

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

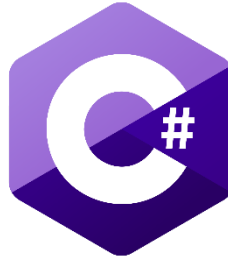
- Unity Scripting is primarily interacting with GameObject components
- Scripts are really just custom components

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

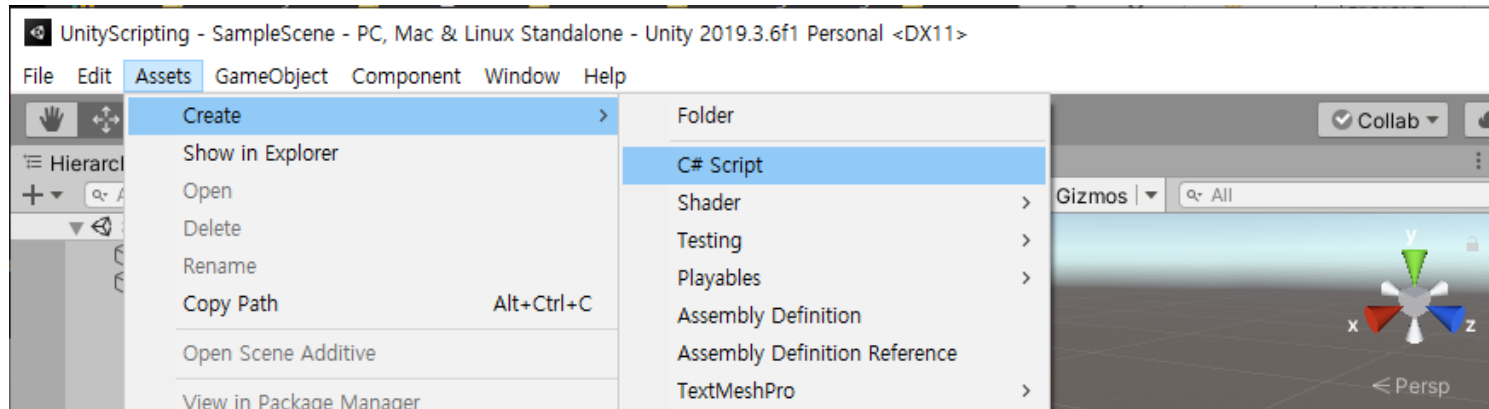


- Unity documentation provides examples for each type.
- Javascript is often used in online Unity examples, but C# is now becoming **first choice** in current Unity tutorials .
- Module using C# to align more with other modules in programs and commercial activities.

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 () {

    }

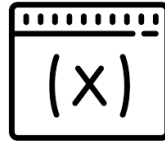
}
```


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;  
    SphereCollider 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)

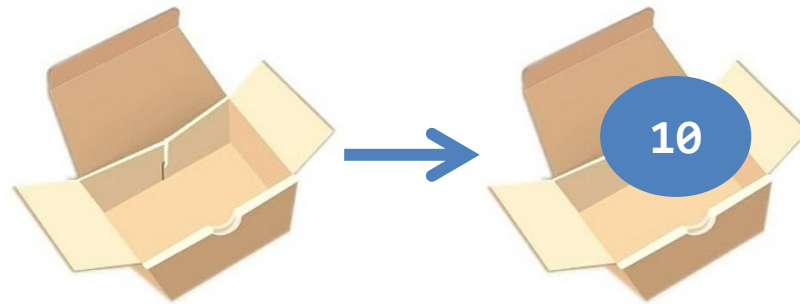

- 대입연산자(=) 사용

```
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

– 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;  
}
```

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
```


Attaching Scripts to Objects

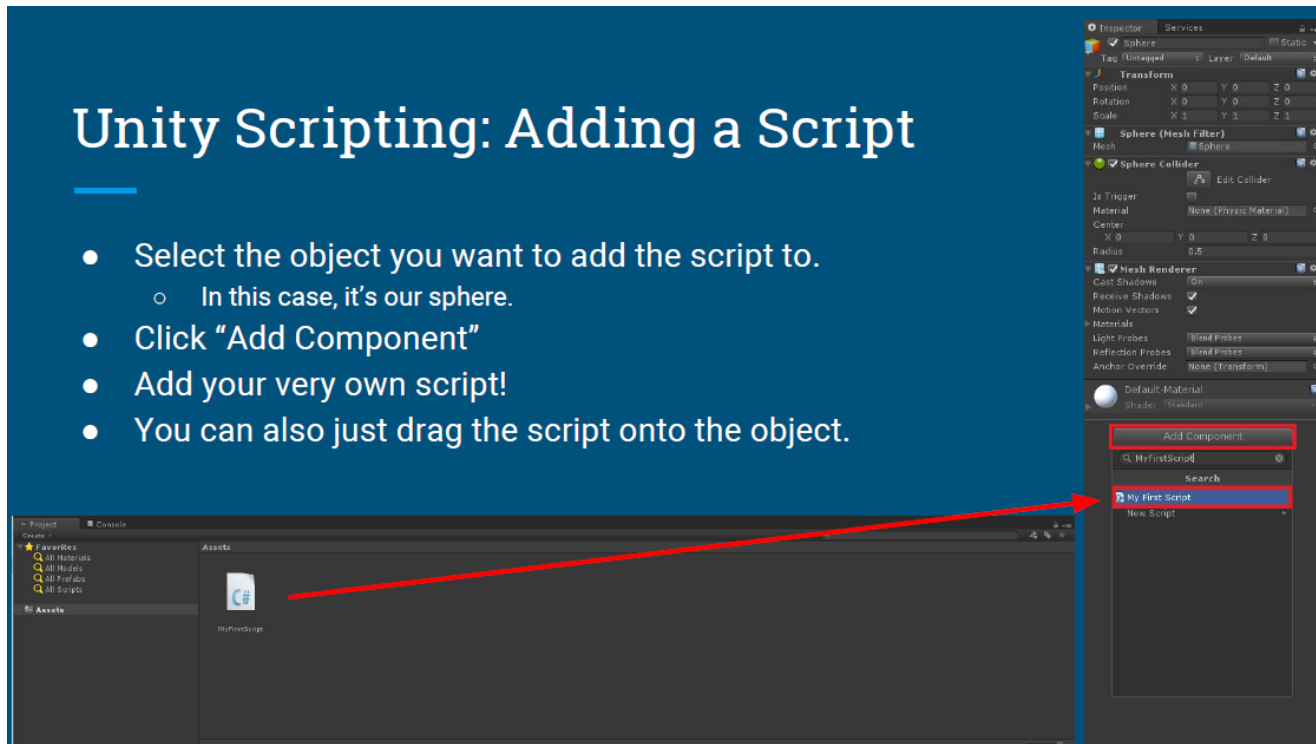
- Attaching the script
 - Create the new script
 - Name the script accordingly
 - Drag the script from the Assets Window onto the object you want the script to affect.

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

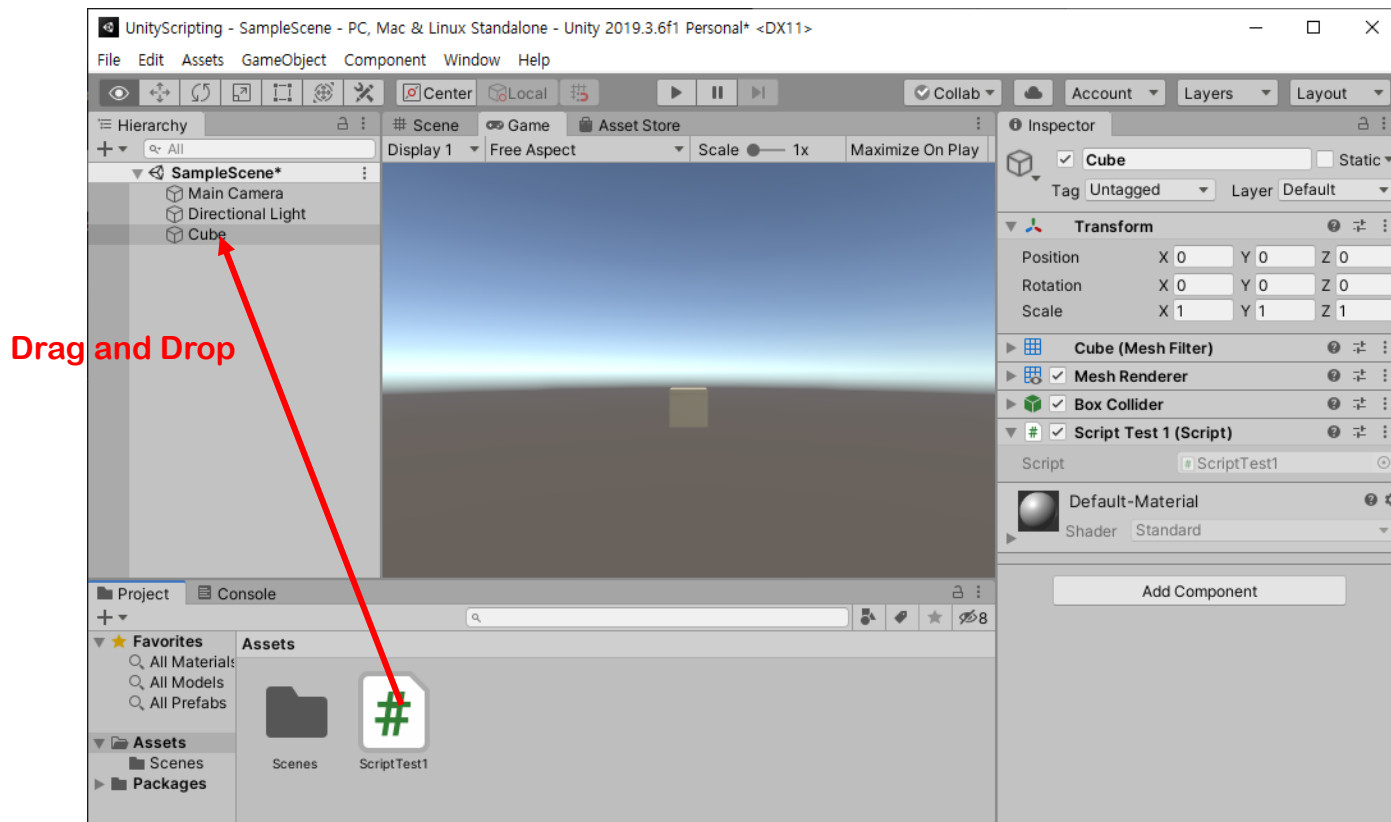
- Select the object you want to add the script to.
 - In this case, it's our sphere.
- Click "Add Component"
- Add your very own script!
- You can also just drag the script onto the object.



[출처] University of California San Diego

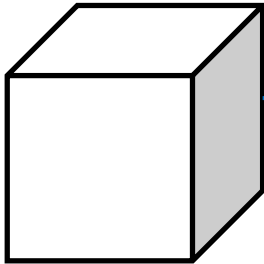
- Attaching the Script to Object

- 스트립트를 (연결하고자 하는) 객체로 드래그앤드롭



- Changing an Object's Properties via Scripts

Cube Object



ScriptTest1.cs

```
using UnityEngine;
using System.Collections;

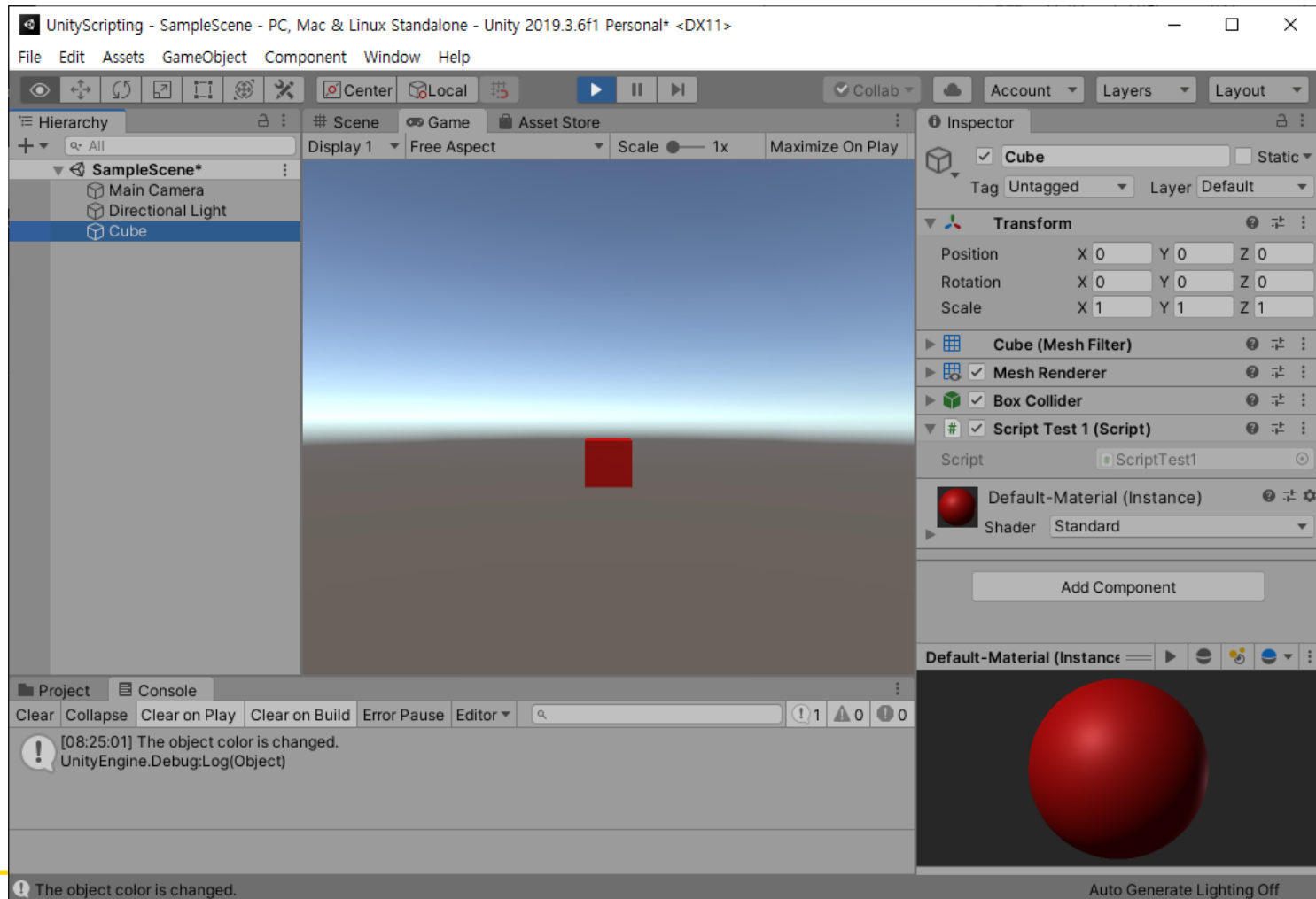
public class ScriptTest1 : MonoBehaviour {
    // Use this for initialization
    void Start () {
    }

    // 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"

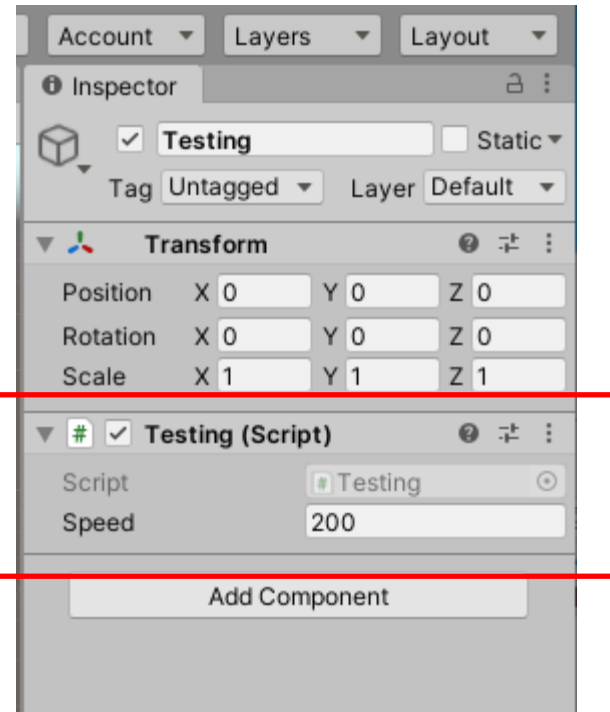


- public vs. private

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



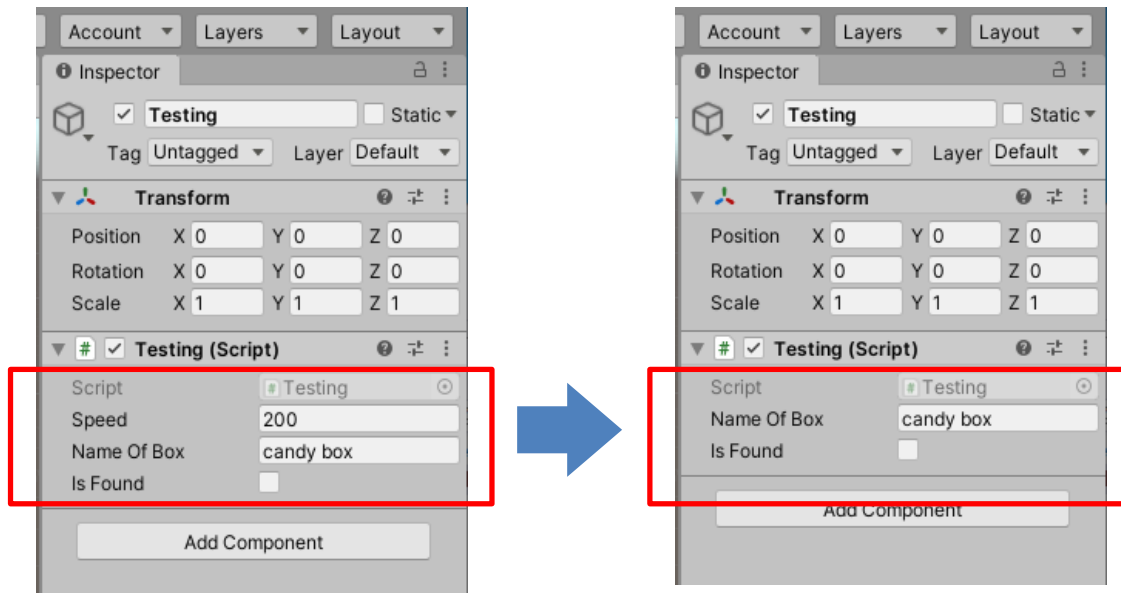
– public 변수는 인스펙터에 보여짐





– 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;
```


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);  
}
```

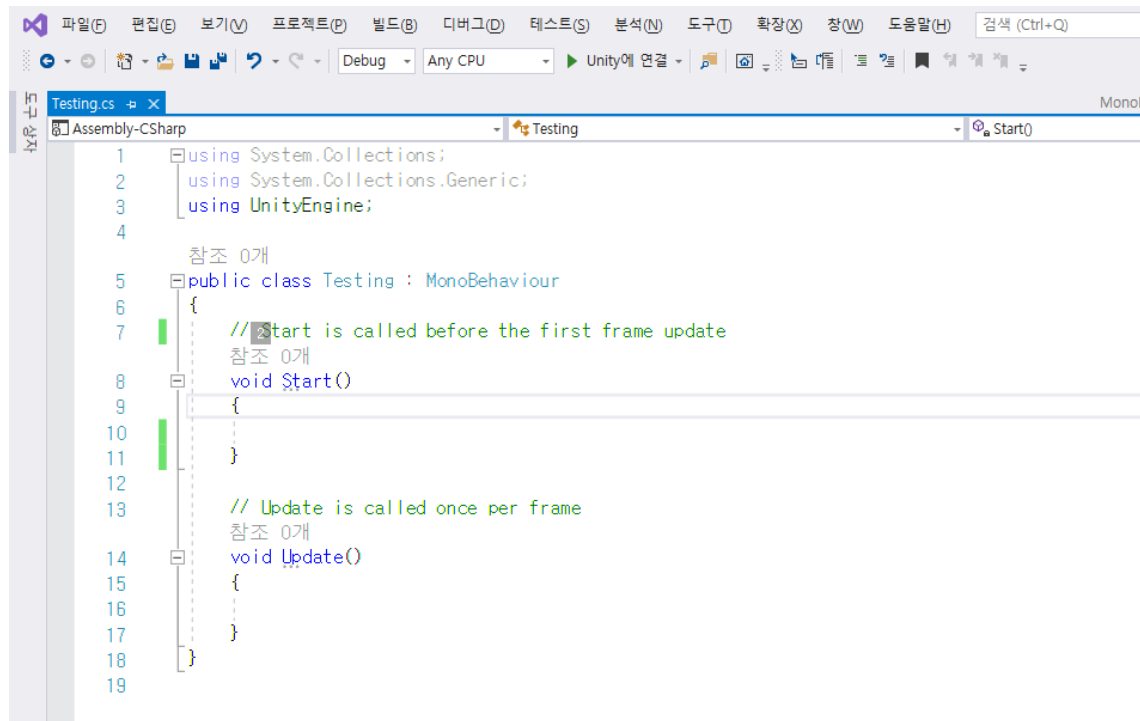
```
void callMethod1() {  
    Debug.Log("callMethod1 is called.");  
}
```

```
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 {
```

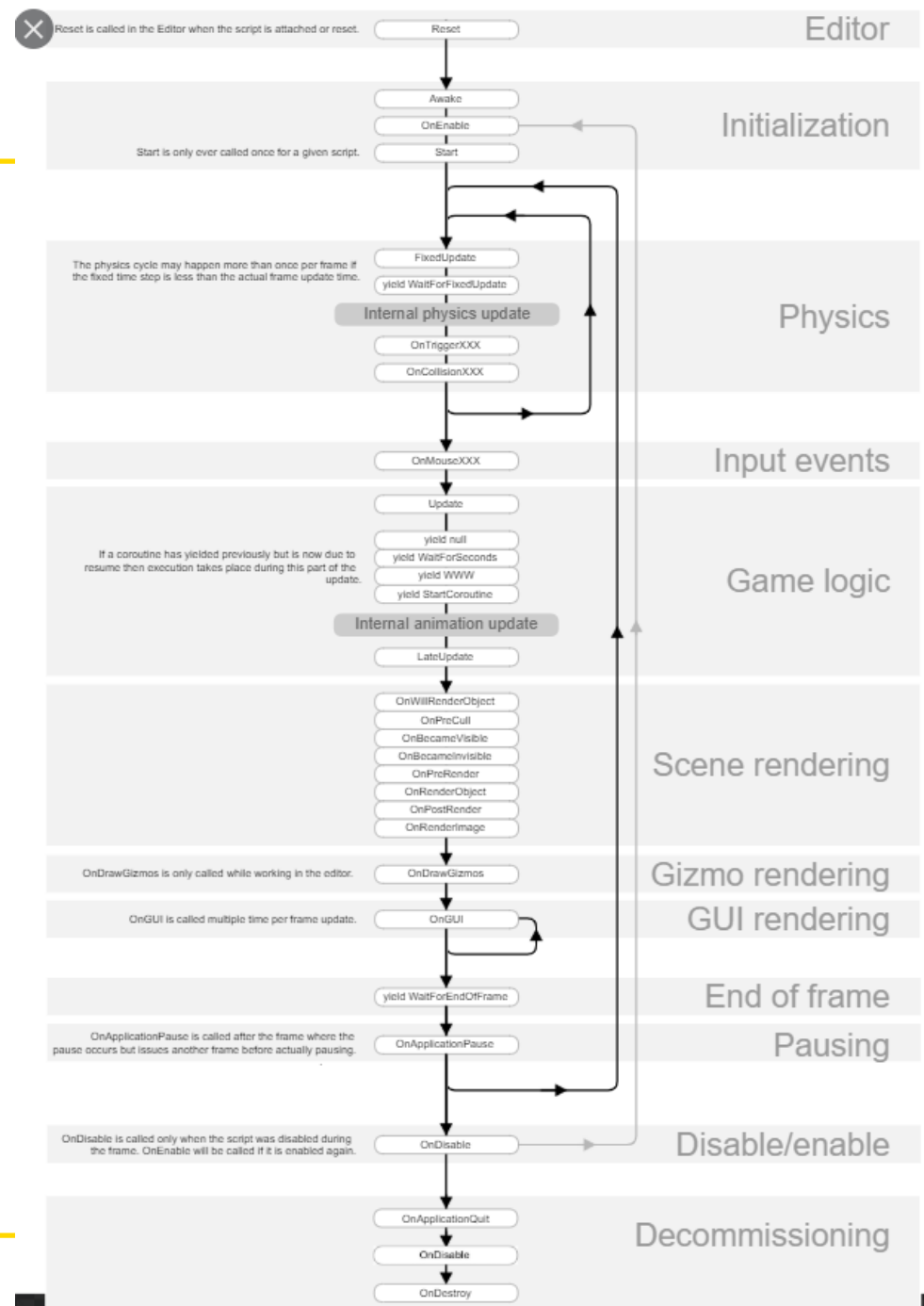
– 미리 정의되어 있는 함수들 ?

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

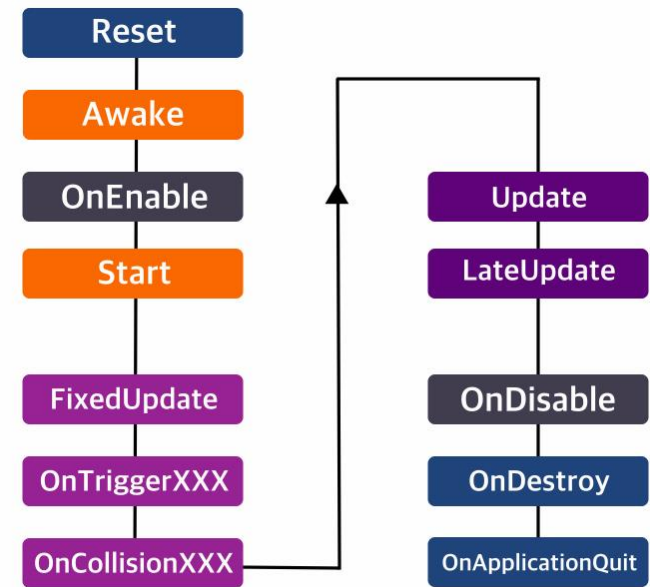
The screenshot shows the Unity Scripting API documentation for the **MonoBehaviour** class. The browser address bar shows `docs.unity3d.com/ScriptReference/MonoBehaviour.html`. The Unity logo and "DOCUMENTATION" text are visible in the header. The left sidebar lists various UnityEngine namespaces, with **UnityEngine** expanded. The main content area is titled **MonoBehaviour** and includes the following information:

- class in UnityEngine / Inherits from: Behaviour / Implemented in: UnityEngine.CoreModule**
- Description**
 - MonoBehaviour is the base class from which every Unity script derives.
 - When you use C#, you must explicitly derive from MonoBehaviour.
 - This class doesn't support the [null-conditional operator](#) (?) and the [null-coalescing operator](#) (??).
 - For code samples, see the individual MonoBehaviour methods.
- Note:** There is a checkbox for enabling or disabling MonoBehaviour in the Unity Editor. It disables functions when unticked. If none of these functions are present in the script, the Unity Editor does not display the checkbox. The functions are:
 - [Start\(\)](#)
 - [Update\(\)](#)

- 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;
    }

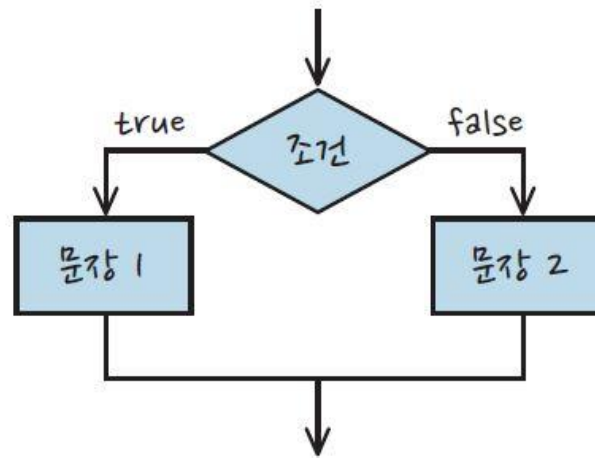
    // Start is called before the first frame update
    void Start()
    {
        string retValue = CalculateSum(100, 200);
    }

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

제어문

- 조건문 if () else

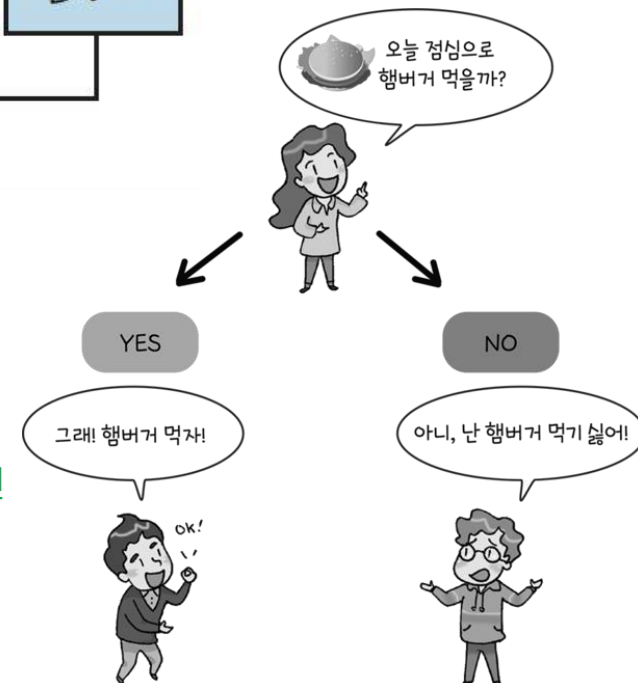
```
if ([불 표현식])  
{  
    // 불 표현식이 참일 때 실행할 문장  
}  
else  
{  
    // 불 표현식이 거짓일 때 실행할 문장  
}
```



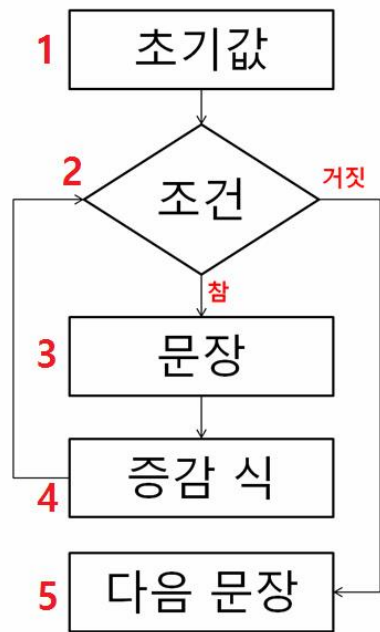
- 다중 조건

if (A && B) // A가 참이면서 동시에 B도 참이라면

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



- 반복문 for()



<for문 순서도>

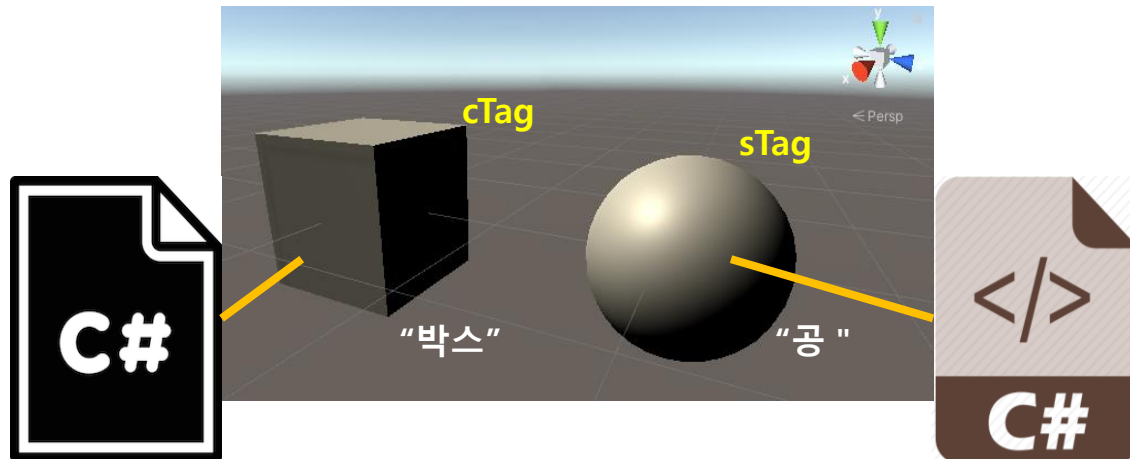
```

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

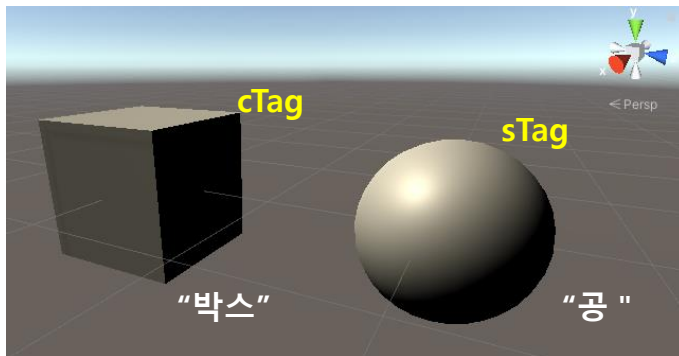
<문법>

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



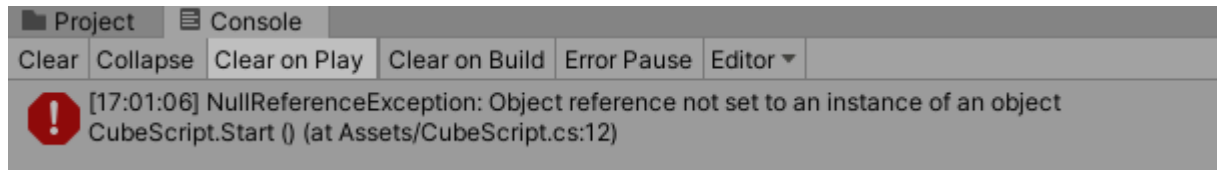
“Cube”에 연결된 “CubeScript”에서 “Sphere”를 찾기 위해 Find() 함수를 사용하였다. 만약에 씬에 있는 오브젝트를 찾지 못할 경우, 어떤 결과 값이 반환되는지 확인하세요.

- 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);  
    }  
}
```

1 Minute Quiz



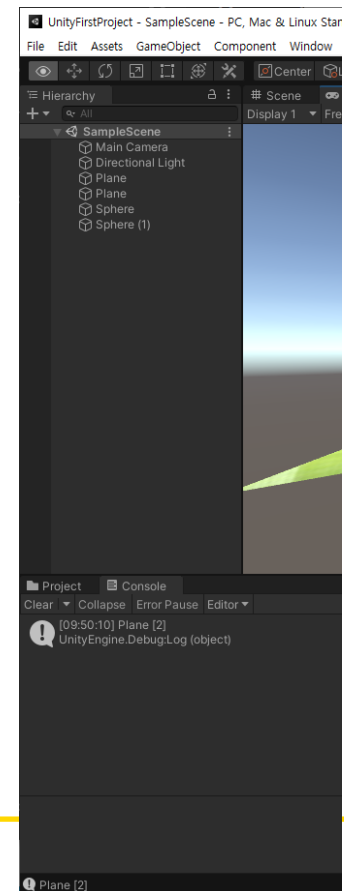
Find() 함수를 사용하여 씬에 있는 오브젝트를 찾는데, 만약 동일한 이름으로 2개 이상의 오브젝트가 존재한다면, 어떤 결과가 나오는지 확인하세요.

1 Minute Quiz




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

- 먼저 탐색되는 객체 하나만을 가져온다.
- 만약, 동일한 이름의 모든 객체를 가져오고자 한다면 ?



- 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



비활성화시킨 후, 검색을 해보세요


Collab

Account

Layers

Layout

Inspector


☒ Sphere

Tag

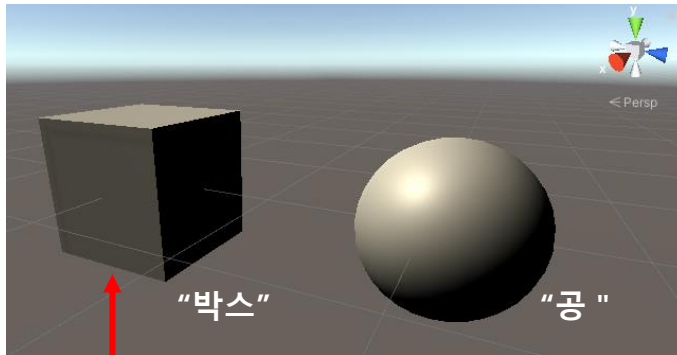
sphereTag

Layer

Default

Transform

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



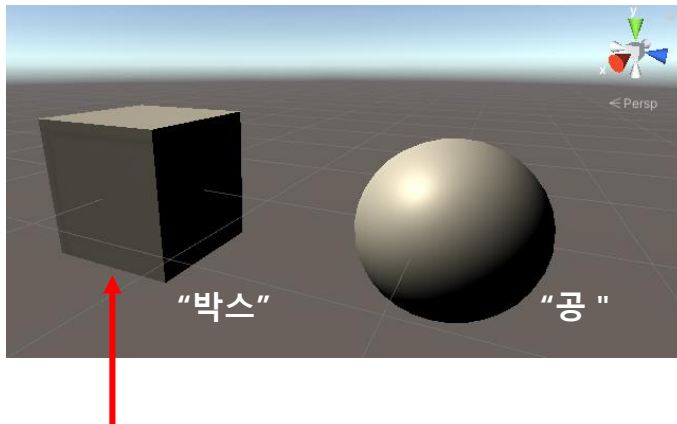
```
void Start() {  
    GameObject.Find("공").SendMessage("WhoAmI", "YongHwan");  
    GameObject.Find("공").SendMessage("WhoAreYou");  
}
```

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

– 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

❖ 여기서 컴포넌트는 인스펙터에 추가되는 컴포넌트를 의미하며, 물리엔진, 셰이더, 스크립트, 오디오, 애니메이션 등의 컴포넌트를 다룰 수 있음



```
void Start() {  
    GameObject.Find("공").GetComponent<SphereScript>().WhoAmI("Call");  
}
```

Script Name이 SphereScript 라고 할 때

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

1 Minute Quiz

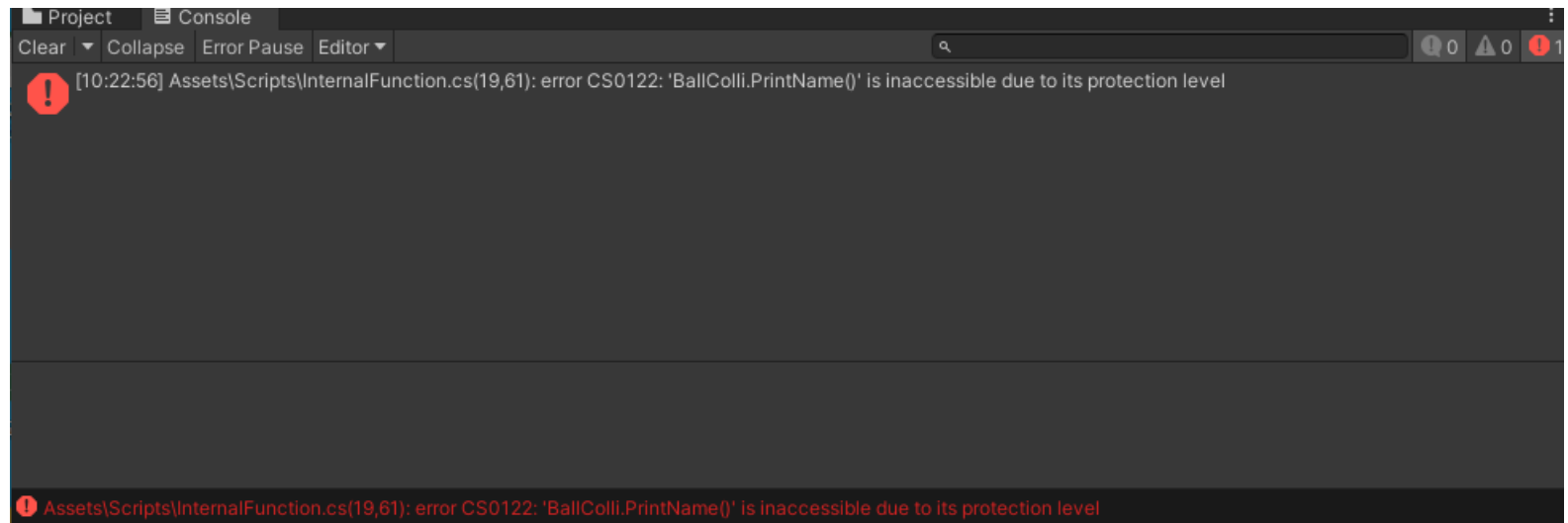


GetComponent<>() 함수를 사용하여 해당 오브젝트에 연결된 함수를 실행할 때, `public` 이 아닌 `private`로 선언된 함수를 실행하면, 어떤 결과가 나오는지 확인하세요.

1 Minute Quiz



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



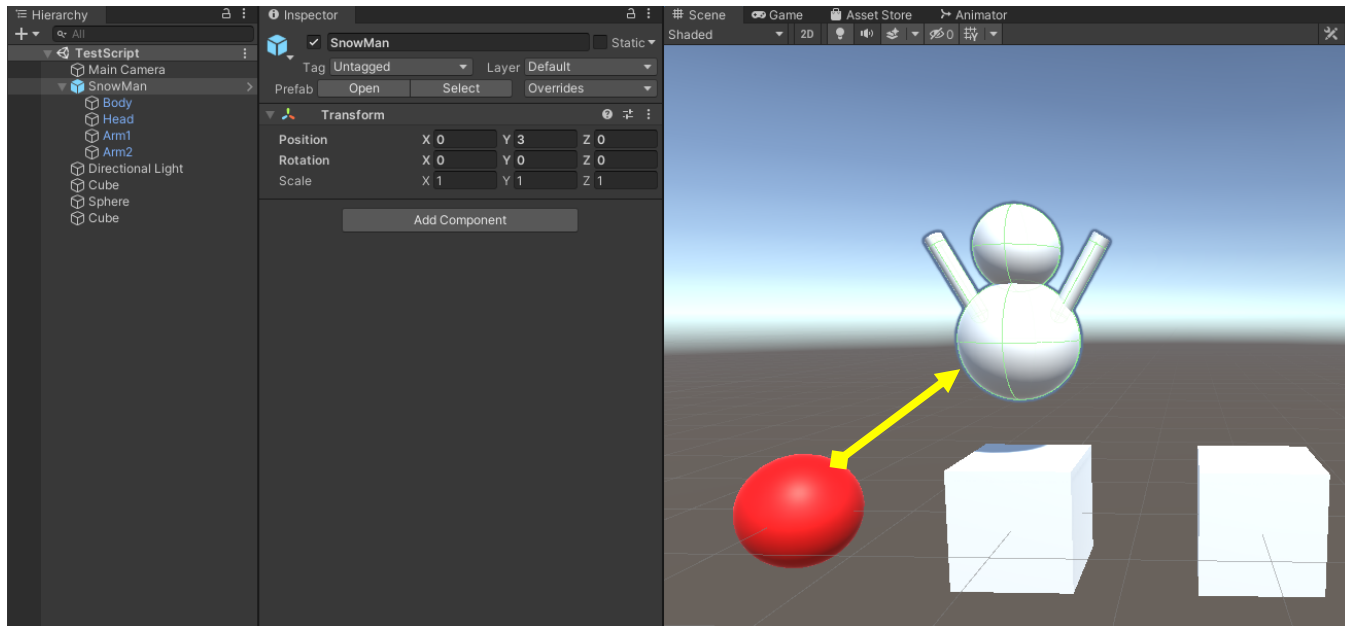
1 Minute Quiz



GetComponent<>() 함수를 사용하여 해당 오브젝트에 연결된 함수를 실행하는 방법과 SendMessage() 함수를 사용하여 해당 오브젝트에 연결된 함수를 실행하는 방법의 차이를 확인하세요.

- 객체.GetComponent<>().해당함수() : 해당함수가 반드시 public 으로 선언되어 있어야 함. 함수의 매개변수 전달 개수에 상관없음.
- 객체.SendMessage("해당함수") : 해당함수가 public 이든 private 든 상관없이 호출이 가능함. 함수의 매개변수 전달의 제한(?)

- 특정 오브젝트의 자식 오브젝트로 접근
 - Find() 등을 이용하여 다른 오브젝트를 찾음
 - 1. 선택 오브젝트 바로 아래에 있는 자식 오브젝트들만 접근
 - 2. 선택 오브젝트 아래에 있는 모든 자식 오브젝트들로 접근



– Find() 등을 이용하여 다른 오브젝트를 찾음

1. 선택 오브젝트 바로 아래에 있는 자식 오브젝트들만 접근
2. 선택 오브젝트 아래에 있는 모든 자식 오브젝트들로 접근

..

```
GameObject getObj = GameObject.Find("SnowMan");
if (getObj != null) {
    Debug.Log("Object is found");

    // getChild = getObj.GetComponentInChildren<Transform>();
    // GetComponentInChildren 는 지정한 컴포넌트 하나만 추출
    // 자식 객체가 여러 개라면 TransformSort 기준으로 가장 상위에 있는 자식 객체의 컴포넌트
    Transform getChild = getObj.transform.GetChild(0);
    Debug.Log(getChild.name);
} else {
    Debug.LogWarning("Object is NOT found");
}
```

– Find() 등을 이용하여 다른 오브젝트를 찾음

1. 선택 오브젝트 바로 아래에 있는 자식 오브젝트들만 접근
2. 선택 오브젝트 아래에 있는 모든 자식 오브젝트들로 접근

```
..
GameObject getChildObject = GameObject.Find("SnowMan");
if (getChildObject != null) {
    Debug.Log("객체를 찾았습니다. 이번에는 여러 자식 객체를 찾아보겠습니다");

    Transform[] allChildren = getChildObject.GetComponentsInChildren<Transform>();
    if (allChildren != null) {
        foreach(Transform child in allChildren) {
            // GetComponentsInChildren 에서는 자기자신도 추출
            Debug.Log(child.name);
        }
    } else {
        Debug.LogWarning("There are no child ..");
    }
} else {
    Debug.LogWarning("객체를 찾지 못했습니다 ..");
}
```

Summary

- Scripts provide sophisticated **object interaction** and **state**
- Scripts are attached to objects to change an **object's behavior**
- 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>

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 various menu items like '프로젝트', '과정', '튜토리얼', '실시간 학습', and '주제'. A search bar is also present. Below the navigation bar, a banner for the 'Unity C# Survival Guide' course is displayed, featuring a thumbnail image of a person coding. The course is labeled 'PREMIUM' and 'Jonathan Weinberger'. Below the banner, there's a section titled '진행 상황' (Progress) with a list of lessons: '1. C# Survival Guide - Quick Tips and Assets', '2. C# Survival Guide - Rotations', and '3. C# Survival Guide - Variables'. To the right of the progress section, there's a '요약' (Summary) section with a paragraph about the course. Further right, there's a '주제' (Topics) section with two buttons: 'Scripting' and 'Editor Essentials'. The bottom of the page shows a Google Chrome logo.

Learning C# and coding in Unity x Unity C# Survival Guide - Unity x

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

Class Projects | 읽어볼꺼리 | Dataset | iOS Programming | Github | 자료검색 | 과제 | 개발자 | 디지털콘텐츠학과 | 교육센터 | 공부 | 콘텐츠 | 기타 북마크

COVID-19 Support: We're providing all users three months of complimentary access to Unity Learn Premium.

unity Learn Premium 프로젝트 과정 튜토리얼 실시간 학습 주제 ▼

무엇을 배우고 싶으세요? YL

과정 시작하기

UNITY C# SURVIVAL GUIDE

PREMIUM

Unity C# Survival Guide

온라인 교육 • 초급 • 22시간 45분 • 3130

Jonathan Weinberger

개요 세부 사항

진행 상황

내 학습 진행 상태

1. C# Survival Guide - Quick Tips and Assets

2. C# Survival Guide - Rotations

3. C# Survival Guide - Variables

요약

Have you been struggling to learn how to code in C# with Unity? If so, you have found the course you've been searching for! This course is designed for beginners and advanced or professional programmers alike.

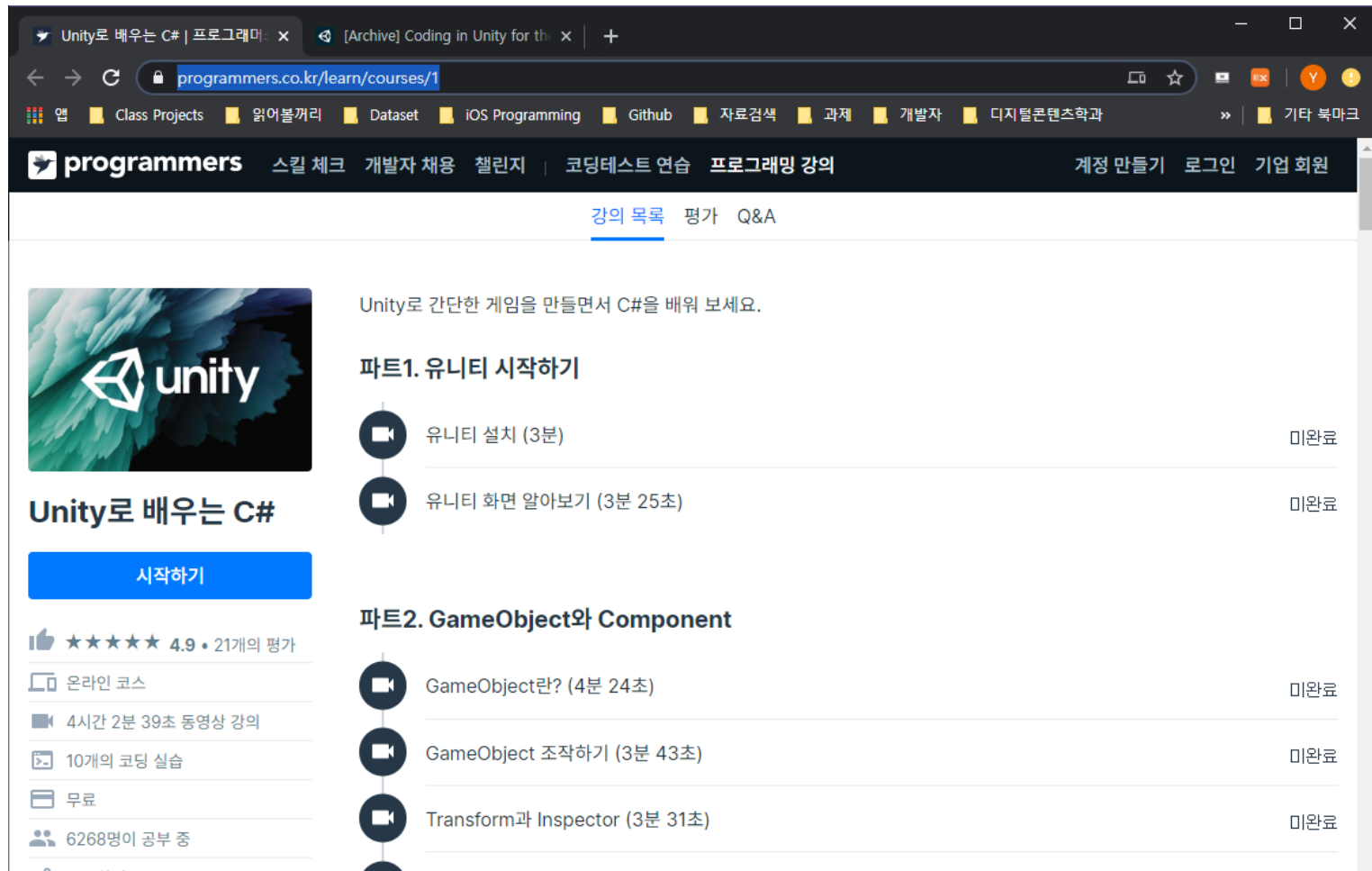
For beginners, sequential completion of the lessons is recommended, as it will teach you the basics of coding using interactive challenges and problem solving techniques. The course

주제

Scripting

Editor Essentials

<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

Class Projects | 읽어볼꺼리 | Dataset | iOS Programming | Github | 자료검색 | 과제 | 개발자 | 디지털콘텐츠학과 | 기타 북마크

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#>

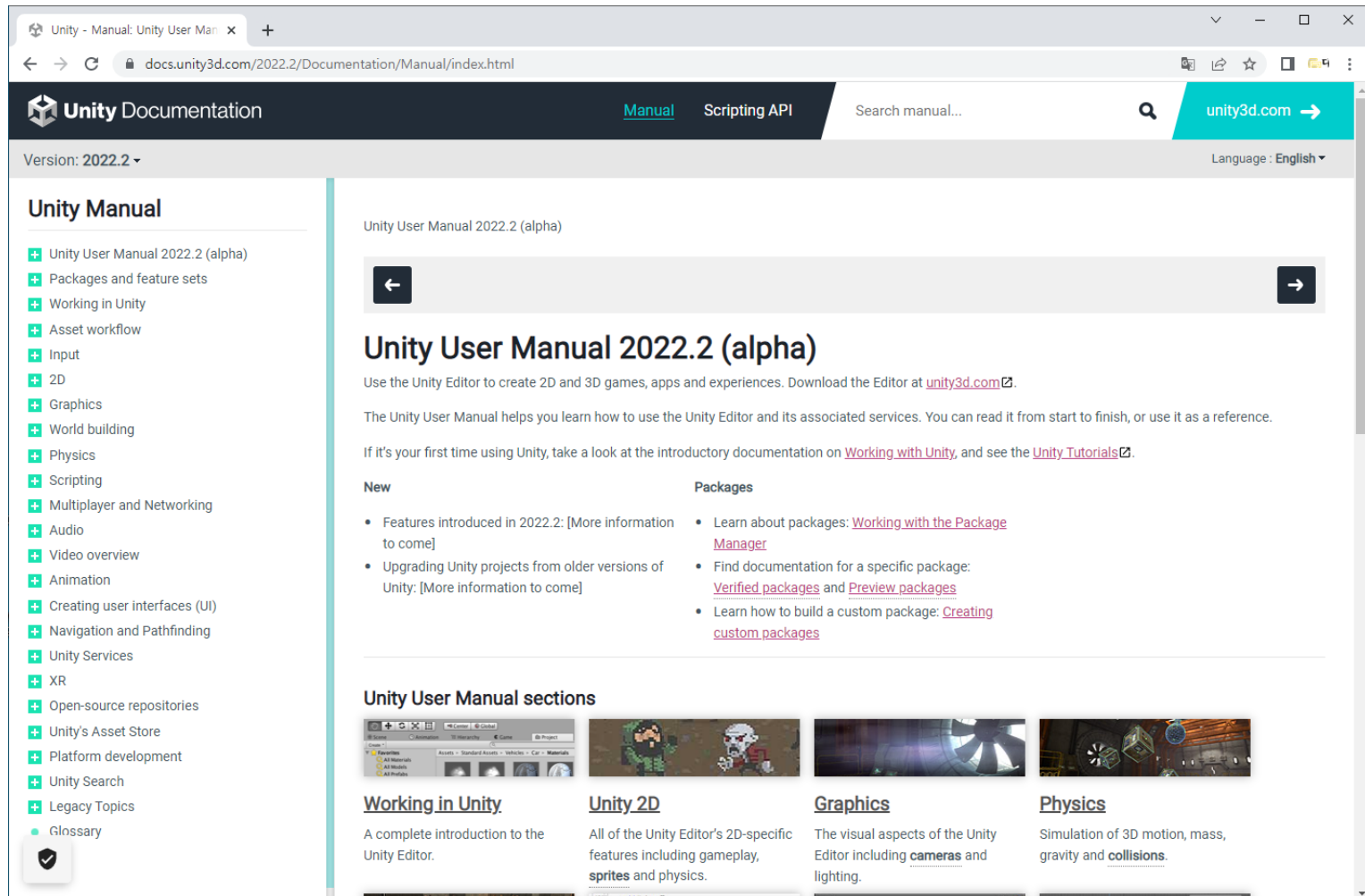


The screenshot shows a web browser window with the URL <https://learn.unity.com/tutorial/coding-in-unity-for-the-absolute-beginner#>. The browser's address bar and tabs are visible at the top. Below the browser window, a dark blue banner reads "COVID-19 Support: We're providing all users three months of complimentary access to Unity Learn Premium." The main header of the website features the "unity Learn Premium" logo, navigation links for "프로젝트", "과정", "튜토리얼", "실시간 학습", and "주제", a search bar with the text "무엇을 배우고 싶으세요?", and a user profile icon labeled "YL". A blue button labeled "튜토리얼 시작하기" is positioned on the right. The main content area has a dark background with a large white title "[Archive] Coding in Unity for the Absolute Beginner". To the left of the title is a small image of a code editor showing C# code. Below the title, it says "튜토리얼 • 초급 • 1시간 10분 • 411" and the Unity Technologies logo.



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

