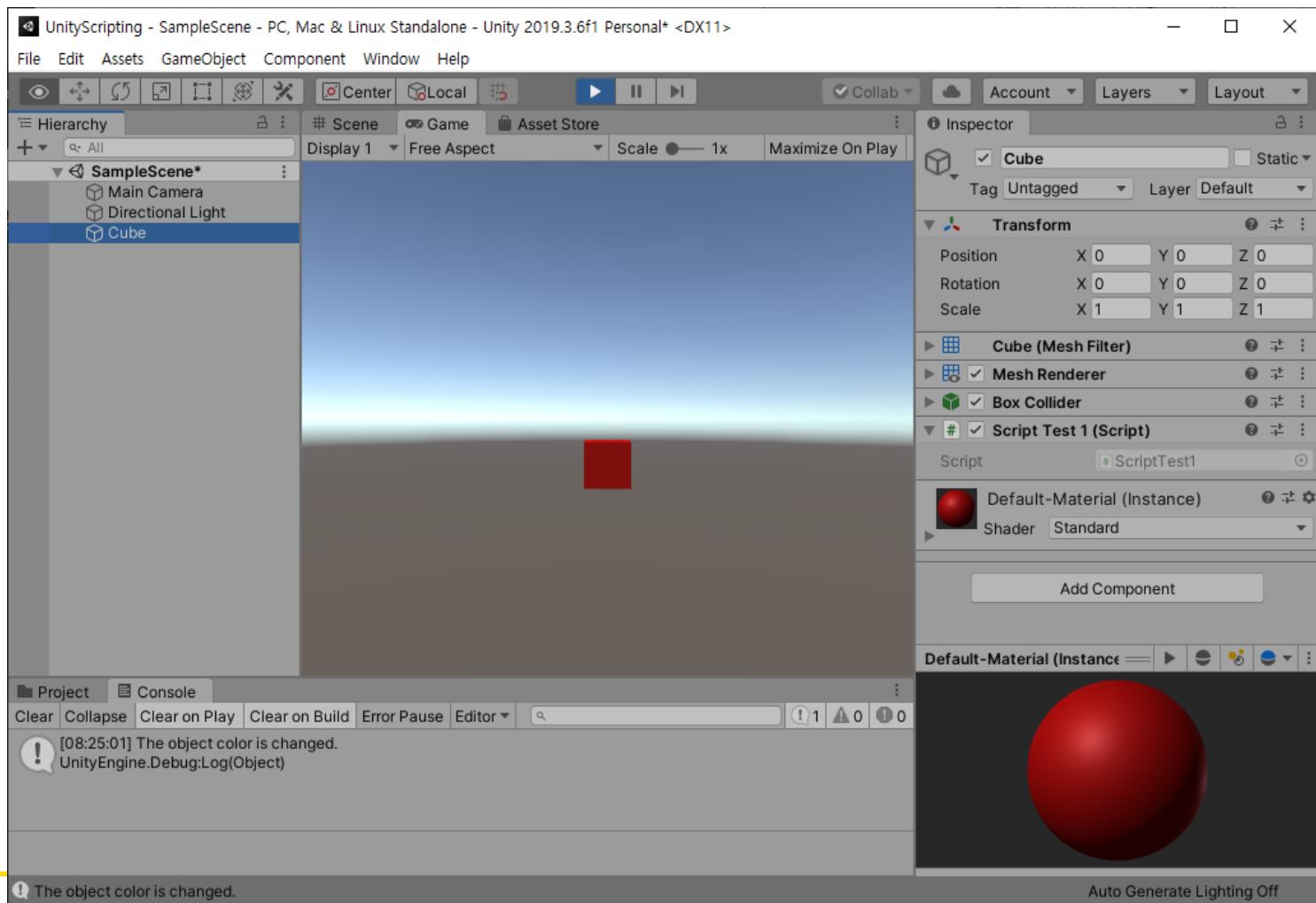
```
using UnityEngine;
using System.Collections;

public class ScriptTest1 : MonoBehaviour {
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if(Input.GetKeyDown(KeyCode.R)){
            gameObject.GetComponent<Renderer>().material.color = Color.red;
            Debug.Log("The object color is changed.");
        }
    }
}
```

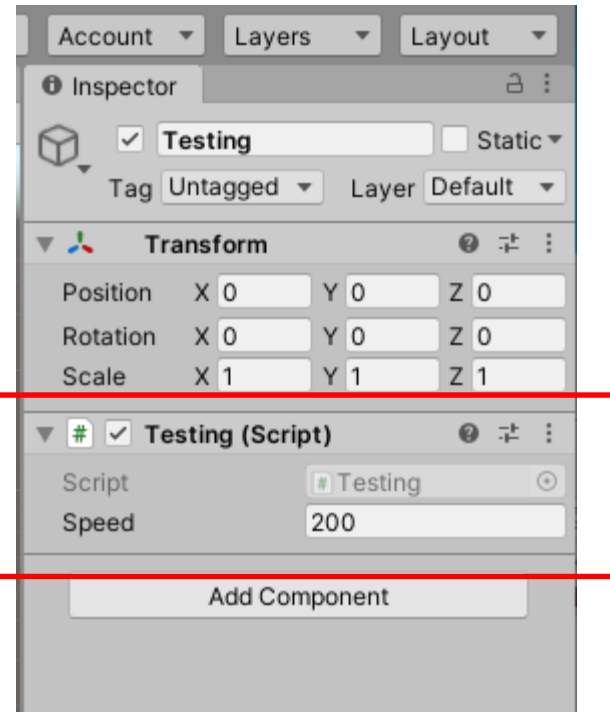
- Play and press the key "R"



- public vs. private

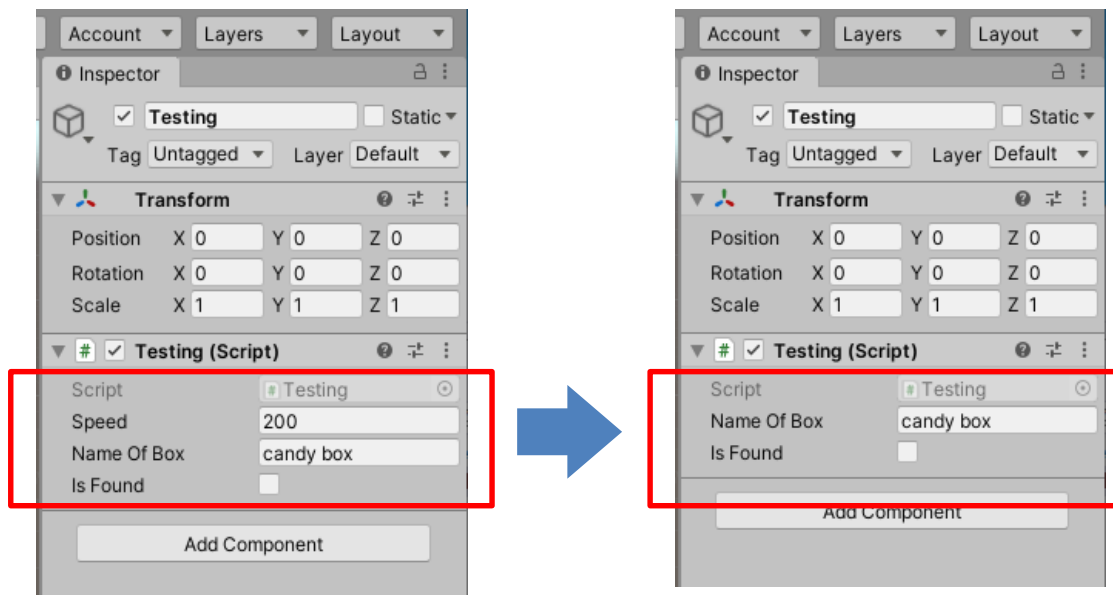
```
public int speed = 100;  
private string nameOfBox = "candy box";  
private bool isFound = false;
```

- public 변수는 인스펙터에 보여짐



– public 변수 감추기

```
[System.NonSerialized]  
public int speed = 200;  
public string nameOfBox = "candy box";  
public bool isFound = false;
```



– private 변수 나타내기

```
[SerializeField] private int power = 100;  
private int age = 20;
```



유니티 실습

Debug Utility Function

- Very useful debug and environment events utility function

```
Debug.Log("Message you to want to send to the Console Window");  
Debug.Log("Hello from the Red Cube - my new color is now:" + aColor);
```

```
// Update is called once per frame  
void Update() {  
    if (Input.GetMouseButtonDown(0)) {  
        Debug.Log("You pressed the left mouse button.");  
    }  
    if (Input.GetKeyDown("a")) {  
        Debug.Log("U pressed A key");  
    }  
}
```

함수

$f(x)$

- 함수(Function) == 메소드(Method)
 - 게임 런타임(실행도중)의 특정 시점에서 호출할 수 있는 명령 또는 명령들의 집합

```
accessPerm returnType methodName(para1, para2 ..) {  
    명령어들 ...  
}
```

- 함수 유형
 1. 사용자 정의 함수
 2. 유니티 기본 내장 함수

```
#include <stdio.h>

int hello(void) {
    printf("Hello");
}
int main(void) {
    hello();
}
```

C:\Users\tera\Desktop\테스트2.exe

Hello

– 매개변수(Parameter)

- 함수 내부에서 사용되는 변수
- 함수를 호출할 때, 함수에 보내지는 정보를 담는다

함수 호출

```
int result = callMethod3(100);
```

```
callMethod4(10, 20);
```

```
...
```

함수 선언

```
void callMethod4(int x, int y) {  
    Debug.Log(x+y);  
}
```

10과 20을 더한 값이
num1에 저장됨 ②

```
int add(int a, int b)  
{  
    return a + b;  
}
```

```
int main()  
{  
    int num1;
```

```
num1 = add(10, 20); ① add 함수 호출  
10과 20 전달
```

```
printf("%d\n", num1);
```

```
return 0;
```

```
}
```

– 사용자 정의 함수



```
void Start() {  
    callMethod1();  
    int num = callMethod2();  
    int res = callMethod3(100);  
    Debug.Log("함수호출=" + num + " 다음함수호출=" + res);  
}
```

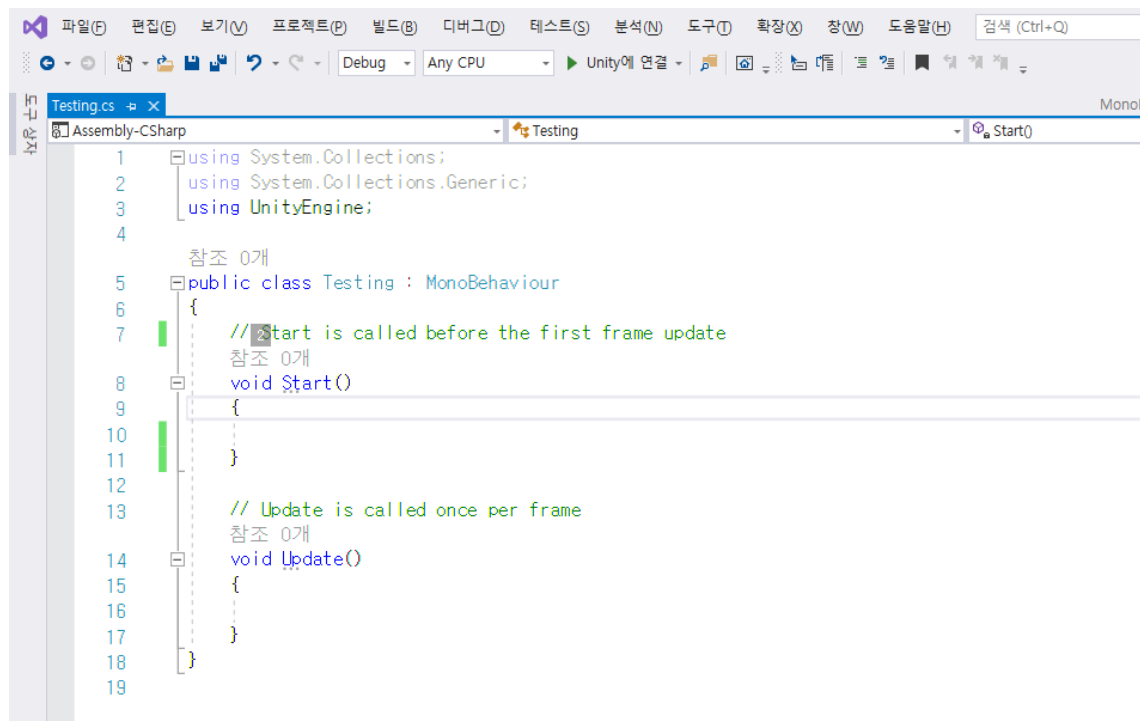
```
void callMethod1() {  
    Debug.Log("callMethod1 is called.");  
}
```

```
int callMethod2() {  
    Debug.Log("callMethod2 is called.");  
    int n1 = 100;  
    int n2 = 200;  
  
    return n1+n2;  
}
```

```
int callMethod3(int a) {  
    return a+1;  
}
```

– 유니티 기본 내장 함수

(1) On~ 이 붙지 않은 함수와 (2) On~ 이 붙어 있는 함수



– 유니티 기본 내장 함수

- Start()

- 스크립트가 동작하는 동안 단 한번만 호출되는 유니티 기본 함수
- 초기화 코드 실행에 활용

- Update()

- 게임의 매 프레임이 랜더링될 때마다 호출되는 유니티 기본 함수
(예, 초당 30 FPS인 게임에서는 1초에 Update() 함수가 30번 호출됨)
- 지속적으로 처리되어야 하는 커맨드, 실시간으로 처리되는 게임 내의 변화를 처리할 때 사용

- On~ 계열 함수

```
void OnMouseDown() {  
    Debug.Log("Mouse click");  
}
```

// 마우스로 오브젝트를 클릭하고 떴을 순간

```
void OnMouseUp() {  
    Debug.Log("Mouse UP");  
}
```

// 마우스가 오브젝트를 클릭했을 때
// (마우스를 눌렀다가 떴을 때)

```
void OnMouseEnter() {  
    Debug.Log("Mouse Enter");  
}
```

// 마우스가 오브젝트에 들어왔을 때

```
void OnMouseExit() {  
    Debug.Log("Mouse out");  
}
```

// 마우스가 오브젝트 위에 머물다가 빠져 나왔을 때



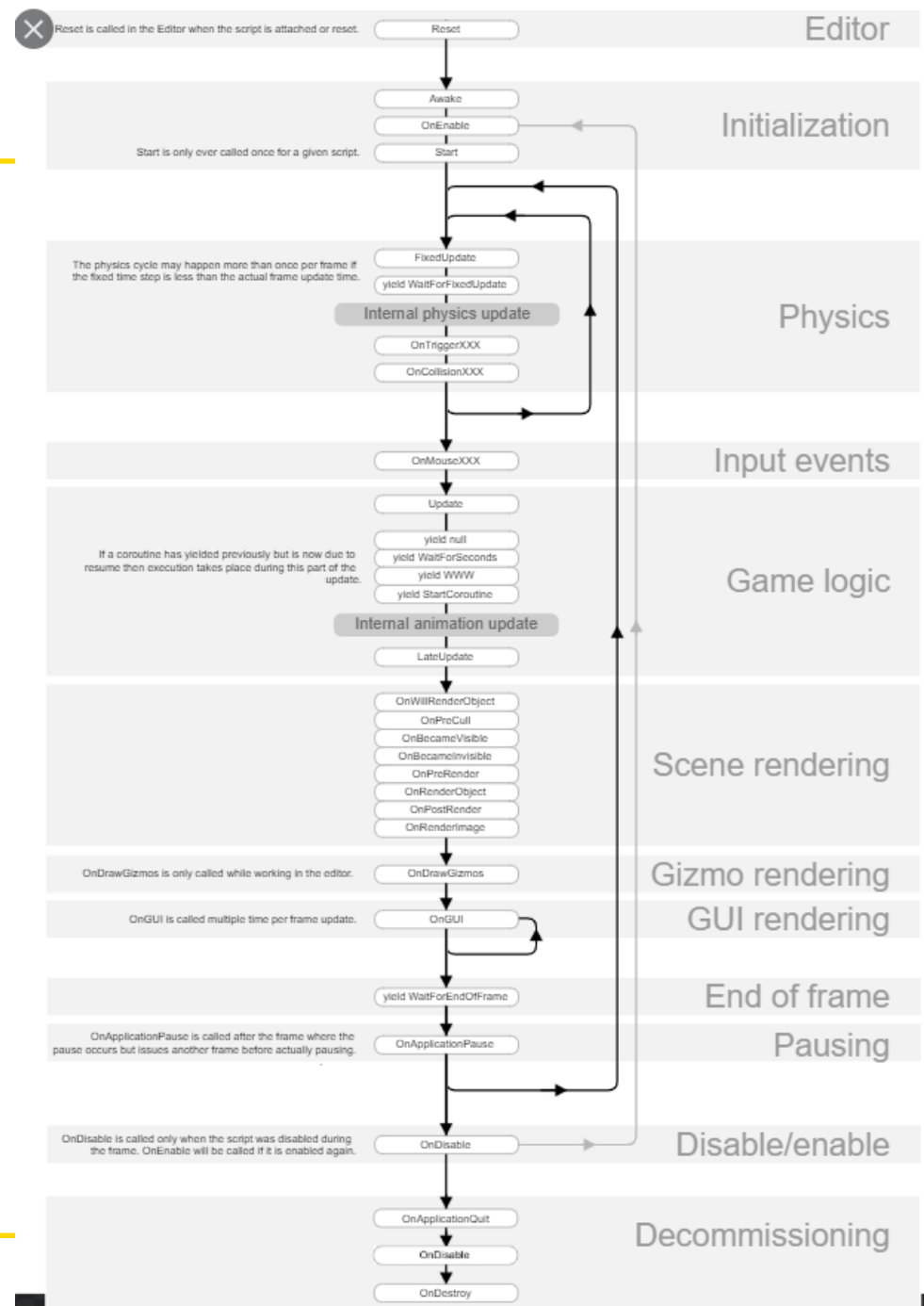

```
참조 0개
5 public class Testing : MonoBehaviour
6 {
```

– 미리 정의되어 있는 함수들 ?

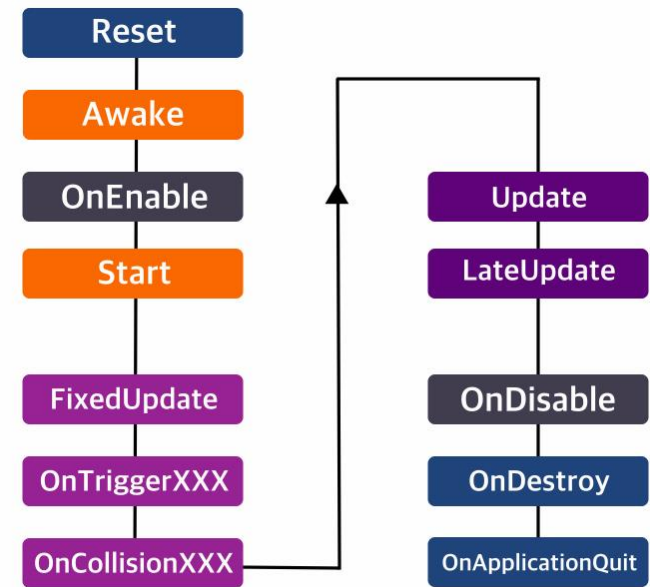
- “Testing” 클래스는 “MonoBehaviour” 클래스로 부터 상속받아 만들어짐
- MonoBehaviour이 가지는 메소드를 확인

The screenshot shows the Unity Scripting API documentation for the **MonoBehaviour** class. The page is titled "MonoBehaviour" and indicates it is a class in UnityEngine that inherits from Behaviour and is implemented in UnityEngine.CoreModule. The "Description" section explains that MonoBehaviour is the base class for all Unity scripts and that users must explicitly derive from it when using C#. It also notes that the class does not support the null-conditional operator (?) or the null-coalescing operator (??). A "Note" section mentions a checkbox in the Unity Editor for enabling or disabling MonoBehaviour. The "Methods" section lists **Start()** and **Update()** as key methods.

- Order of Execution for Event Functions



- Awake()
 - 스크립트가 비활성화 되어도 실행
 - 주로 게임의 상태 값 또는 변수 초기화에 사용
 - 1번만 실행, Start() 함수 실행 전에 실행
 - 코루틴(Couroutine) 사용이 불가능
- Start()
 - 1번만 실행, Update() 함수 실행 전에 실행
 - 스크립트가 활성화 되어야 실행
 - 코루틴(Couroutine) 사용 가능
- Update()
 - 매 프레임마다 호출
 - 정기적인 변경, 간단한 타이머, 입력 값 탐지, 카메라 이동 로직에 사용
 - 시간 간격이 동일하지 않음 (이전 프레임에서 오래 걸리면 다음 프레임에 딜레이가 발생)
- FixedUpdate()
 - 규칙적인 시간 간격으로 호출 (호출 사이의 시간 간격이 동일)
 - 리지드바디 등 Physics 오브젝트에 영향을 주는 것은 FixedUpdate() 사용을 권장
- OnEnable()
 - 스크립트, 게임 오브젝트가 비활성화 → 활성화 할 때마다 호출
 - 이벤트 연경을 종료할 때 주로 사용
 - 코루틴 사용이 불가능
- OnGUI()
 - Legacy GUI 관련 함수 사용



<http://itmining.tistory.com>

1 Minute Quiz



새로운 스크립트를 “Sample” 이라는 만들었습니다. 편집기에서 스크립트를 얼마간 작성을 하고 난 이후, 스크립트의 이름을 “Test”로 수정하였습니다. 어떤 오류가 있을까요 ?

- 해당 오류를 고치려면 어떻게 고칠 수 있을까요?

1 Minute Quiz



아래의 스크립트를 보고, 어떤 오류가 나오는지 유니티에서 확인하세요

```
public class ErrorCheck : MonoBehaviour
{
    int CalculateSum(int x, int y)
    {
        return x + y;
    }

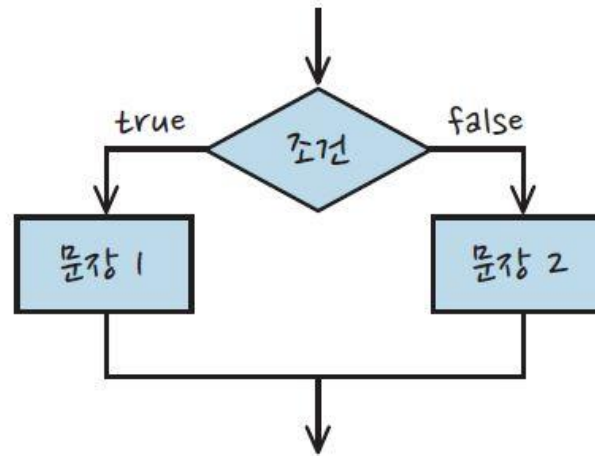
    // Start is called before the first frame update
    void Start()
    {
        string retValue = CalculateSum(100, 200);
    }

    // Update is called once per frame
    void Update() {}
}
```

제어문

- 조건문 if () else

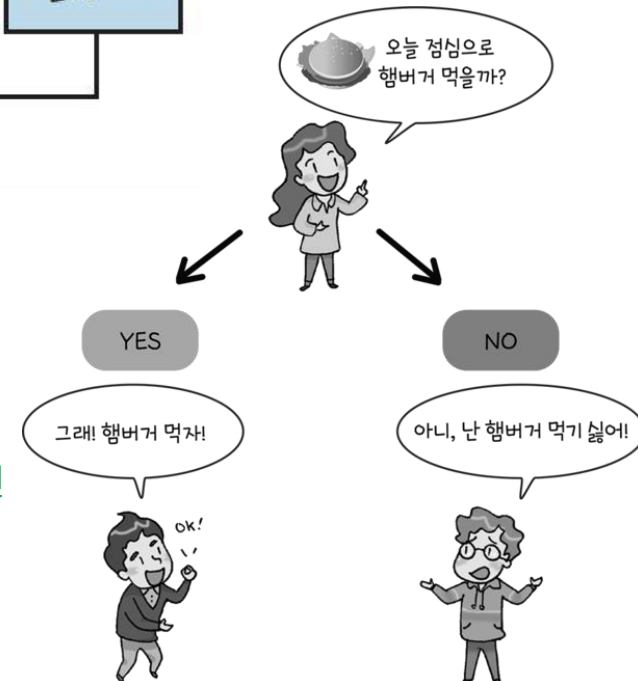
```
if ([불 표현식])  
{  
    // 불 표현식이 참일 때 실행할 문장  
}  
else  
{  
    // 불 표현식이 거짓일 때 실행할 문장  
}
```



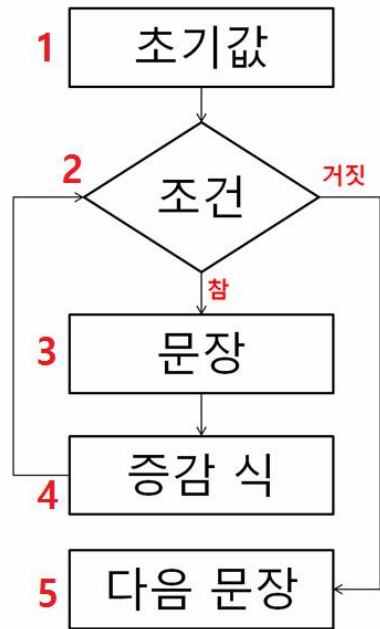
- 다중 조건

if (A && B) // A가 참이면서 동시에 B도 참이라면

if (A || B) // A 또는 B 둘 중에 하나 이상이 참이라면



- 반복문 for()



<for문 순서도>

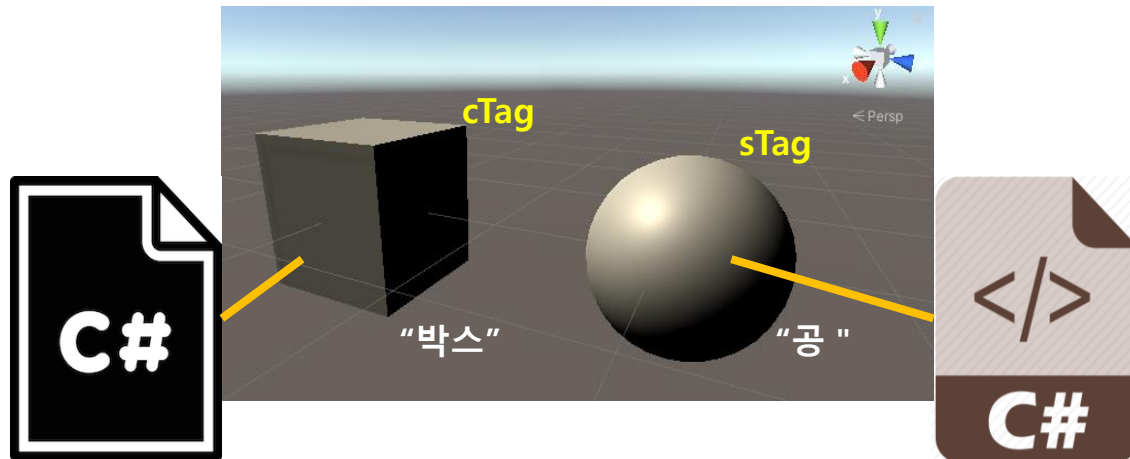
```

1      2      4
for(초기값; 조건식; 증감식)
{
    문장; 3
}
다음 문장; 5
  
```

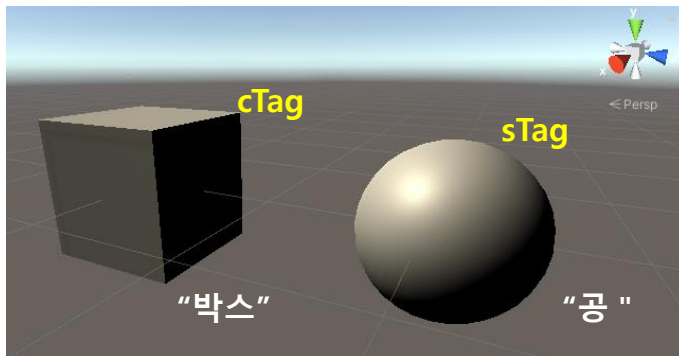
<문법>

스크립트 간의 호출

- 다른 오브젝트에 연결된 스크립트를 실행
 - “박스”에 연결된 스크립트에서 “공”에 연결된 스크립트의 함수를 호출
 - 대상 오브젝트를 찾음
 - 해당 오브젝트에 연결된 스크립트 함수를 호출



- 다른 오브젝트로 접근
 - GameObject.Find() : 이름으로 찾기
 - GameObject.FindWithTag() : 태그로 찾기



```
void Start() {  
    GameObject targetObj = GameObject.Find("박스");  
    Debug.Log(targetObj.transform.position.x);  
  
    GameObject targetObj2 = GameObject.FindWithTag("sTag");  
    Debug.Log(targetObj2.transform.position.x);  
}
```

1 Minute Quiz



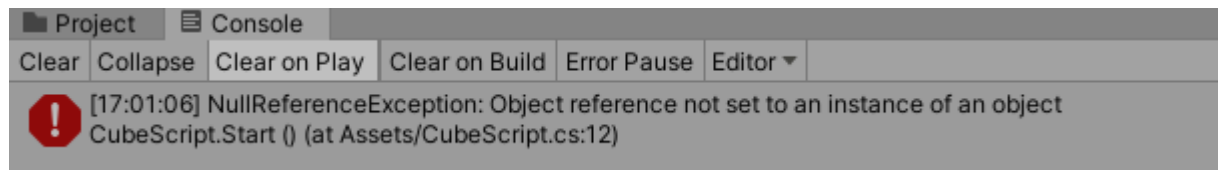
“Cube”에 연결된 “CubeScript”에서 “Sphere”를 찾기 위해 Find() 함수를 사용하였다. 만약에 씬에 있는 오브젝트를 찾지 못할 경우, 어떤 결과 값이 반환되는지 확인하세요.

- Find() 함수를 사용하여 씬에 있는 오브젝트를 찾는데 만약 동일한 이름으로 2개 이상의 오브젝트가 존재한다면, 어떤 결과가 나오는지 확인하세요.

1 Minute Quiz




“Cube”에 연결된 “CubeScript”에서 “Sphere”를 찾기 위해 Find() 함수를 사용하였다. 만약에 씬에 있는 오브젝트를 찾지 못할 경우, 어떤 결과 값이 반환되는지 확인하세요.



```
void Start() {  
    GameObject tarObj = GameObject.Find("test");  
    if (tarObj == null) {  
        Debug.Log("오브젝트를 찾지 못했습니다");  
    } else {  
        Debug.Log(tarObj.transform.position.x);  
    }  
}
```

- Find() 함수는 활성화된 오브젝트만 찾아 반환


unity | DOCUMENTATION

[Manual](#)
[Scripting API](#)

Version: 2019.3

Scripting API

- UnityEngine
- UnityEditor
- Unity
- Other

GameObject.Find

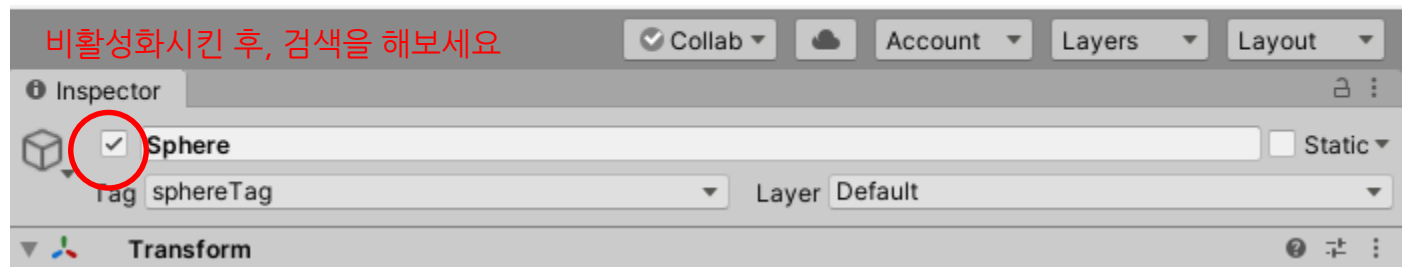
SWITCH TO MANUAL

```
public static GameObject Find(string name);
```

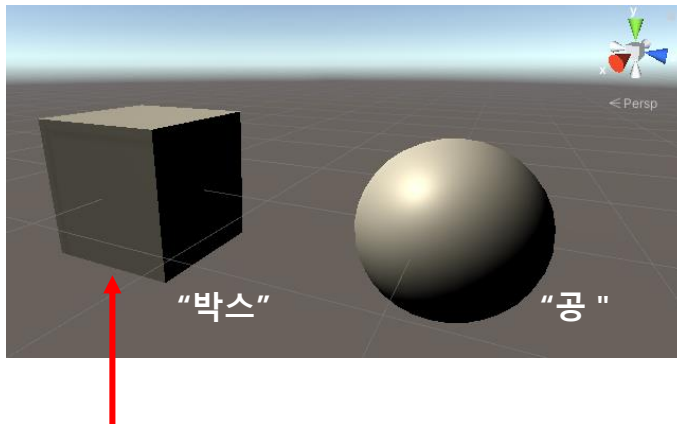
Description

Finds a GameObject by `name` and returns it.

This function only returns active GameObjects. If no GameObject with `name` can be found, null is returned. I



- 다른 오브젝트에 있는 명령 실행
 - Find() 등을 이용하여 다른 오브젝트를 찾음
 - SendMessage(함수명, 매개변수)를 실행



```
void WhoAmI(string parName) {  
    Debug.Log("My name is" + parName);  
}  
void WhoAreYou() {  
    Debug.Log("My name is" + gameObject.name);  
}
```

```
void Start() {  
    GameObject.Find("공").SendMessage("WhoAmI", "YongHwan");  
    GameObject.Find("공").SendMessage("WhoAreYou");  
}
```

– GetComponent<>() 함수

- 같은 게임오브젝트에서 컴포넌트를 호출
 - GetComponent<컴포넌트이름>() 으로 컴포넌트를 가져옴
- 다른 게임오브젝트가 갖고 있는 컴포넌트를 호출
 - Find() 등을 이용하여 다른 게임오브젝트를 찾음
 - GetComponent<컴포넌트이름>() 으로 컴포넌트를 가져옴
 - 컴포넌트에 들어있는 public 함수를 실행

1 Minute Quiz



GetComponent<>() 함수를 사용하여 해당 오브젝트에 연결된 함수를 실행할 때, public 이 아닌 private로 선언된 함수를 실행하면 어떤 결과가 나오는지 확인하세요.

Summary

- Scripts provide sophisticated **object interaction** and **state**
- Scripts are attached to objects to change an **object's behaviour**
- Unity provides for scripts in JavaScript, Boo and **C#**
- The scripting language used in the module is **C#**
- C# is an **Object Oriented Programming** Language
- C# is a **strongly typed** language
- Variable types **must be declared** before compile-time
- In C#, scripts can interact with each other via their public methods
- C# has very **similar** structure and syntax **to Java**
- The **Debug Utility** provides for message output and code debugging

Unity3D Scripting Resources

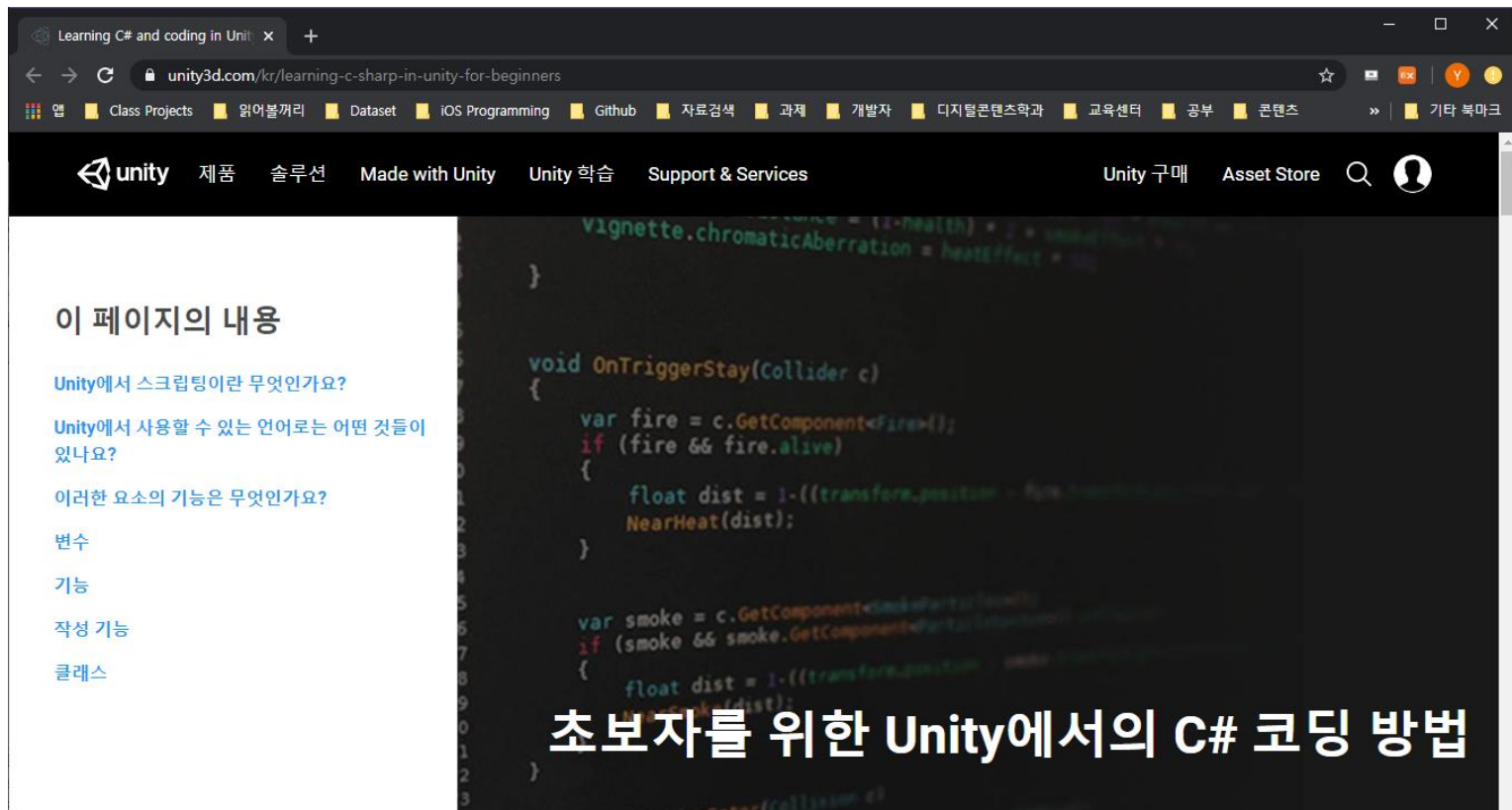
- <http://docs.unity3d.com/Documentation/ScriptReference/>
- <http://answers.unity3d.com/index.html>
- <http://www.lynda.com>



Unity Scripting

References

<https://unity3d.com/kr/learning-c-sharp-in-unity-for-beginners>



이 페이지의 내용

- Unity에서 스크립팅이란 무엇인가?
- Unity에서 사용할 수 있는 언어로는 어떤 것들이 있나요?
- 이러한 요소의 기능은 무엇인가?
- 변수
- 기능
- 작성 기능
- 클래스

```
Vignette.chromaticAberration = heatEffect * 100;
}

void OnTriggerStay(Collider c)
{
    var fire = c.GetComponent<Fire>();
    if (fire && fire.alive)
    {
        float dist = 1 - ((transform.position - fire.transform.position).magnitude / 100f);
        NearHeat(dist);
    }

    var smoke = c.GetComponent<SmokeParticleSystem>();
    if (smoke && smoke.GetComponent<ParticleSystem>().IsActive())
    {
        float dist = 1 - ((transform.position - smoke.transform.position).magnitude / 100f);
        Cook(dist);
    }
}
```

초보자를 위한 Unity에서의 C# 코딩 방법



Unity Scripting

- Unity Scripting 이란 ?
 - 스크립팅(Scripting)은 GameObject 의 동작을 정의하는 것
 - GameObject 에 연결된 스크립트와 구성 요소가 상호 작용하여 게임을 플레이하는 방식
 - Unity는 거대한 루프처럼 작동 : Unity는 게임 씬(Scene)에 위치한 모든 데이터를 (예, 조명, 메시, 동작 등) 읽은 다음, 모든 정보를 처리
 - Unity는 개별적인 단일 프레임을 연속적으로 실행 : 작성한 스크립트 명령을 통해 Unity에 지시를 내리면 Unity는 프레임마다 이를 실행
 - Unity에서 스크립트를 호출하려면 씬에 위치한 GameObject 에 연결함 : 스크립트는 Unity가 이해할 수 있는 언어로 작성하며, 사용자는 이 언어를 통해 엔진에 명령을 내림

- Unity에서 사용할 수 있는 언어로는?
 - Unity에서 사용되는 언어는 C# : Unity의 작동을 가능하게 하는 모든 언어는 오브젝트 중심의 스크립트 언어이며, 모든 언어와 마찬가지로 스크립팅 언어에도 구문이 있으며, 기본적인 요소는 변수, 함수 및 클래스이다
 - Unity 2017.3 이하 버전에는 MonoDevelop이라는 텍스트 에디터를 사용 : MonoDevelop은 코드 완성을 지원하고, 코드 작성이 잘못되면 알려주며, 단축키를 사용할 수 있다
 - 2018.1 버전부터는 Unity 커뮤니티용 Visual Studio 또는 Visual Studio, Notepad, Sublime text 등 기타 텍스트 에디터도 사용할 수 있다

– 샘플 코드가 포함된 스크립트

```
1 using System.Collections;
2 using UnityEngine;
3
4 public class DemoScript: MonoBehaviour {
5
6     //Variables
7     //Functions
8     //Classes
9
10    //Use this initialization
11
12    void Start () {
13
14    }
15
16
17    //Update is called once per frame
18
19    void Update () {
20
21    }
22
23 }
```

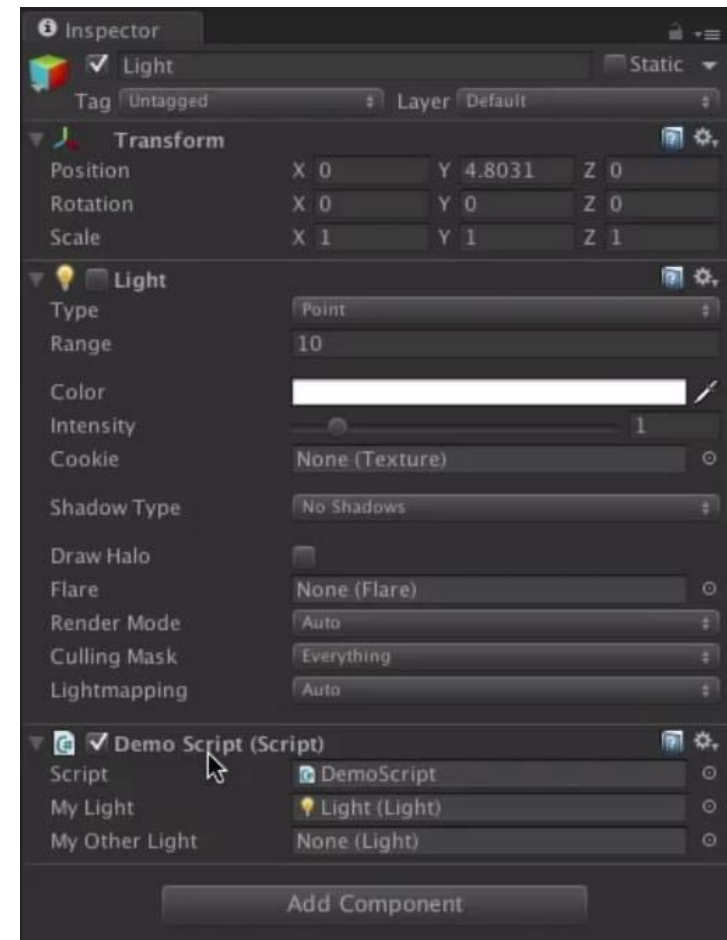
- 이러한 요소의 기능은 무엇인가요?
 - **변수**는 오브젝트의 값과 참조를 포함 : 오브젝트는 “더 큰” 변수에 해당합니다. 변수는 우리가 사용할 수 있는 요소를 포함한 상자와도 같으며, 소문자로 시작한다
 - **함수**는 이러한 변수를 비교하고 조종하는 코드 컬렉션 : 함수는 대문자로 시작하며, Unity는 프로그램의 다른 부분에서 함수를 반복적으로 재사용될 수 있도록 함수에 코드를 포함시킨다
 - **클래스**는 코드 구조화를 통해 변수와 기능의 컬렉션을 함께 래핑하여 오브젝트의 속성을 정의하는 템플릿을 만드는 방법이다
 - 스크립팅은 기본적으로 이러한 오브젝트와 오브젝트의 현 상태 및 값을 비교하는 것으로, 결과나 해결책을 논리적으로 결정하는 데 기반한다

변수

- 변수
 - Unity에서 스크립트는 대부분의 경우 변수를 선언하여 필요한 툴을 상단에 정렬하는 것으로 시작
 - 선언된 변수(여기)는 가시성 키워드 "public"이나 "private"으로 시작하며, 유형과 이름이 이어집니다.

```
5 public class DemoScript: MonoBehaviour {  
6  
7     public Light myLight;  
8     private Light myOtherLight;  
9  
10 }
```


- 변수에서 중요한 다른 한가지는 유형 : 유형은 변수가 메모리에 포함하는 값의 종류를 정의한다 (예를 들어 이러한 값은 숫자, 텍스트 또는 아래와 같이 더 복잡한 유형일 수 있다)
- 아래의 Transform, Light 및 Demo 스크립트는 구성 요소의 참조 : Unity는 이러한 구성 요소를 처리하기 위해 해당 구성 요소의 오브젝트 유형을 알아야 한다



- 기능 (함수)

- 스크립트는 기능을 사용하여 변수를 조종 : Unity에는 자동으로 실행되는 기능이 여러 개 있다

```
54 /*  
55     Awake()  
56     Start()  
57     Update ()  
58     FixedUpdate()  
59     LateUpdate  
60  
61 */
```

- 기능 작성

- 함수를 작성할 때에는 함수의 맨 앞부분에 반환된 함수 유형을 배치
- 뒤이어 함수의 이름과 괄호 안의 매개변수(있는 경우) 순으로 작성
- 함수 이름은 대문자로 시작
- 함수의 본문은 중괄호 안에 배치해야 한다

```
13  
14     void MyFunction () {  
15  
16     }  
17 }
```

- 클래스

- 이러한 변수와 함수의 컬렉션
- 클래스 이름이 C# 스크립트의 파일 이름과 일치해야 클래스가 작동한다
- 최초로 스크립트를 생성할 때 자동으로 함께 생성되는 MonoBehaviour 클래스에서 파생된 클래스만 GameObject에 첨부할 수 있다
- 클래스는 공개 또는 비공개로 설정할 수 있다

```
6 public class DemoScript: MonoBehaviour {
7
8     public Light myLight;
9
10    void Awake () {
11        int myVar = AddTwo(9,2);
12        Debug.Log(myVar);
13    }
14
15    void Update () {
16        if (Input.GetKeyDown ("space")) {
17            MyFunction ();
18        }
19    }
20
21 }
22
23 void MyFunction () {
24
25     myLight.enabled = !myLight.enabled;
26 }
27
28 int AddTwo (int var1, int var2) {
29     int returnValue = var1 + var2;
30     return returnValue;
31 }
32
33
34 }
```

References

https://learn.unity.com/course/unity-c-survival-guide?_ga=2.20645818.36413353.1585084919-2014196362.1584951635

The screenshot shows the Unity C# Survival Guide course page. At the top, there's a navigation bar with the Unity Learn Premium logo and a search bar. Below this, a banner for the course is displayed, featuring the title 'Unity C# Survival Guide' and the instructor's name 'Jonathan Weinberger'. The course is marked as 'PREMIUM' and includes details like '온라인 교육' (Online Education), '초급' (Beginner), '22시간 45분' (22 hours 45 minutes), and '3130' (likely a price or rating). A '과정 시작하기' (Start Course) button is visible. Below the banner, there's a section for '주제' (Topics) with buttons for 'Scripting' and 'Editor Essentials'. On the left, there's a '진행 상황' (Progress) section with a list of topics: '1. C# Survival Guide - Quick Tips and Assets', '2. C# Survival Guide - Rotations', and '3. C# Survival Guide - Variables'. The page also features a 'COVID-19 Support' banner at the top, stating 'We're providing all users three months of complimentary access to Unity Learn Premium.'



<https://programmers.co.kr/learn/courses/1> (Unity로 배우는 C#)

Unity로 배우는 C# | 프로그래머 x [Archive] Coding in Unity for th x +

programmers.co.kr/learn/courses/1

programmers 스킬 체크 개발자 채용 챌린지 코딩테스트 연습 프로그래밍 강의 계정 만들기 로그인 기업 회원

강의 목록 평가 Q&A

Unity로 배우는 C#

시작하기

★★★★★ 4.9 • 21개의 평가

온라인 코스

4시간 2분 39초 동영상 강의

10개의 코딩 실습

무료

6268명이 공부 중

Unity로 간단한 게임을 만들면서 C#을 배워 보세요.

파트1. 유니티 시작하기

- 유니티 설치 (3분) 미완료
- 유니티 화면 알아보기 (3분 25초) 미완료

파트2. GameObject와 Component

- GameObject란? (4분 24초) 미완료
- GameObject 조작하기 (3분 43초) 미완료
- Transform과 Inspector (3분 31초) 미완료

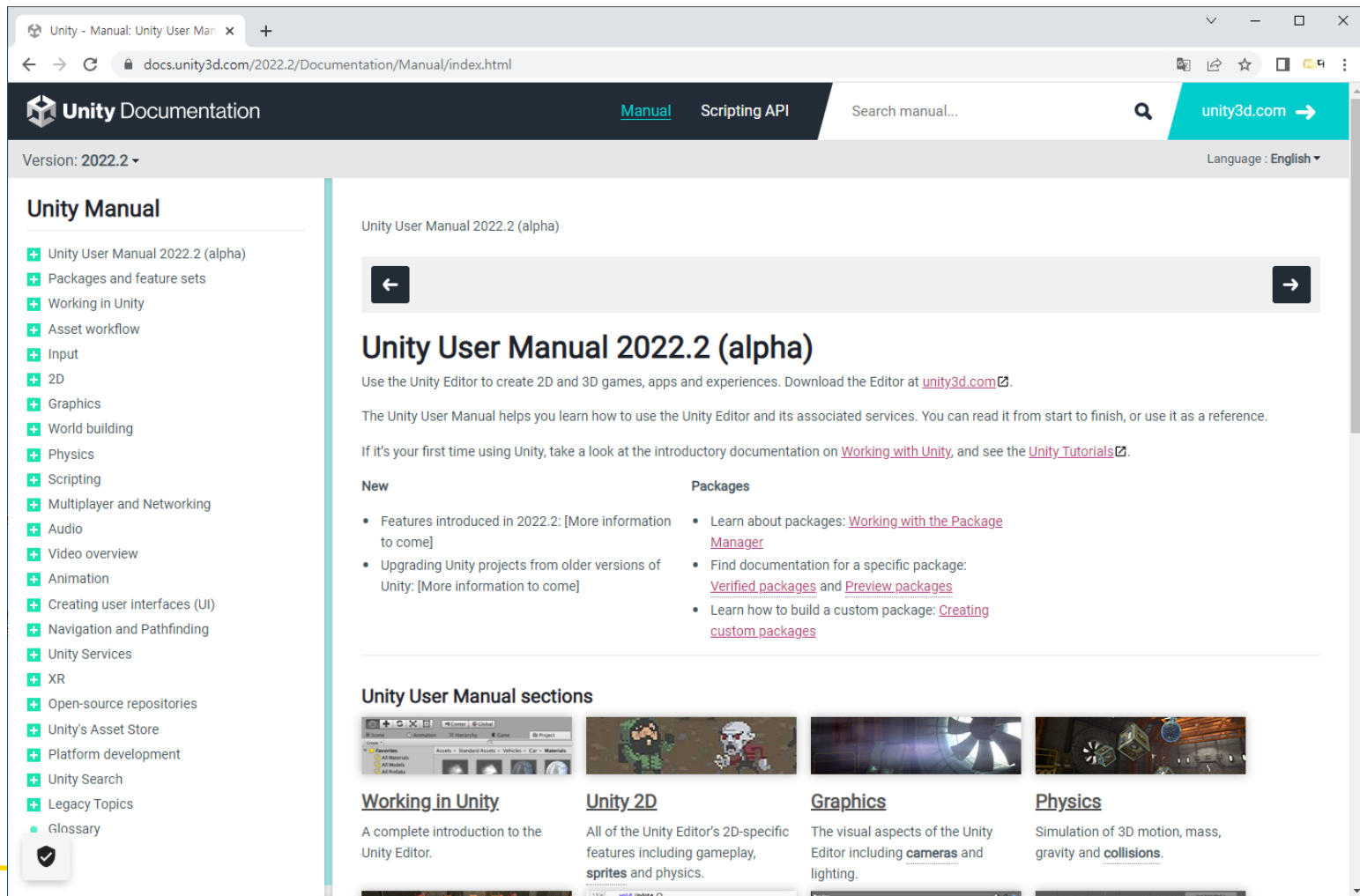


<https://learn.unity.com/tutorial/coding-in-unity-for-the-absolute-beginner#>



Reference Website

- <https://docs.unity3d.com/2022.2/Documentation/Manual/index.html>



- Script Editor를 Visual Studio 로 설정
 - 메뉴 [Edit – Preferences] 에서 External Tools 선택
 - External Script Editor 에서 “Visual Studio” 선택

