



Çaylaklar İçin UnityScript'in Temelleri

By iGraphic Visuals LLC 2010

Orijinal Kaynak: <http://www.unityscript.com/lessons1/basics.php>

Yazar: Terry Norton

ÇEVİRİ

Süleyman Yasir KULA

WEB URL

<http://yasirkula.wordpress.com/>

İÇİNDEKİLER

1- Özellikle Çaylaklar İçinindir

1.1	1: Giriş	4
1.2	2: Akılsız Objelerden İbaret Şey - Oyun	8
1.3	3: Yaratma Kompleksi	12

2- GameObject (Oyun Objesi): Scriptler ve Componentler (Parçalar)

2.1	4: Class (Sınıf), Component (Parça) ve Script	15
2.2	5: Scriptlerde Değişkenler (Variable)	18
2.3	6: Scriptlerde Fonksiyonlar (Function)	21
2.4	7: Scriptlerde Dot Syntax: İletişim Kurmak	23
2.5	8: Sonuç	25

3- Değişkenlere Detaylı Bakış

3.1	9: Değişkenler ve Component Özellikleri	27
3.2	10: Değişken (Variable) isimleri ve “çeşitleri”	32

4- Fonksiyonlara Detaylı Bakış

4.1	Unity'nin Update() ve Start() fonksiyonları	44
4.2	Fonksiyon İsimleri	49
4.3	Fonksiyonun çağrılması ve akışı	53

5- Seçimler, seçimler, her zaman seçim yapmak zorundayız

5.1	if (eğer)?	58
-----	------------	----

6- Dot Syntax (Noktasal Yapı): Component'ler (Parça) Arası İletişim

6.1	Scriptten dışarısıyla iletişim	70
6.2	Yumurta scripti, 1 - 5 arası satırlar	74
6.3	Yumurta scripti, satır 7 - 11, Update() fonksiyonu	79
6.4	Yumurta scripti, 13. satır, Yumurtayı Döndürmek	81
6.5	Yumurta scripti, 15. satır, Eğimi (Rotation) İzlemek	84
6.6	Yumurta scripti, 17 - 23, Döndürmeyi Durdurmak	87

7- Inspector Panelini Kullanarak Atama Yapmak (Assign)

7.1	Ustaca düşünülmüş bir Unity özelliği (feature)	95
7.2	Inspector ve tek bir script kullanarak işlem yapmak	98

8- Tebrikler, artık scriptlemenin en önemli temel bilgilerini biliyorsunuz

8.1	Sırada ne var?	101
-----	----------------	-----

ÖZELLİKLE ÇAYLAKLAR İÇİNDİR

1: Giriş

Başlamadan önce bir uyarı – Bu kullanım kılavuzu özellikle hiçbir programlama bilgisi olmayan ve UNITY kullanmak isteyen kişiler için hazırlanmıştır. Buradan elde edeceğiniz bilgilerle programlamaya isinip, bir objeye (GameObject) script (kod) yazarken yaşanan korkuyu yenmeniz amaçlanmıştır

Scripting (Kod yazma), veya bir başka deyişle programlama; özellikle tamamen çaylaklar (noob) için çok basit adımlarla öğrenilecektir. Bir obje için muazzam kodların nasıl oluştuğuyla ilgili korkunuzu yeneceksiniz.

Aslında UnityScript'in temellerini kavramak oldukça kolaydır. Bizler scriptlerde uyguladığımız işlemleri aslında gerçek hayatımızda da her zaman yapıyoruz.

Programlamayla ilgili gerçekten çok detaylı kitaplar, kılavuzlar, eğitim dosyaları mevcuttur. Ayrıca Unity için pek çok video ders de bulunmaktadır. Bunların çoğu da scriptlemenin temellerini anlatmaktadır. Şu an okuduğunuz bu eğitim kılavuzunu tamamladıktan sonra tüm bu diğer eğitici dosyalara bakarken çok daha rahatlayacak, her şeyin nasıl geliştiğini kavrayacaksınız.

Ben de Unity öğrenmeye çalışan bir çaylağım. Umarım bu kılavuz script yazmadaki konseptleri anlaşılır kılar. Bir başka "çaylak" olan benden ders almak; aynı perspektiften olaylara bakabilmemizi sağlayacaktır.

1'den 7'ye kadar olan dersler herhangi bir programlama geçmişi olmayan, tamamen çaylak kimseler içindir. Ciddi anlamda düşük düzeyde, basitçe anlatım yapacağım. Her gün yaptığımız şeyleri örnek vererek ilerleyeceğim. Pes etmeye neden olan kötümser düşüncenin sizlerde oluşmaması için gayret edeceğim.

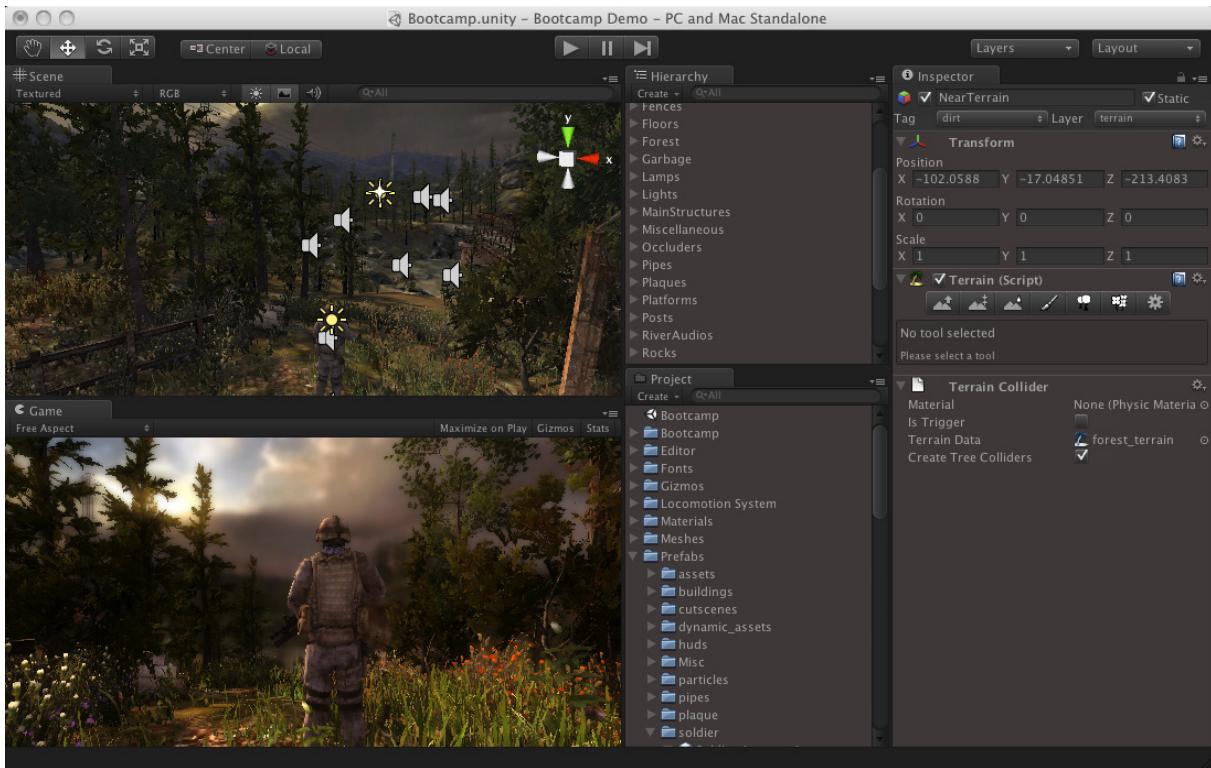
8. ders ve sonrası biraz daha gelişmiş düzeyde anlatım sunacak ve UnityScript'in konseptlerini anlatacak. Gerçek olan bir şey var ki herkes Unity programını kullanmayı sökebiliyor, ancak bu herkesin ortak korkusu ise scriptleme işlemi. Bu kılavuzu okumadan önce Unity'nin arayüzü, çeşitli komutları hakkında bilgi sahibi olduğunuzu varsayıyorum.

NOT: Bu kılavuz boyunca kimi yerlerde bulunan ve Script Referans Dosyası'na yönlendirme yapan linkler teknik bir sorundan dolayı olsa gerek işlememektedir. Bunun için şöyle bir şey yapabilirsiniz:

<http://unity3d.com/support/documentation/ScriptReference/index.html>

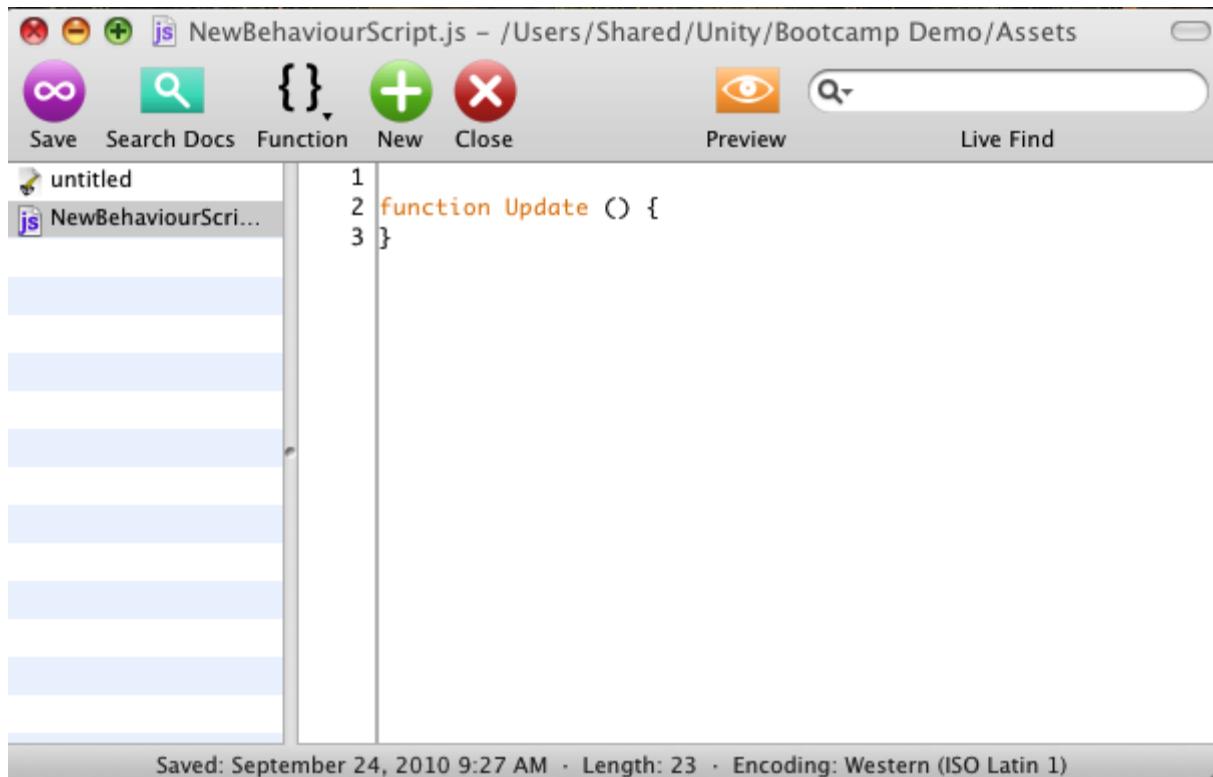
Ne zaman Script Referans Dosyası'na giden bir link olursa bu linkে tıklayın ve ilgili yerdeki link sizi hangi konuya götürüyorrsa, buradaki açılan sayfadan sol üstteki Search vasıtasyyla o konuyu aratın. (Çevirmen Eklemesi)

Unity Editörü



Bu görüntü Unity 3'deki Bootcamp Demo oyunundandır. Bu derse başlamadan önce başka eğitim dosyaları vasıtıyla **Unity'nin arayüzüünü kullanmayı biliyor olmalısınız**. Buradaki dersler arayüzü kullanmak ile ilgili DEĞİLDİR. Buradaki dersler sadece Unity'de UnityScript ile scriptlemeyi anlatmaktadır.

Script Editörü



Bu Unitron, Unity'nin Mac işletim sistemi sürümündeki varsayılan script editörü. Windows'ta bu editör farklı olacaktır, ancak sonuçta hepsinin amacı aynıdır.

Eğer UnityScript ile ilgili en ufak bilginiz yoksa, ve hatta "script" kelimesinin ne demek olduğuyla ilgili bir fikriniz yoksa o zaman bu dersler tam size göre. **Eğer sırf bu editörün görünümü bile sizi korkutuyorsa o zaman bu dersler tam size göre.** Eğer Unity'de oyun tasarlama isteyen bir artistseniz ve yaratıcı düşünce sizde varsa o zaman bu dersler tam size göre.

Ne Öğreneceksiniz?

The screenshot shows the Unity Script Reference - Scripting Overview page. The title bar says "Unity Script Reference - Scripting Overview". The main navigation bar has tabs for "Manual", "Reference", and "Scripting", with "Scripting" being the active tab. On the left, there's a sidebar with a search bar and a menu. The menu includes sections like "Menu", "Overview", "Runtime Classes", "Editor Classes", "Scripting Overview", "Common Operations", "Time", "Accessing Components", "Accessing Objects", "Variables", "Instantiate", "Coroutines & Yield", "Using C#", "Important Classes", "Performance Optimization", and "Script Compilation". The main content area is titled "Scripting Overview" and contains several sections: "This is a short overview of how scripting inside Unity works.", "Scripting inside Unity consists of attaching custom script objects called behaviours to game objects. Different functions inside the script objects are called on certain events. The most used ones being the following:", "Update:", "FixedUpdate:", "Code outside any function:", "Note: Sections of this document assume you are using Javascript, but see Writing scripts in C# for information about how to use C# or Boo scripts.", "You can also define event handlers. These all have names starting with on, (i.e. OnCollisionEnter). To see the full list of predefined events, see the documentation for MonoBehaviour.", and "Subsections" which lists "Common Operations", "Keeping Track of Time", "Accessing Other Components", "Accessing Other Game Objects", "Vectors", "Member Variables & Global Variables", "Instantiate", "Coroutines & Yield", "Writing Scripts in C#", "The most important classes", "Performance Optimization", and "Script compilation (Advanced)".

[Scriptleme Referans Arşivi](#)

Burada; aslında scriptleme ile epeydir samimi olduğunuzu öğreneceksiniz, siz böyle düşünmeseniz bile. Bu yüzden ilk birkaç derste çok yüzeysel anlatım yapacağım, böylece Unity'de scriptlemenin aslında farketmesenizde her zaman yaptığınız şeylerle çok benzer olduğunu göreceksiniz.

2: Akılsız Objelerden İbaret Şey – Oyun

Oyun ekipmanlarınızı (asset) hazırladınız. Karşınızda oyun objelerinizi (GameObject) gösteren pencere bulunuyor. Bu GameObject'lerin etrafta gezinmesini, dinlemesini, konuşmasını, bir objeyi alıp kaldırmasını, kötü adamları vurmasını veya hayalinizdeki herhangi bir şeyi yapmasını istiyorsunuz. Ne bahtsızlık ki tüm bu GameObject'ler tamamen akılsızdır, tüm bu eşyalarla ne yapacaklarını bilemezler.

Tam burada devreye scriptleme girer. “Ama ben programlama bilmiyorum” diyebilirsiniz. Pekâlâ, aslında hepiniz programlama biliyorsunuz. Bunu her gün yapıyorsunuz ancak buna programlama, ya da scripting demiyorsunuz. Her gün boş boş oturuyor musunuz yoksa insanlarla etkileşime geçiyor, evcil hayvanları seviyor, başkalarıyla konuşuyor, tuvalete gitmeyi düşünüyor musunuz?

Adım 1



Hadi egzersiz yapalım:

Bir koltukta oturmuş, arkadaşınızla sohbet ettiğinizi düşünün.

Adım 2



Ardından telefonunuz çalışıyor. Siz "Afedersin" diyorsunuz ve telefona bakıyorsunuz.

Step 3



Telefonla konuşmanızı yapıyor ve ardından arkadaşınızla sohbetinize devam ediyorsunuz.

Şimdi düşünün

Hiç hayatınızda bu veya buna benzer bir şey yaşadınız mı? Bunu yapmak zor muydu? Bahse girerim telefonu konuşmanızın yarısında alıp cevaplamak sırasında bunun farkında bile değildiniz.

Basitçe bu eylemleri yaptınız sadece, ve tahmin edin noldu? Bu yaptığınız şeyler de scriptlerin GameObject'lere yaptıklarıyla tamamen aynıdır.

Hadi bunu biraz daha açıklığa kavuşturalım:

Adım 1

Az önceki olayda siz bir objeydiniz, bir kişi, bir GameObject; ve içinde bulunduğuuz oyunun adı da hayat'tı. Siz “**sohbet**” scriptinizi uygularken çalan telefon sizin “**sohbet**” scriptinizi **böldü** (**interrupt**).

Adım 2

Sizden başka bir **fonksiyonu** (**function**) yapmanız **istendi** (**call**) – telefonla konuşmak. Ve şimdi siz de “**telefonla konuşmak**” scriptini uyguluyorsunuz.

Adım 3

Telefonla işiniz bittiğinde, “**sohbet**” scriptinize **geri döndünüz** (**return**).

NOT:

Kırmızılar içindeki kelimelere dikkat edin. Bu kelimeler scriptlerde kullandığınız başlıca kelimelerle neredeyse tamamen aynıdır. Zaten bildiğiniz, anladığınız, hayatınızda uyguladığınız kelimeler.

Script Fobisi



Şimdi size soruyorum, az önceki işlem ne kadar zordu? Bunları yapamayacağınızı mı düşündünüz? Ya da hiç bir foruma gidip de biriyle nasıl sohbet edileceğini, sohbetin yarısında telefon çalarsa nasıl davranışınızı sordunuz mu?

Script Fobisi: Talimat verememe korkusu (Bu terimi ben uydurdum)

Sizde olan şey de bu mu? Birkaç tane birbiriyile tutarlı davranış biçimini yazamama korkusu mu?

Asıl nokta, aslında tüm hayatını scriptliyorsunuz, sadece bir şeyi yapmadan önce aşama aşama olarak bir kağıda yazmıyorsunuz. Eğer isteseniz yazabilirsiniz de, ancak bu çok zaman alırdu ve niye böyle bir şey yapalım ki?! Ancak artık yapmalısınız. Hayır, kendi hayatınızdaki olayları değil; hayal gücünüzün derinliklerinde yaşayan kendi dünyinizdaki cisimlerin davranışlarını yazacaksınız.

3: “Yaratma Kompleksi”



Unity programına sahipsiniz çünkü bir oyun yapmak istiyorsunuz, ya da interaktif bir şey. Oluşturacağınız oyununuza pek çok akılsız objeyle (GameObject) dolduruyorsunuz ve artık “*God (Türkçe’ye çevirmek istemedim.)*” rolünü oynamalısınız. Bu GameObject’leri oluşturduğunuz, şimdi onlara bu hayal ürünü dünyانızda nasıl yaşayacaklarıyla ilgili ihtiyaçları olan her şeyi öğretmelisiniz. İşte burası davranış biçimlerini yazmanız gereken kısım, böylece GameObject’leriniz de tıpkı sizin gibi zeki olacaklar.

Öğretmen Olmak

Şimdi yapmanız gereken öğretmen rolünü üstlenmek. Tüm GameObject’lerinize, Prefab’larınıza; ne yapmaları gerekiyorsa ayrı ayrı onları öğretmelisiniz. Şansınıza, tüm ihtiyaçları olan şeyler Unity’nin tedarik ettiği Script Referans Dosyası’nda sizin için bekliyor. Tek yapmanız gereken GameObject’lerinize yapacakları davranışlarını sırasına göre yazmak.

Uygun Davranışlar

Scripting Overview

This is a short overview of how scripting inside Unity works.

Scripting inside Unity consists of [attaching custom script objects](#) called [behaviours](#) to game objects. Different functions inside the script objects are called on certain events. The most used ones being the following:

Unity’nin tasarımcıları epey zekiler. Sizin; oluşturduğunuz oyunlarınızda, GameObject’lerinize nasıl davranış göstermeleri gerektiğini öğreteceğinizi biliyorlar. Peki niçin onları zekilikle övdüm? Bunu anlamak için resimdeki kelimeye bakın. (**behaviour: Davranış**) ([Script Referans Dosyası](#))

(Cümlenin çevirisi: “**Scriptlemek, Unity için; GameObject’lere (Oyun objeleri) script adındaki çeşitli davranış (behaviour) dosyaları atamaktır.**”)

Peki sizce de bu yöntem GameObject'lere yapacağımız işlemi açıklamak için zekice düşünülmemiş midir? Onlara çeşitli davranışlar (behaviour) kazandırıyorsunuz. Tıpkı bir ebeveynin çocuğuna nasıl davranışacağını öğretmesi gibi. Unity sırf tüm bu davranışların neler olduğunu listelemekle kalmazmış gibi bir de bunları tek tek [Script Referans Dosyası](#)'nda açıklamıştır.

Bunun anlamı; yapmanız gereken bu davranış listesinden uygun davranışları seçip GameObject'inize atamaktır. Unity tüm bu davranışların sahip olduğu ağır programlama işlemini kendi halletmektedir. Yani tek yapmanız gereken istediğiniz davranışı seçip yazacağınız scriptlerde kullanmaktır.

GameObject (Oyun Objesi): Scriptler ve Componentler (Parçalar)

4: Class (Sınıf), Component (Parça) ve Script

Her şeyden önce, bu kelimelerin ne anlamına geldiğini bilmelisiniz. Sonra da bu kelimelerin nasıl kullanıldığını bilmelisiniz. İlkokuldaykenki anılarınıza dönüş yapın ve Edebiyat öğrendiğiniz zamanları hatırlayın. Ne kadar da sıkıcı ders, ancak nasıl yazı yazacağınızı bilmeniz açısından önemli. Belki de bu kelimelerin anlamını zaten biliyor ve script'le ne alakası olduğunu anlamaya çalışıyorsunuz. Ancak bu kelimeler scriptleme aşaması için oldukça önemli ve kavranması kolay terimler.

Bu aşamada bir script'in ne olduğunuyla ilgili birkaç basit açıklama yapacak ve bir script'i oluşturan iki elemanı tanıtacağım: **variable(değişken)** ve **function(fonksiyon)**. Bunları açıklamak için genel, günlük yaşamdan örnekler vererek gerçekte bu terimlerin ne kadar da basit olduğunu göstereceğim. Sonraki aşamalarda ise bu basit konseptleri nasıl kullanacağımızı öğreneceğiz.

Unity GameObject (Oyun Objesi) Referans Linki

<http://unity3d.com/support/documentation/Manual/GameObjects.html>

Class (Sınıf): Genel bir tanım



Class'lar Unity'de önemli bir role sahiptir. Şimdi bir Class nedir bunu göreceğiz ve ileride göreceğimiz diğer terimlerin Class ile olan alakalarını öğreneceğiz. Peki nedir bu Class dediğimiz şey?

Basit cevap – bu herhangi bir şeyin tanımıdır, tıpkı bir sözlükte olduğu gibi.

Mesela sözlüğü açıp da “köpek” ya da “ördek” kelimelerine bakarsanız bir köpeğin ya da ördeğin ne olduğunu öğrenirsiniz. Peki “köpek” kelimesi için sözlükteki tanım ‘köpek’ midir? Elbette hayır, Ancak sözlüğün yaptığı bir köpeğin nasıl göründüğünü açıklamak ve köpeklerin sahip olduğu temel davranışları tanımlamaktır.

Unity cevabı – şöyle bir tanımlamadır:

- 1) Veri (**data**) depolamak için kullanılan **değişkenlerin** (**variable**) tanımlandığı,
- 2) **Değişkenler** (**variable**) üzerinde işlem yapılmasını sağlayan **fonksiyonların** (**function**) tanımlandığı bir terimdir.

Başka bir deyişle, bir **Class** (**Sınıf**) şunları tanımlamak için kullanılır:

- 1) Birkaç kelime – **değişken** (**variable**) adları; ve
- 2) Bunların yapabildiği bir şeyler – **fonksiyonlar** (**function**)

Component'ler (Parça) ve Scriptler: Biraz sihire bulaşalım :)



Unity ile çalışırken iki tür şapka giyersiniz:

- 1) Bir **Oyun Tasarımcısı** (**Game-Creator**) şapkası; veya
- 2) Bir **Scriptleme** (programcı) şapkası.

Oyun Tasarımcısı şapkanızı giydiğinizde; bir **GameObject**'i (Oyun Objesi) seçebilir ve bu objeye atanmış herhangi bir **Component**'in (Parça) özelliklerini (Properties) inceleyebilirsiniz.

Ne zaman ki **Scriptleme** şapkanızı giydiniz, o zaman işin boyutu değişir:

- Bir script; bir GameObject'e çeşitli şeyler yaptırmak için yazılır.
- Bir script kaydedildiğinde, artık o scriptin adı bir **Class (Sınıf)** adı olur – bir takım **değişkenlerin (variable)** ve **fonsiyonların (function)** depolandığı.
- Bu **Class** (script dosyası) **Project (Proje)** dosyanızda depolanır, burada boş boş hareketsiz bir şekilde kullanılmak için bekler.
- Şimdi **Scriptleme** şapkanızı çıkarın.

Ardından tekrar **Oyun Tasarımcısı** şapkanızı giydiğinizde ve **Class'ınızı** (script dosyası) bir **GameObject**'e (Oyun Objesi) atadığınızda sihir etkisini gösterir;

- Sihirli değneği sallayın vee – **ZINN** - artık **Class (Sınıf)** (script dosyası) objeniz bir **Component (Parça)** olarak tanımlanmakta ve **Sınıfin (Class)**; **public (genel) değişkenleri (variable)** artık **Inspector**'da (UNITY arayüzündeki, sağdaki parça) göreceğiniz özelliklere (**Properties**) dönüşecektir.

5: Scriptlerde Değişkenler (Variable)

Bir **değişken** (**variable**) nedir? Teknik olarak bu, bilgisayarınızın hafızasının; bir bilgiyi barındıran minicik bir parçasıdır. Bir oyun çalışırken bu değişken bilginin (information) depolandığı yeri, miktarını ve türünü sürekli gözetimi altında tutar. Sizin tek bilmeniz gerekense bir scriptte bir **değişkenin** (**variable**) nasıl çalıştığını bilmektir ki bu da oldukça kolaydır.

Bir değişken bir posta kutusuna benzer



Bir posta kutusunun içinde hava haricinde genelde ne olur? Aslında genelde içinde bir şey olmaz, boş olur. Ancak zaman zaman içinde bir şeyler olabilir. Mesela para (maaş çeki), halanızdan bir mektup, bir örümcek vb. Asıl nokta, bir posta kutusunun içindeki şeyin farklı farklı olabilmesi, değişken olmasıdır. Peki, biz de şimdilik bu posta kutularını birer **değişken** (**variable**) olarak tanımlayalım.

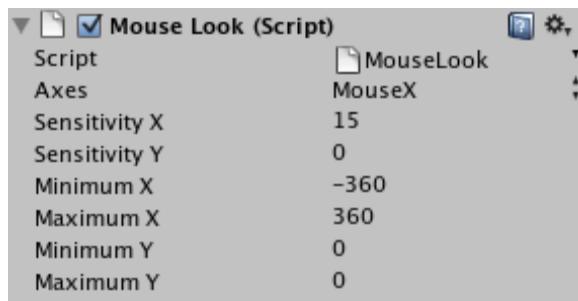
Bir değişkenin adı

Size yukarıdaki resmi gösterip de posta kutusunun içerisinde ne olduğunu sorsam heralde ilk söyleyeceğiniz şey “Hangisi?” olurdu. Eğer Ali'nin posta kutusu, kahverengi posta kutusu veya yuvarlak posta kutusu desem tam olarak hangisini kastettiğimi anlayabilirdiniz.

Scriptlerde **değişkenlerle** (**variable**) ilgili yapmamız gereken bir başka şeye o değişkeni adlandırmaktır. Böylece örneğin **benimNumaram** isimli değişkende (Veya sizin uygun gördüğünüz başka bir isim) ne depolandığını bulabiliriz.

Biraz daha teknik konuşursak, değişkenler; bilgisayarınızdaki birer hafıza adresleridirler (**Memory address**) (Türkçe karşılığını tam olarak bilmiyorum.). Ne zaman ki bir değişken kullanırsak bilgisayar tam olarak hafızasının (Memory) hangi kısmına bakarak veriyi bulacağını, depolayacağını bilir.

Değişkenlerin (**Variable**) tek yaptığı bir verinin yerine geçmektir



Haa??? Ne???

Resmin sol tarafındaki isimlere bakın: Sensitivity (Mouse Hassaslığı) X, Sensitivity Y, vb.? Bunlar bir script'teki **değişkenlerin** (**variable**) isimleridirler. Sağdaki sayıları görüyor musunuz? İşte onlar da bu **değişkenlerin** (posta kutusu) içerisinde depolanan veridirler. Peki bu şey nasıl çalışır?

Sizin **myNumber** isminde bir **değişkeninizin** (posta kutusu) olduğunu varsayıyalım. Bu **myNumber** **değişkeninin** (**variable**) içerisinde ise 9 rakamı depolanmakta (yani posta kutusunun içinde 9 rakamı var.). Şu an bu sayı bir şey ifade etmiyor olabilir, tabi işin içine okulda gördüğünüz cebirsel matematiği koyarsanız işler değişir:

$$2 + \text{myNumber} = \text{Haa}???$$

Normalde rakamlarla yazıları toplayıp da sonuç elde edemezseniz. Ancak bu yazı bir script'in içindeki bir **değişkense** işler değişir. Çünkü **değişkenin** içerisinde ne depolanmışsa o değer; ilgili rakamla toplanır. Yani bu basit toplama işleminde, 9 rakamı, depolandığı **myNumber** isimli değişken nerede kullanılırsa oraya yazılır, yani:

$$2 + 9 = 11$$

Uzun lafın kısası, script yazarken tanımladığınız **değişkenler** (**variable**) basit anlamda; onlara atanınan değerler için birer yer tutucudurlar.

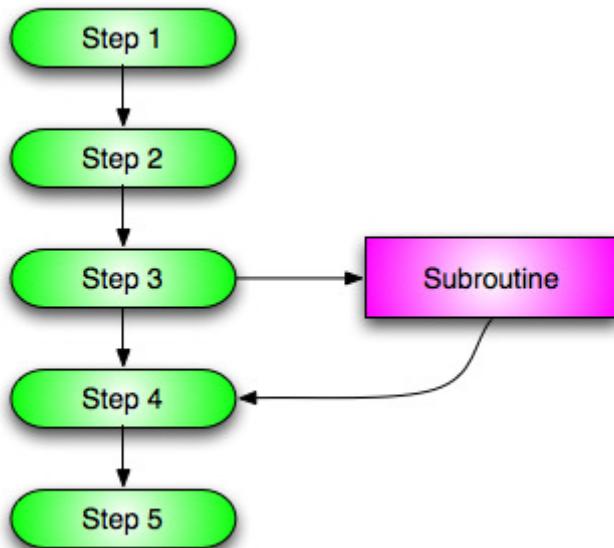
Daha Sonra

Böylece basit anlamda değişkenleri görmüş olduk. Onlar bir scriptte, istediğiniz herhangi bir veriyi depolamak için kullanılır. **Değişkenler** birden çok türde veri depolayabilirler, örneğin yazılar, sayılar ya da bir **GameObject**'nın (**Oyun Objesi**) kendisi gibi. Değişkenlerle ilgili **değişken söz dizimi** (**variable syntax**) ve bir değişkene veri atamak ve o veriyi kullanmak gibi konuları ise daha sonra öğreneceğiz.

6: Scriptlerde Fonksiyonlar (Function)

Fonksiyonlar asıl eylemlerin uygulandığı yerdir. Kararlar alınır, işlemler yapılır. Bir scriptin çalışan kısımlarıdır. Bir arkadaşımızla sohbet ederken araya telefonun girdiği örneğimizde; arkadaşımızla “sohbet etmek” fonksiyonunu gerçekleştiriyorduk ancak sonrasında, başka bir fonksiyon olan “telefonla konuşmak” fonksiyonuna geçiş yapmak zorunda kaldık.

Adım, adım, kaçınmak, adım...



Bir script çalıştırılırken her satır kod düzenli olarak yerine getirilir (işlenir).

Script çalışırken bazı engellemeler çıkabilir (**interruption**), arkadaşımızla sohbet ederken telefona bakmak zorunda kalmamız gibi. “Arkadaşımızla sohbet etmek” **fonksiyonunu** (**function**) gerçekleştiriyorduk ki bir **engelleme** (**interruption**) çıktı, telefonun çalması. Artık başka bir fonksiyonu çalıştmak zorunda kaldık. Bu fonksiyonun da kendine has adımları (davranışları) vardır. Bu engelleyici fonksiyonu tamamladıktan sonra ise “arkadaşla sohbet etme” fonksiyonumuza **kaldığımız yerden** devam ettik.

Fonksiyon demek mantık demektir

Oyun objeleri (**GameObject**) pek çok **girdi** (**Input**) alabilirler, bir şeyle çarpışmak (temas etmek) (**collision**) veya kullanıcının bir tuşla girdi sağlama bunlara sadece basit iki örnek. Bu girdiler belirli bir **fonksiyonu**, çeşitli eylemleri uygulaması için iş başına çağırırlar. Bu fonksiyonları da scriptlerinizde oluşturursunuz.

Bir fonksiyonun oluşturulma sebebi defalarca kullanılmasından dolayıdır. Yani aynı işlemleri tek tek her seferinde yazmak yerine o işlemleri tek bir taşıyıcının (**container**) içine yerleştiririz ve bu taşıyıcıya bir isim veririz. Böylece bir fonksiyonumuz olmuş olur. Bu fonksiyonu bir yerde kullanmamız gerekince de orada fonksiyonun ismini yazarız ve böylece fonksiyon ilgili kısımda işini yapar.

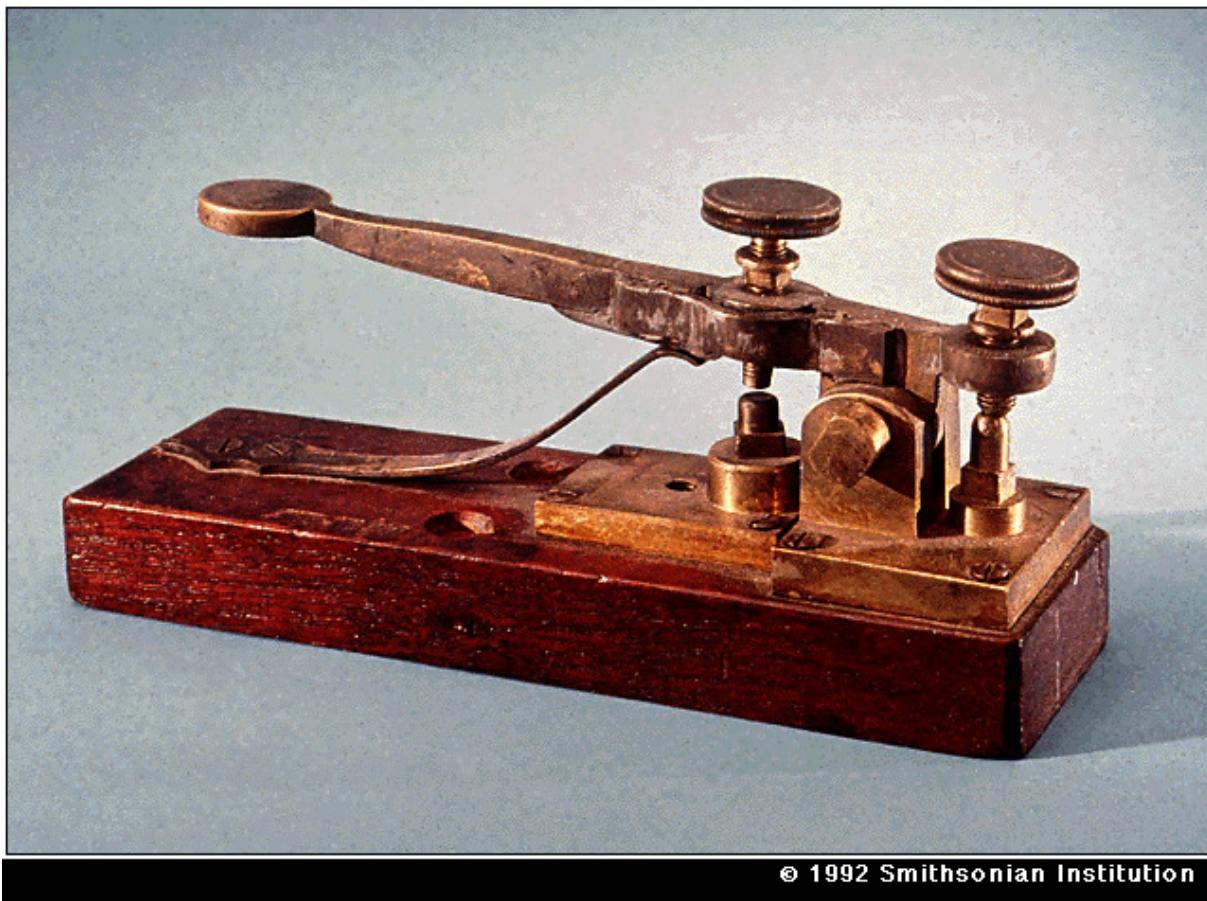
Daha Sonra

İlerleyen zamanlarda bir fonksiyonu nasıl çağrıracığınızı göreceksiniz. Şimdilik bilmeniz gereken, bir fonksiyonun belli bir işlem kümesi olduğudur ve fonksiyonların oluşturulmalarının sebebi ise fonksiyona yazılan işlemin defalarca kullanılacak olmasıdır. Yani bir dizi işlemi kendi taşıyıcısına koyup o taşıyıcıya bir isim vererek fonksiyonları oluşturmak epey akıllıcadır, sonradan da bu fonksiyonu; ismiyle dilediğimiz yerde çağırabiliriz.

7: Scriptlerde Noktasal Yapı (Dot Syntax): İletişim Kurmak

Scriptimiz bir veriyi depolayan **değişkenlere** (**variable**) ve bir işlemi yapmaya yarayan **fonsiyonlara** (**function**) sahip. Şimdi sizlere bir **GameObject**'teki başka **Component**'lerle (**Parça**) iletişim kurmayı göstereceğim. Bu konuya sadece burada değineceğim, çünkü **Component**'ler arası iletişim de scriptlemenin bir parçasıdır. Bunun sayesinde etkileşimler mümkün olur.

Noktalarla işimiz ne?



Başkaları tarafından yazılmış kodlara bakarsanız, birbirlerinden noktalarla ayrılan çeşitli kelimelerin birer küme oluşturduklarını görürsünüz. Bu da nesi? Oldukça da karmaşık duruyor, değil mi?

Unity'nin kendi dökümanından basit bir örnek vereyim:

transform.position.x

Yukarıdaki kodun ne işe yaradığıyla ilgili kafanızı yormayın, bunu sonradan öğreneceksiniz. Buradaki amaç noktasal yapıyı göstermekti.

İşte bu yapıya **Noktasal Yapı** (**Dot Syntax**) denir.

UnityScript'in de uyulması gereken çeşitli gramer kuralları bulunmaktadır. İşte size başka bir örnek: Evimin adresi:

USA.Vermont.Essex.22 benimSokagim

Eğlenceli duruyor, değil mi? Çünkü UnityScript'in **Dot Syntax** gramerini kullandım, posta ofislerinde kullanılan grameri değil. Ancak bahse girerim biraz yakından bakarsanız pekâlâ evimi rahatça bulabilirsiniz.

Başka bir örnek daha. Sizinle Dünya üzerinde rasgele bir yerde buluşsaydık ve sizden gözlüğümü almanızı isteyip şu bilgiyi verseydim:

USA.Vermont.Essex.22 benimSokagim.2nci kat.ofis.masa.orta cekmece.gozluk

Herhalde gözlüğümü bulurken herhangi sorun yaşamazsınız, değil mi?

Yani bu bir adres (Address) mı?

Evet, hepsi bu. **Noktasal Yapı** (**Dot Syntax**) basit anlamda bir **GameObject**'i, **Component**'ı (**Parça**), **değişkeni** (**variable**) veya bir **fonksiyonu** (**function**) tanımlamak için kullanılan adrestir.

8: Sonuç

Sonraki Adım

Bu öğrendiğiniz temel kavramlar UnityScript'i anlamanız açısından ileride epey işinize yarayacak.

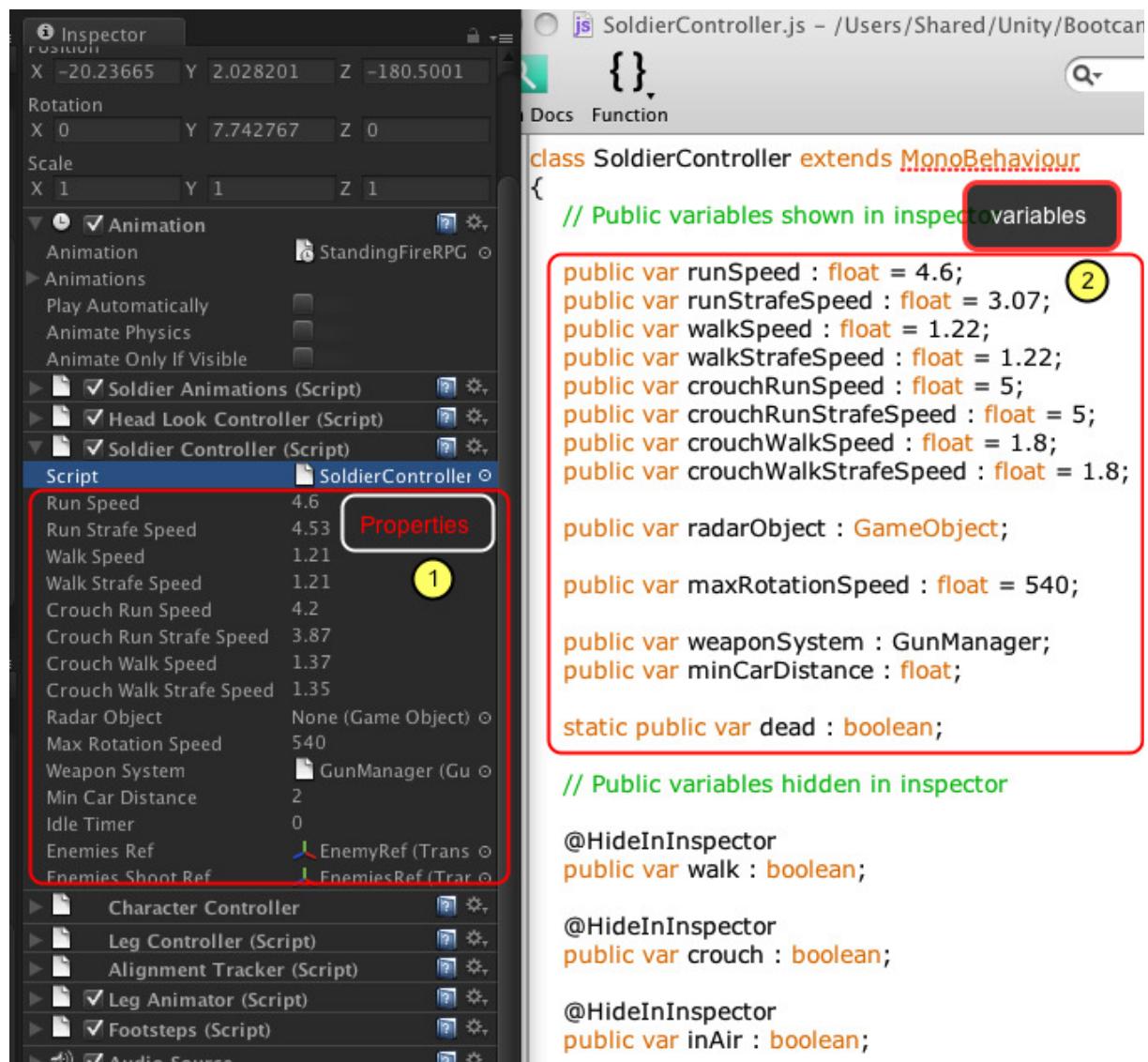
Artık değişkenleri, fonksiyonları ve Dot-Syntax yapısını öğrendiğinize göre gerçek bir script oluşturmanın detayları için sonraki bölümme geçebilirsiniz.

Değişkenlere Detaylı Bakış

9: Değişkenler (Variable) ve Component (Parça) Özellikleri

Oluşturacağınız **Oyun Objeleri** (**GameObject**), onları işlevli kıلان çeşitli özelliklere (**Properties**) sahip olacaklardır. Bir GameObject'e Component eklediğinizde o objenin yapısını değiştirmiş olursunuz. Bu nedenle bir **Component**'in **GameObject**'e eklediği bilgi, **Component**'ten bağımsız bir yerde depolanmalıdır, bu yerse **değişkenlerdir**.

Değişkenin Component Özelliği (Property) Oluşu



Bu resim Soldier Controller (Asker Kontrolörü) isimli scriptin bir parçasıdır. (SoldierController.js)

1. Sol tarafta Unity'nin Inspector panelinde, bu scriptin **Component** (**Parça**) özellikleri gösterilmektedir.
2. Sağda ise Unitron editöründe scriptin kendisi gösterilmektedir.

Bir değişken adı küçük harfle başlar

The screenshot shows the Unity Editor interface. On the left is the Inspector window for a GameObject named 'SoldierController'. It lists various components and their settings. On the right is the code editor for the script 'SoldierController.js'.

```
js SoldierController.js - /Users/Shared/Unity/Bootcamp/Assets/Scripts/SoldierController.js
{
    // Public variables shown in inspector

    public var runSpeed : float = 4.6;
    public var runStrafeSpeed : float = 3.07;
    public var walkSpeed : float = 1.22;
    public var walkStrafeSpeed : float = 1.22;
    public var crouchRunSpeed : float = 5;
    public var crouchRunStrafeSpeed : float = 5;
    public var crouchWalkSpeed : float = 1.8;
    public var crouchWalkStrafeSpeed : float = 1.8;

    public var radarObject : GameObject;
    public var maxRotationSpeed : float = 540;
    public var weaponSystem : GunManager;
    public var minCarDistance : float;

    static public var dead : boolean;

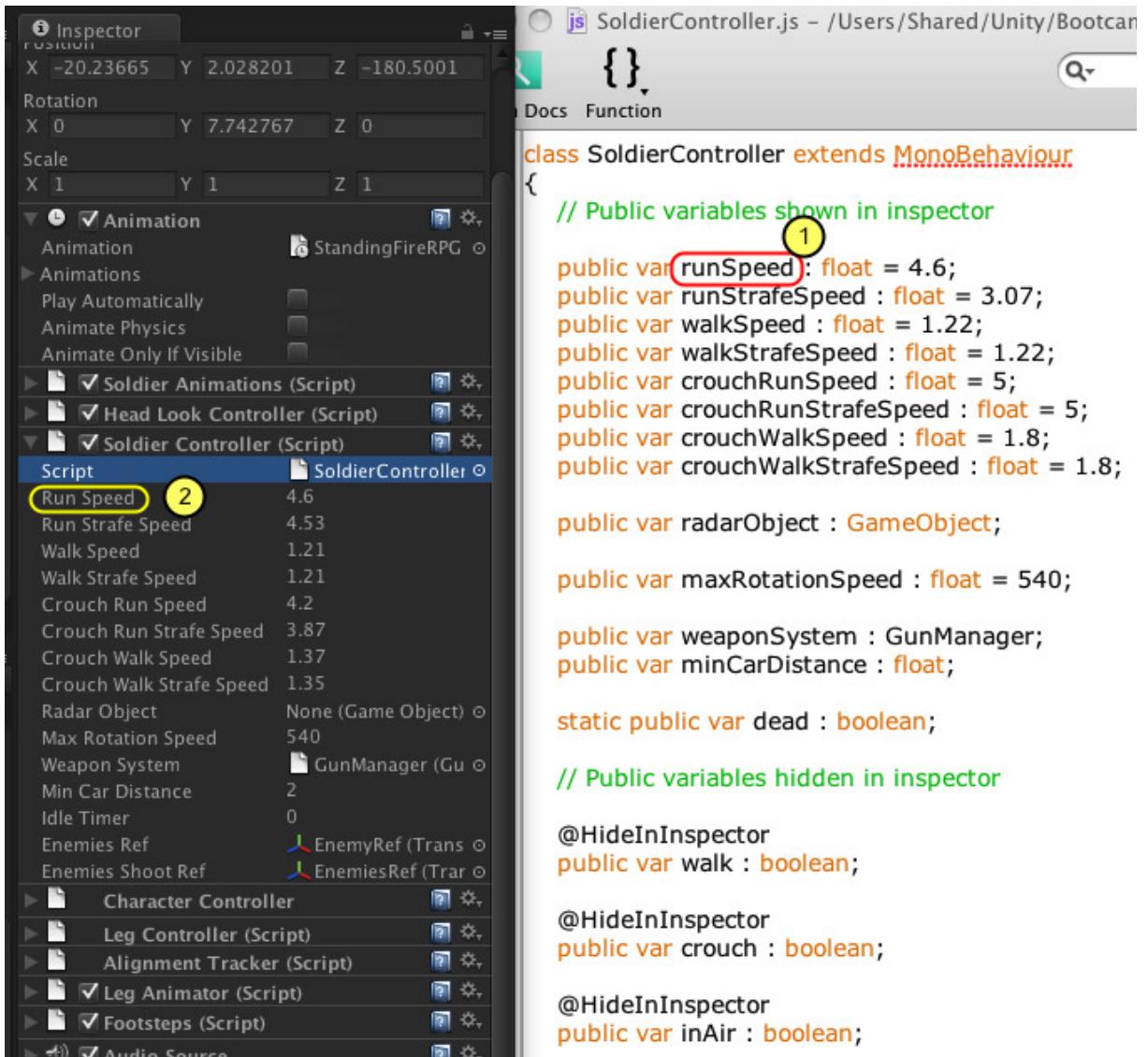
    // Public variables hidden in inspector

    @HideInInspector
    public var walk : boolean;
    @HideInInspector
    public var crouch : boolean;
    @HideInInspector
    public var inAir : boolean;
}
```

The code editor shows several variable declarations with small red circles around the variable names 'runSpeed', 'runStrafeSpeed', 'walkSpeed', 'walkStrafeSpeed', 'crouchRunSpeed', 'crouchRunStrafeSpeed', 'crouchWalkSpeed', 'crouchWalkStrafeSpeed', 'radarObject', 'maxRotationSpeed', 'weaponSystem', 'minCarDistance', 'dead', 'walk', 'crouch', and 'inAir'. These circled names correspond to the public variables listed in the Inspector window.

Gördüğünüz gibi, her değişkenin adı küçük harfle başlamaktadır. Bu bir kuraldır.

Çok kelimeli değişken adları



1. **Variable (Değişken)** isimleri sadece bir kelimedenden ibaret olabilir, boşluk karakteri kesinlikle kullanılamaz. Ancak bir değişken adını birden çok kelimeli yapmak için sonraki kelimelerin baş harflerini büyük yazarsınız, bu örnekteki **runSpeed** (kosuhizi) gibi. Böylece değişken adında büyük harfle başlayan her parça yeni bir kelime olarak algılanır. Üstteki resimde de bunu görebilirsiniz.

2. Unity basit bir sihir uygular ve **runSpeed** değişken (**variable**) ismini **Run Speed** özellik (**Property**) ismine çevirir.

3. (*Çevirmen Eklemesi*): *Bir değişken adında kesinlikle ama kesinlikle Türkçe karakter bulunduramazsınız, eğer bulundurursanız o değişkenle işlem yaparken hata alırsınız. Örneğin “Şen Pirasalar” isminde bir değişken oluşturmak isterseniz onu ancak şöyle isimlendirebilirsiniz: “senPirasalar”*

```

Inspector
Position X -20.23665 Y 2.028201 Z -180.5001
Rotation X 0 Y 7.742767 Z 0
Scale X 1 Y 1 Z 1

Animation
Animation StandingFireRPG
Animations
Play Automatically
Animate Physics
Animate Only If Visible
Soldier Animations (Script)
Head Look Controller (Script)
Soldier Controller (Script)
Script SoldierController
Run Speed 4.6
Run Strafe Speed 4.53 (2)
Walk Speed 1.21
Walk Strafe Speed 1.21
Crouch Run Speed 4.2
Crouch Run Strafe Speed 3.87
Crouch Walk Speed 1.37
Crouch Walk Strafe Speed 1.35
Radar Object None (Game Object)
Max Rotation Speed 540
Weapon System GunManager (Gu)
Min Car Distance 2
Idle Timer 0
Enemies Ref EnemyRef (Trans)
Enemies Shoot Ref EnemiesRef (Trar)
Character Controller
Leg Controller (Script)
Alignment Tracker (Script)
Leg Animator (Script)
Footsteps (Script)
Audio Source

SoldierController.js – /Users/Shared/Unity/Bootcar
{
    // Public variables shown in inspector

    public var runSpeed : float = 4.6;
    public var runStrafeSpeed : float = 3.07; (1)
    public var walkSpeed : float = 1.22;
    public var walkStrafeSpeed : float = 1.22;
    public var crouchRunSpeed : float = 5;
    public var crouchRunStrafeSpeed : float = 5;
    public var crouchWalkSpeed : float = 1.8;
    public var crouchWalkStrafeSpeed : float = 1.8;

    public var radarObject : GameObject;

    public var maxRotationSpeed : float = 540;

    public var weaponSystem : GunManager;
    public var minCarDistance : float;

    static public var dead : boolean;

    // Public variables hidden in inspector

    @HideInInspector
    public var walk : boolean;

    @HideInInspector
    public var crouch : boolean;

    @HideInInspector
    public var inAir : boolean;
}

```

1. Scriptte değişkenin değeri 3.07

2. Inspector panelinde ise değeri 4.53

Burada bir yanlışlık mı var? Hayır. Bir script oluştururken tanımladığınız bir değişkene atadığınız değer **varsayılan (default)** olarak adlandırılırlar.

Bu değeri Inspector panelinde istediğiniz gibi değiştirmek ilgili GameObject'in istediğiniz gibi davranışını sağlayabilirsiniz. Ayrıca yaptığınız bu değişiklik script üzerinde herhangi bir değişiklik yapmanızda sebep olmaz.

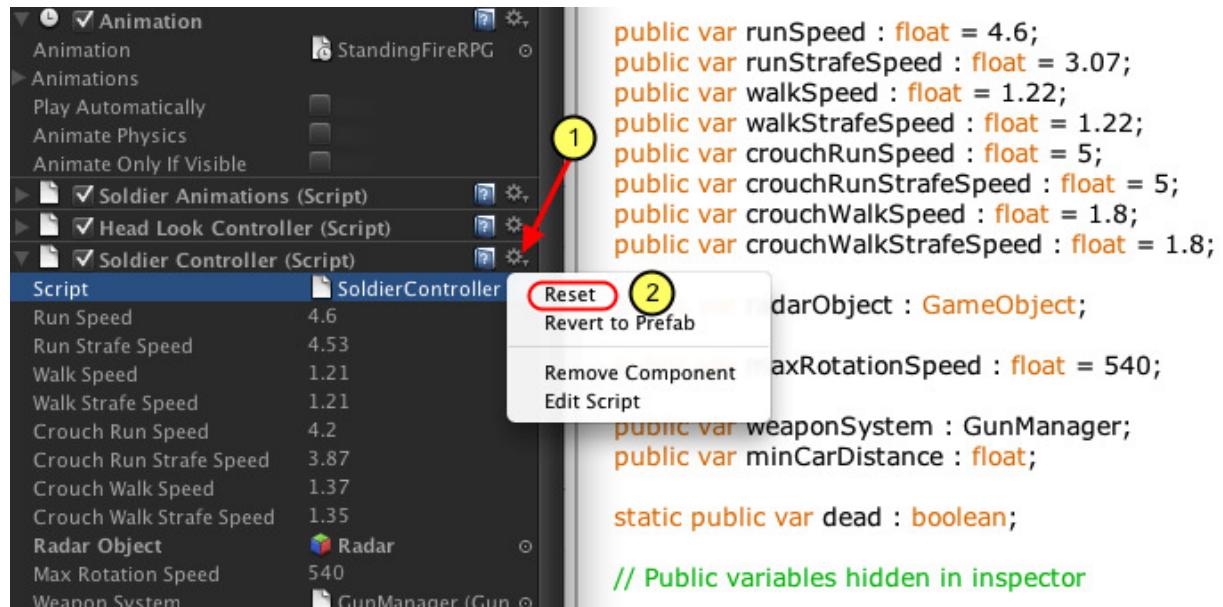
Siz bir değişkenin değerini Inspector'dan değiştirdiğinizde yaptığınız şey o değişkenin değerinin **üzerine yazmak (override)**'dır. Oyun boyunca değişkenin değeri, Inspector'da değiştirdiğiniz değer olarak alınır, script'teki değer ise bu işlem sırasında değişikliğe uğramaz.

Not: Eğer **Play Mode** (Oyun Modu) (Oyunu test ederken kullandığınız mod) sırasında Inspector üzerinde bir değişkenin değerini değiştirisreniz, test aşamasından çıkışınca yaptığınız değişiklik kaybolur. Bu yüzden eğer test

aşamasında bir değişken değerinde oynama yaparsanız, test aşamasından çıkmadan önce yaptığınız değişiklikleri bir kenara not edin.

Şunu tekrarlamakta fayda var: Inspector'da bir ayarı değiştirmek script'in kendisine herhangi etkide bulunmaz, script aynen yazdığınız gibi kalmaya devam eder.

Inspector'daki değerleri resetleyerek orijinal scriptteki değere döndürme



Inspector'daki değerleri orijinal scriptteki değerlere döndürmek (resetlemek) için şu yolu izleyebilirsiniz:

1. Component'in sağındaki dişli simgesine tıklayın;

2. Reset'e tıklayın.

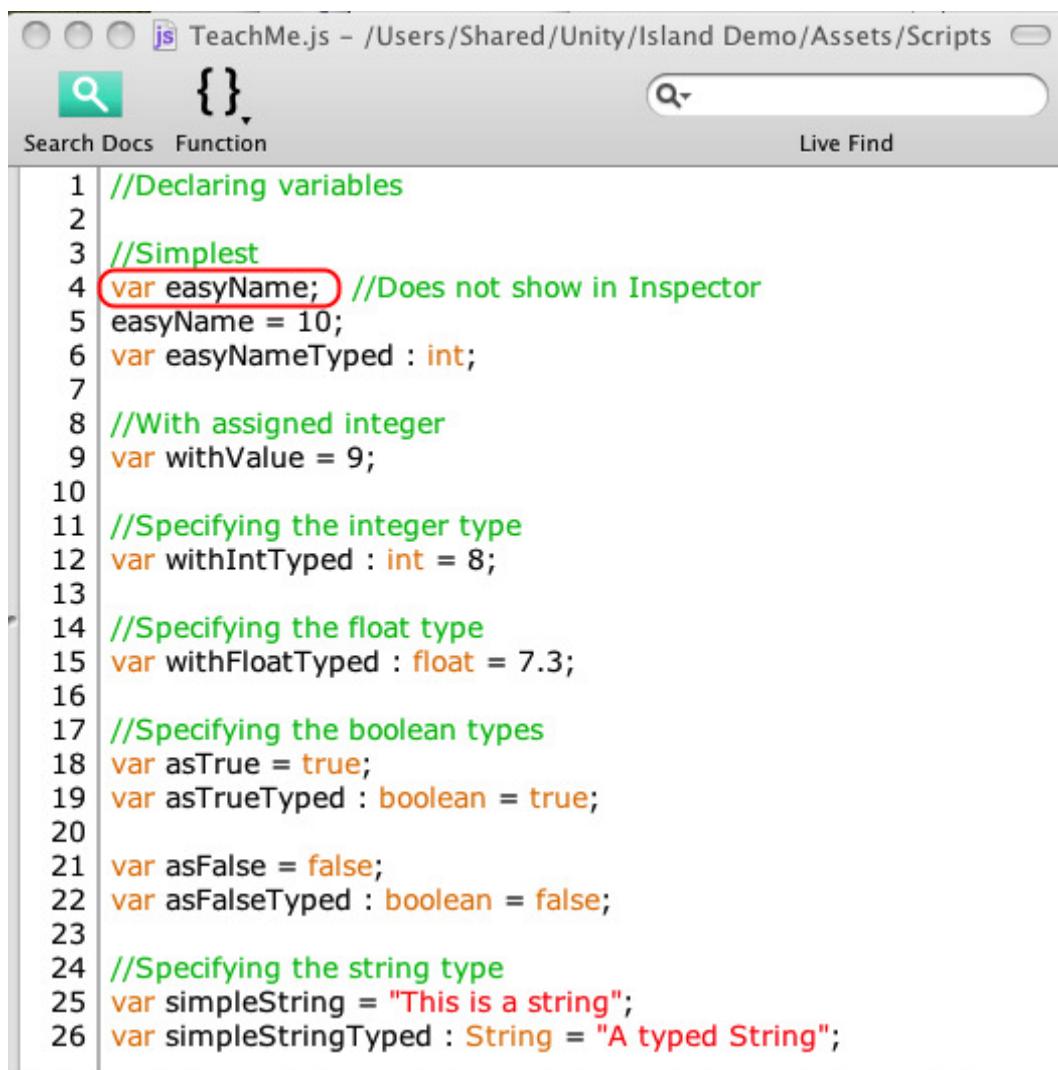
Bu Inspector'daki değerleri, scriptteki değişkenlerin orijinal değerlerine geri çevirecektir.

10: Değişken (Variable) isimleri ve "çeşitleri"

Daha önceden posta kutusuörneğinde açıkladığım gibi bir posta kutusu içerisinde yer alan şeyin türü değişimdir (para, mektup vb.), **değişkenler** ayrıca farklı türlerde verileri de depolayabilirler. Basit bir sayıdan koca bir GameObject'e veya Prefab'a kadar pek çok şeyi depolayabilirler.

Scriptlerinize değişkenler eklemek için uymanız gereken basit bir gramer vardır.

Bir Scriptte değişken tanımlamak



```
1 //Declaring variables
2
3 //Simplest
4 var easyName; //Does not show in Inspector
5 easyName = 10;
6 var easyNameTyped : int;
7
8 //With assigned integer
9 var withValue = 9;
10
11 //Specifying the integer type
12 var withIntTyped : int = 8;
13
14 //Specifying the float type
15 var withFloatTyped : float = 7.3;
16
17 //Specifying the boolean types
18 var asTrue = true;
19 var asTrueTyped : boolean = true;
20
21 var asFalse = false;
22 var asFalseTyped : boolean = false;
23
24 //Specifying the string type
25 var simpleString = "This is a string";
26 var simpleStringTyped : String = "A typed String";
```

Bir scriptte kullanılacak tüm **değişkenler (variable)** öncelikle **tanımlanmalıdır** (**declare**). Peki bunun anlamı nedir? Siz Unity'e ne oluşturmak istediğiniz söylersiniz, böylece Unity oluşturduğunuz bu değişkeni bir scriptte kullanınca neyi kullandığını bilir. Üstteki resimde 4. satırı bakın.

Burada **easyName** (basitlsim) adında bir değişken oluşturduk. Bu ismin hemen başında ise **var** takısı var (**var = variable = değişken**), bu da variable'nin kısaltılmış hâlidir. Değişkenin isminin başına **var** takısı koymak Unity'e yeni bir değişken oluşturmakta olduğumuzu söyler.

Yani, Unity'de scriptlerimizde bir değişken kullanmadan önce Unity'e bu değişkeni tanıtmalıyız (Değişkeni tanımlamalıyız.).

Unity'e bir **değişkeni** (**variable**) tanıtmak işlemine **tanımlama** (**declare**) denir ve bu işlemi her değişken için sadece 1 kere yapmanız yeterlidir. Artık **easyName** değişkenimizi scriptimizin içerisinde istediğimiz yerde kullanabiliriz.

Bir **tanımlamanın** ";" (**Noktalı Virgül**) işaretiley son bulduğunu fark ettiniz mi? Unity'de tanımlamalar da dâhil her işlemin (**statement**) sonuna "**noktalı virgül**" işaretini koymalısınız. Bunun işlevi ise Unity'e o işlemin sona erdiğini söyleyip sonraki işleme geçtiğimizi belirtmektir. Yoksa Unity bunu kendi başına bilemez. Bunu bir yazıda her cümlein sonuna koyulan "**nokta**" işaretini gibi de düşünebilirsiniz. Böylece o yazıyı okurken cümlelerin bitip bitmediğini anlayabiliriz, değil mi?! Hiç bir cümleyi noktasız okumayı denediniz mi? Biraz garip olur, haksız mıym? Aslında zeki birer canlı olduğumuz için bir yazıyı noktasız okumayı da söylebiliriz ancak Unity; bir işlemin nerede bittiğini söylemezseniz bunu kendi bulacak kadar zeki değildir. Unity sadece bir yazılımdır.

Unutmamanız gereken çok önemli bir noktaya değineyim, şimdi söyleyeceğim şey büyük ihtimalle sizi epey şaşırtacaktır. Yukarıdaki resimdeki tüm bu kodların arasında birkaç tane "=" (**Eşittir**) işaretini görüyorsunuz, değil mi? Programlamada (**scriptleme**) bunun anlamı "**Eşittir**" değildir. Bu bir **atama operatöründür** (**assignment operator**). Yani şöyle bir şey gördüğünüzde;

`easyName = 10;`

Bunun anlamı **easyName** değişkeninin 10'a eşit olduğu değildir. Bunun anlamı **easyName** **değişkenine 10 değerinin atandığıdır**.

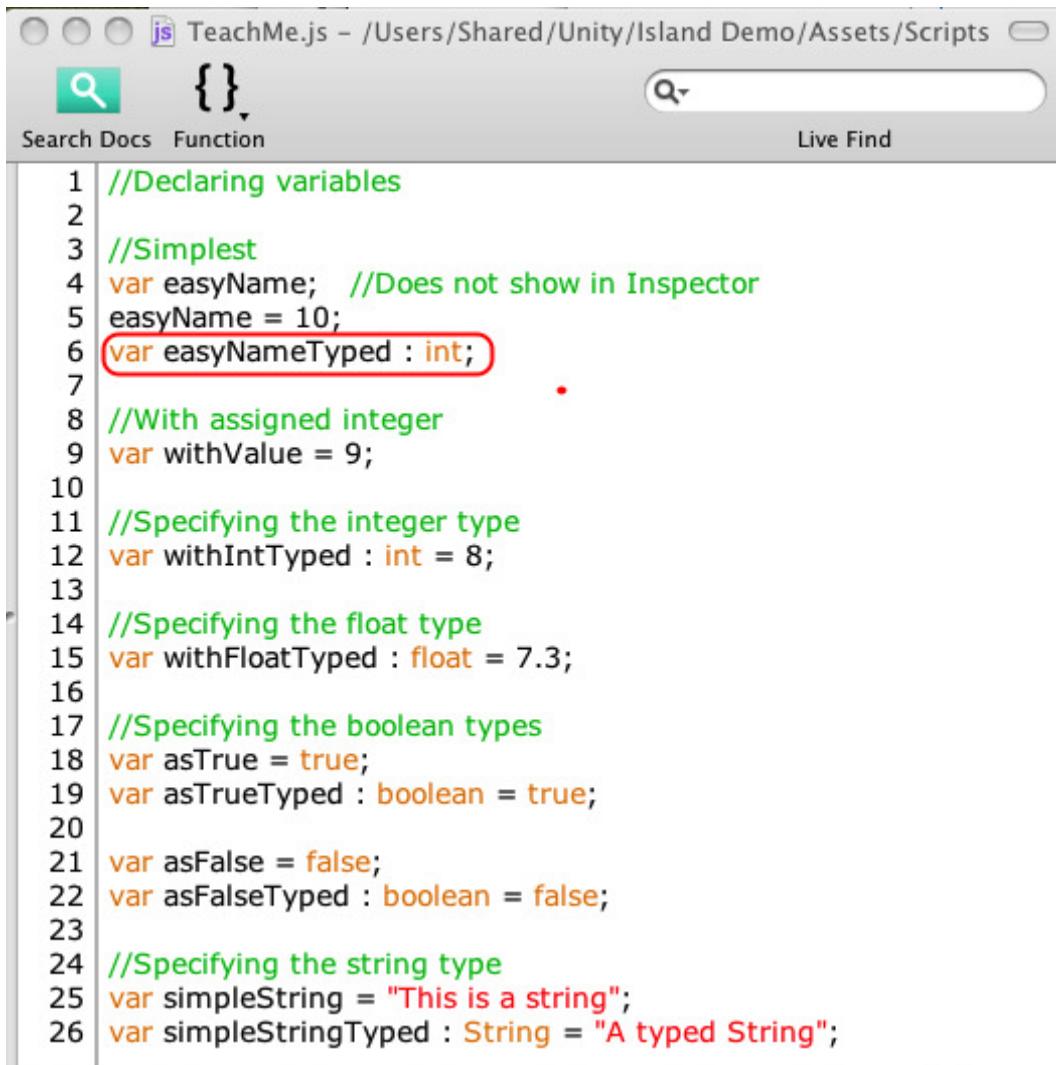
Eşitlik konusunda daha sonra konuşuruz.

Yaygın Değişken (Variable) Türleri

TÜR	DEĞİŞKENİN İÇERİĞİ
int	Bir tamsayı; mesela 3 veya -10 gibi
float	Kesirli bir sayı; 3,14 veya -6,8 gibi
String	Tırnak işaretleri içinde bir yazı; "Sonra görüşürüz" gibi
boolean	2 değer alır: true (doğru) veya false (yanlış)

Burada gördükleriniz tüm programlama dillerindeki en yaygın değişken türleridir.

Değişkenin ‘Tür’ü



```
1 //Declaring variables
2
3 //Simplest
4 var easyName; //Does not show in Inspector
5 easyName = 10;
6 var easyNameTyped : int;
7
8 //With assigned integer
9 var withValue = 9;
10
11 //Specifying the integer type
12 var withIntTyped : int = 8;
13
14 //Specifying the float type
15 var withFloatTyped : float = 7.3;
16
17 //Specifying the boolean types
18 var asTrue = true;
19 var asTrueTyped : boolean = true;
20
21 var asFalse = false;
22 var asFalseTyped : boolean = false;
23
24 //Specifying the string type
25 var simpleString = "This is a string";
26 var simpleStringTyped : String = "A typed String";
```

6. satıra bakın. Bu **değişken tanımlaması** (**variable declaration**) işleminde (**statement**) ekstradan bir ekleme var. Bu ‘int’ de neyin nesi? (**int = integer = tamsayı**)

Bunun anlamı ‘**integer**’(**tamsayı**) dır.

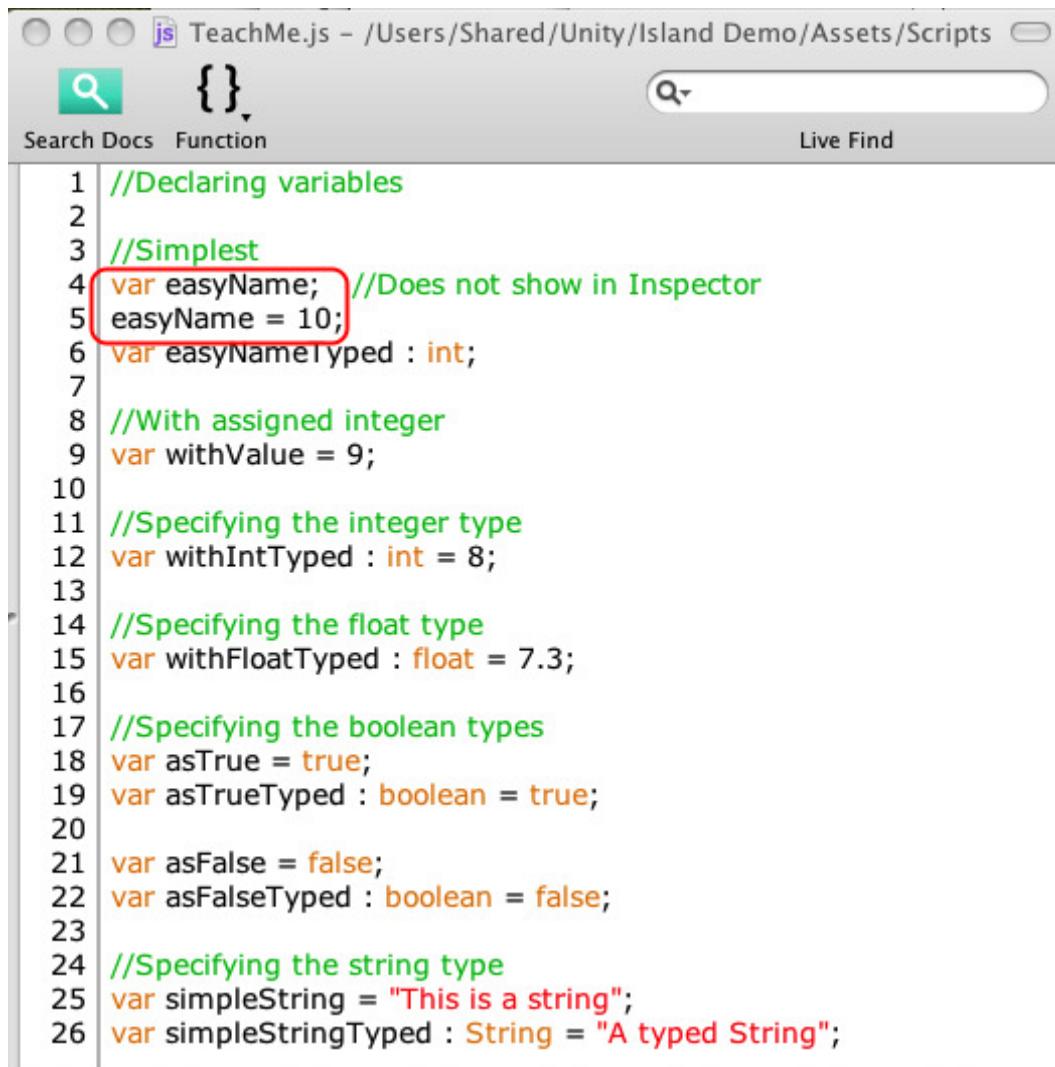
easyNameTyped (basitİsimTuru) adında bir **değişken** (**variable**) oluşturдум, ardından “:” (**iki nokta**) işaretini koydum ve sonra da “int” takısı ekledim.

UnityScript’ta bir **değişkenin** belli bir türde (**type**) veriyi depolayacağını böyle belirtirsiniz. Değişken isminin ardından “**iki nokta**” işaretini koyarsınız ve ardından değişkenin türünün ismini girersiniz. Bu örnekte Unity’ye değişkenimizin türünü özellikle söylemiş olduk, bir “**integer**”, yani **tamsayı**.

Peki, tahminimce ilk başta yaptığımız **easyName** ismindeki **değişken (variable)** tanımlamasında neden **türü (type)** belirtmediğimizi merak etmiş olmalısınız. UnityScript biraz uyuşuk olmanızı izin vermektedir ve diğer programlama dillerinde olan bazı zorunlulukları uygulamanızı zorunlu kılmaz.

UnityScript ile scriptleme yaparken Unity genelde, oluşturduğunuz değişkene atadığınız ilk değere göre o **değişkenin** türünü kendisi tahmin edebilir. Okumaya devam edin.

Türün tahmin edilmesi



```
1 //Declaring variables
2
3 //Simplest
4 var easyName; //Does not show in Inspector
5 easyName = 10;
6 var easyNameTyped : int;
7
8 //With assigned integer
9 var withValue = 9;
10
11 //Specifying the integer type
12 var withIntTyped : int = 8;
13
14 //Specifying the float type
15 var withFloatTyped : float = 7.3;
16
17 //Specifying the boolean types
18 var asTrue = true;
19 var asTrueTyped : boolean = true;
20
21 var asFalse = false;
22 var asFalseTyped : boolean = false;
23
24 //Specifying the string type
25 var simpleString = "This is a string";
26 var simpleStringTyped : String = "A typed String";
```

4. ve 5. satırlara geri dönelim.

easyName değişkenini tanımlarken **türünü (type)** belirtmedim ve bu şu anda Unity'nin hiç umurunda değil; çünkü henüz o değişkeni herhangi bir yerde kullanmadık. Sadece Unity'e **easyName** isminde bir **değişken (variable)** tanımladığımızı söylediğim. 5. satırda ise ilk defa **easyName** değişkenini kullandım ve ona bir tamsayı olan 10 sayısını atadım. Artık Unity kendisi **easyName** değişkeninin türünü tahmin ederek buldu, bu bir "**int**" (**integer**).

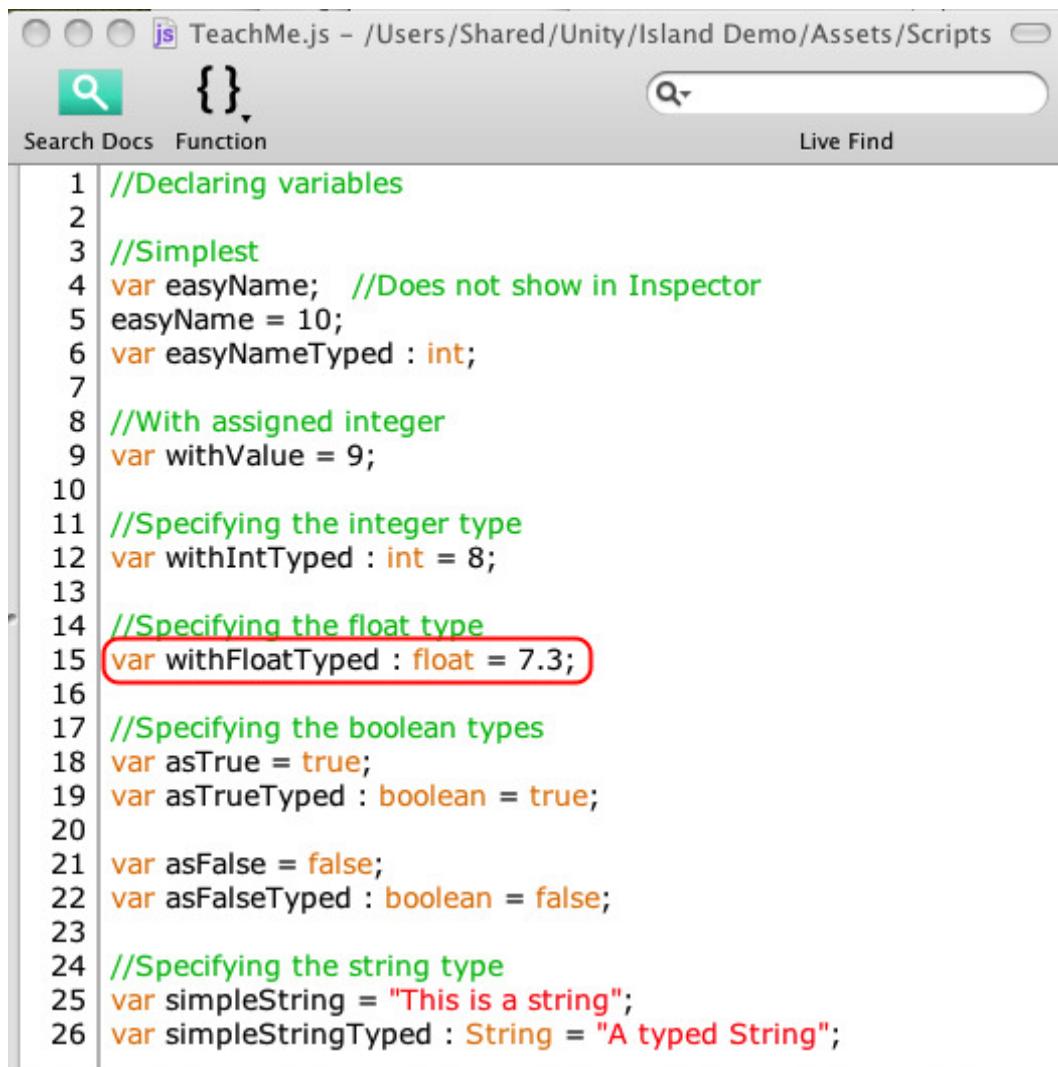
Türün tahmin edilmesi işte böyle işliyor. Değişkene ilk olarak nasıl bir veri atandıysa ona göre tür tahmin yoluyla otomatik olarak belirleniyor. Artık oyun boyunca Unity **easyName** değişkenini bir tamsayı olarak görecek ve örneğin bu değişkene “**2.3**” gibi kesirli bir değer atamaya kalkarsak kesirli kısım olan “**.3**” kısmını kendisi atacak ve değişkeninin değerini “**2**” olarak ayarlayacak.

Ancak ben size kesinlikle UnityScript öğrenirken **değişkenin (variable)** türünü elle girmenizi tavsiye ediyorum, böylece oyununuzda sorun yaşarsanız nereden kaynaklandığını belki daha rahat bulabilirsiniz. Programlama konusunda epey ustalaşınca ise isterseniz uyuşukluk yapabilirsiniz.

NOT: Eğer bir scriptinizin başına “**#pragma strict**” kodunu eklerseniz artık bir değişken oluştururken türünü de belirtmek zorunda kalırsınız, aksi hâlde hata alırsınız. Güzel bir özellik.

NOT: Ne zaman ki bir değişken başına “**var**” takısı konup tanımlandıktan sonra scriptte herhangi bir yerde kullanılırsa; artık bu, o değişkenle işlem yaptığımız andır. Tıpkı 5. satırda olduğu gibi.

'float' (Kesirli Sayı) Türü



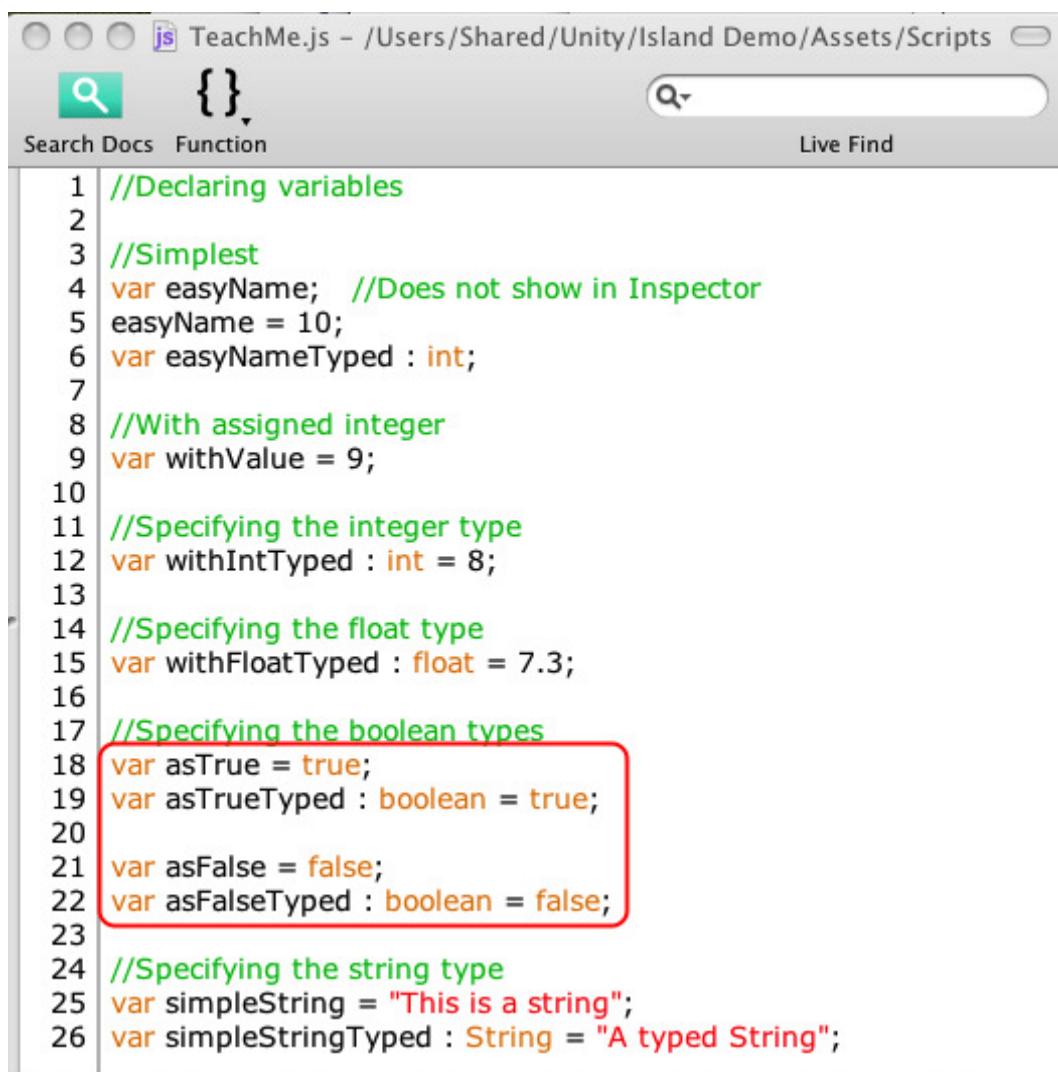
```
1 //Declaring variables
2
3 //Simplest
4 var easyName; //Does not show in Inspector
5 easyName = 10;
6 var easyNameTyped : int;
7
8 //With assigned integer
9 var withValue = 9;
10
11 //Specifying the integer type
12 var withIntTyped : int = 8;
13
14 //Specifying the float type
15 var withFloatTyped : float = 7.3; // This line is highlighted with a red rectangle.
16
17 //Specifying the boolean types
18 var asTrue = true;
19 var asTrueTyped : boolean = true;
20
21 var asFalse = false;
22 var asFalseTyped : boolean = false;
23
24 //Specifying the string type
25 var simpleString = "This is a string";
26 var simpleStringTyped : String = "A typed String";
```

15. satırda gördüğünüz gibi **float** türünde bir **değişken** (**variable**) tanımlanıyor. Bunun anlamı değişken kesirli bir sayıyı depolayacaktır.

Bu örnekte **withFloatTyped** değişkeninin, **tanımlama** (**declaration**) sırasında, sırf türünü belirlemekle kalmadım, ayrıca ona tanımlama sırasında bir değer de atadım "**7.3**". Bu kullanıma alışın çünkü bol bol kullanacaksınız.

Eğer ki değerini kesirsiz bir şekilde direk "**7**" olarak girseydim Unity bunu "**7.0**" olarak alacaktı.

'boolean' Türü



The screenshot shows a Unity script editor window titled "TeachMe.js". The code examples demonstrate various ways to declare boolean variables:

```
1 //Declaring variables
2
3 //Simplest
4 var easyName; //Does not show in Inspector
5 easyName = 10;
6 var easyNameTyped : int;
7
8 //With assigned integer
9 var withValue = 9;
10
11 //Specifying the integer type
12 var withIntTyped : int = 8;
13
14 //Specifying the float type
15 var withFloatTyped : float = 7.3;
16
17 //Specifying the boolean types
18 var asTrue = true;
19 var asTrueTyped : boolean = true;
20
21 var asFalse = false;
22 var asFalseTyped : boolean = false;
23
24 //Specifying the string type
25 var simpleString = "This is a string";
26 var simpleStringTyped : String = "A typed String";
```

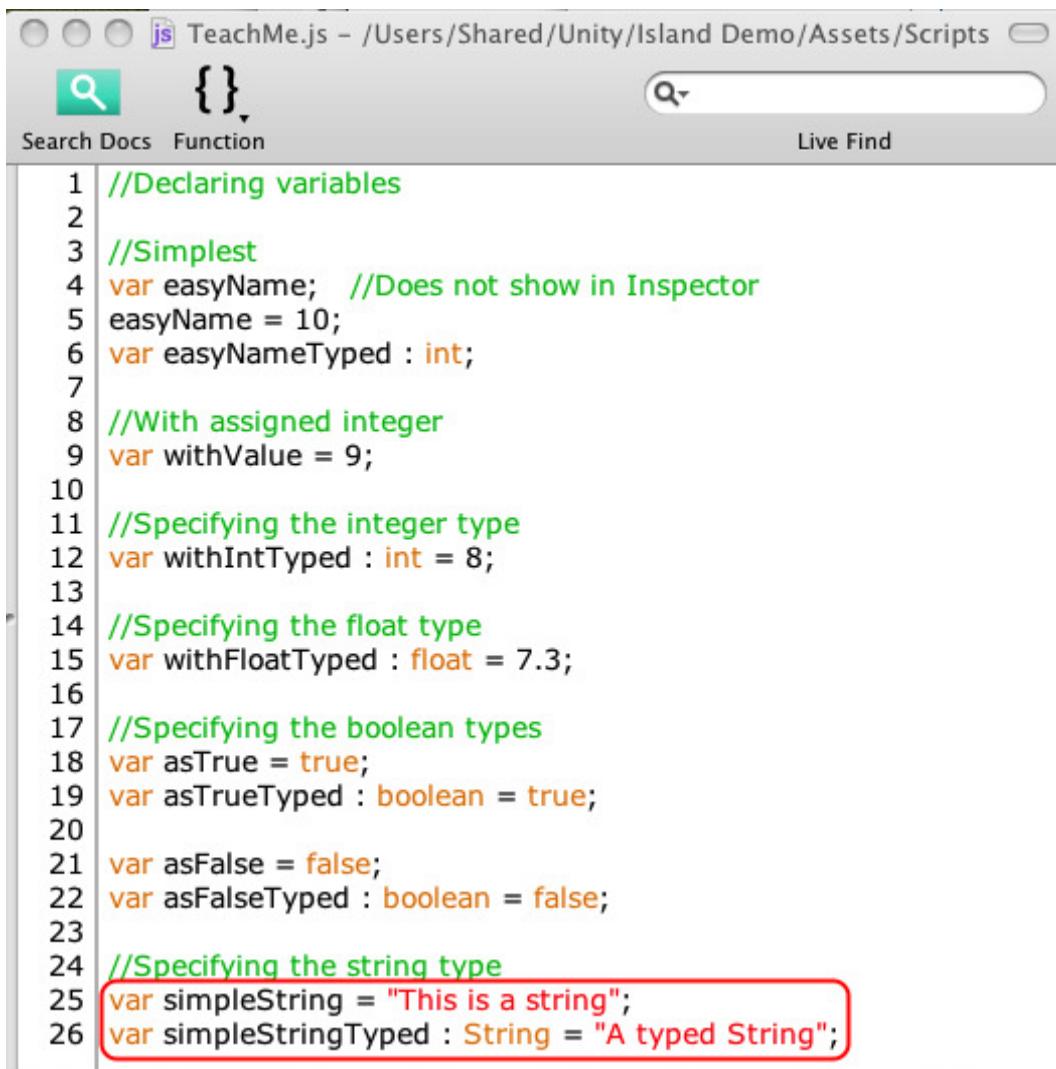
Lines 18 through 22, which declare boolean variables using the `true` and `false` keywords, are highlighted with a red rounded rectangle.

Bu **boolean** da neyin nesi? (Türkçe karşılığını bulamadım.)

Unity'nin arayüzündeki **Inspector** panelinde bu türde değişkenler birer " **işaret kutucuğu**" (**checkbox**) olarak gözükmektedir. Yani **boolean** türünde bir **değişken** (**variable**) ya **true** (**doğru**) ya da **false** (**yanlış**) değerini alabilir. Bu da Inspector'daki kutucuğun işaretli (tikli) ya da işaretetsiz olmasını sağlar.

Bu **true** ve **false** ile daha sonra; script yazarken daha yakından ilgileneceğiz.

‘String’ (Yazı) Türü



```
1 //Declaring variables
2
3 //Simplest
4 var easyName; //Does not show in Inspector
5 easyName = 10;
6 var easyNameTyped : int;
7
8 //With assigned integer
9 var withValue = 9;
10
11 //Specifying the integer type
12 var withIntTyped : int = 8;
13
14 //Specifying the float type
15 var withFloatTyped : float = 7.3;
16
17 //Specifying the boolean types
18 var asTrue = true;
19 var asTrueTyped : boolean = true;
20
21 var asFalse = false;
22 var asFalseTyped : boolean = false;
23
24 //Specifying the string type
25 var simpleString = "This is a string";
26 var simpleStringTyped : String = "A typed String";
```

The screenshot shows a Unity Editor window titled 'TeachMe.js - /Users/Shared/Unity/Island Demo/Assets/Scripts'. The code editor displays a script with various variable declarations. Line 26, which contains the declaration 'var simpleStringTyped : String = "A typed String";', is highlighted with a red rounded rectangle.

26. satırı bakın.

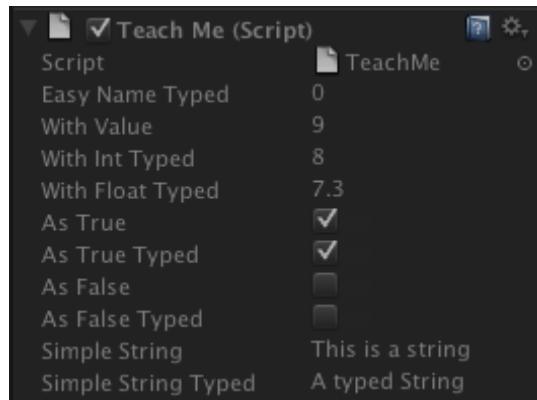
simpleStringTyped değişkeninin (**variable**) “**String**” olarak tanımlanması sırasında “**String**” türünün isminin büyük harfle başladığını dikkat edin. Bunun sebebi **String**’in bir **Class** (**Sınıf**) olmasıdır. “**int**”, “**float**” veya “**boolean**” türleri ise UnityScript içerisinde hali hazırda bulunan birer parçadırlar.

<http://unity3d.com/support/documentation/ScriptReference/String.html>

Tüm **class** (**Sınıf**) isimleri **büyük harfle başlar**. **Class** türünde değişken tanımlamalarını daha sonra daha detaylı olarak göreceğiz, típkı **Transform class’ı** gibi.

String türünde değişken tanımlarken değerini; alıntı yapar gibi iki tırnak işaretini arasında yazdığımızıza dikkat edin. (“**Bunun gibi.**”)

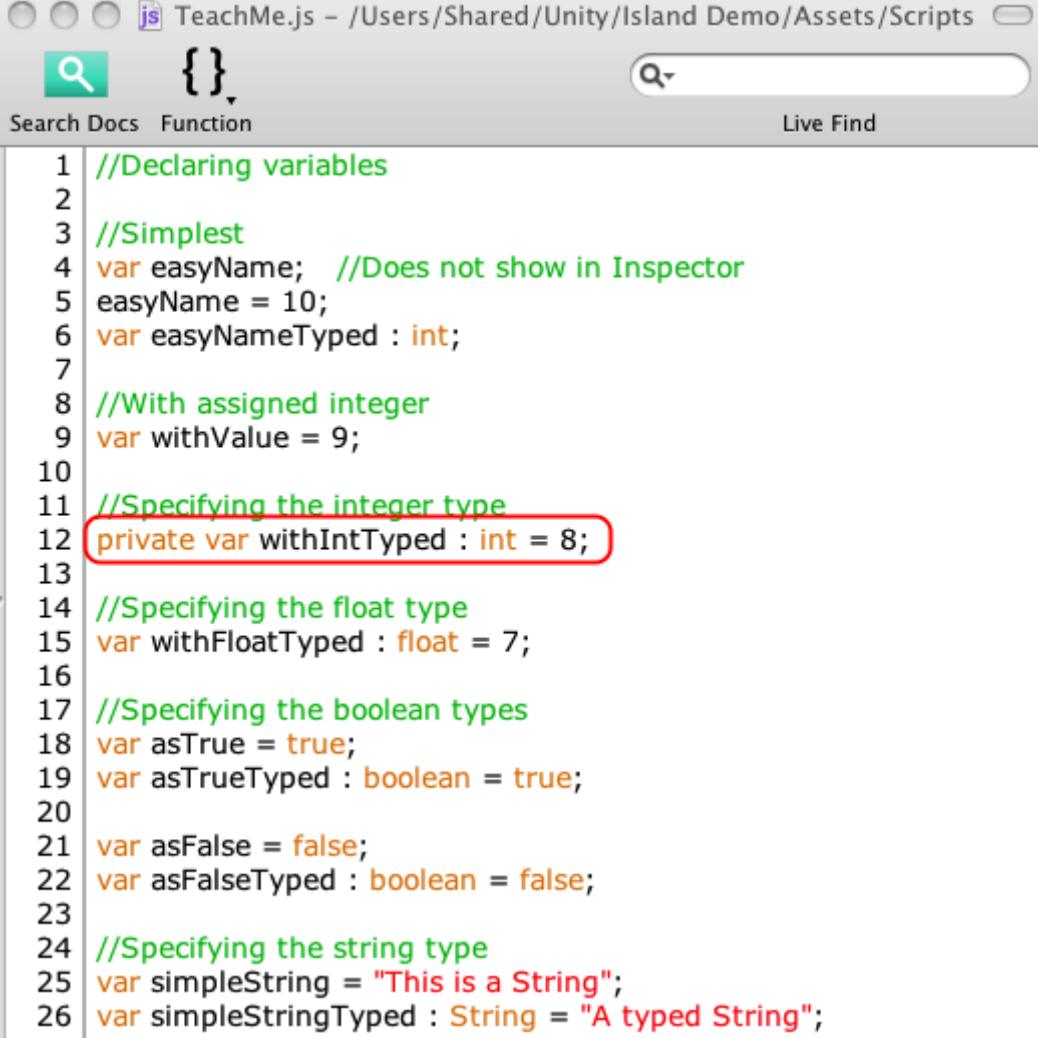
'Public' (Görünür, Açık) değişkenler



Benim az önce oluşturduğum **değişkenler (variable)** birer **public** (kamuya açık) değişkendiler. Bunun anlamı o değişkenlere başka scriptler tarafından da erişimin açık olmasıdır. Ayrıca o değişkenler Inspector panelinde de görünür durumdadırlar. Tek istisna 4. satırda bulunmaktadır. Çünkü o satırındaki **değişken tanımlamasında (declaration)** değişkenin (**easyName**) **türünü (type)** belirtmedim. Bu yüzden de Unity bu değişkeni Inspector'da göstermez. Çünkü değişkenin türünün ne olduğunuyla ilgili herhangi bir fikri yoktur.

Not: Daha sonradan **fonksiyonlar (function)** hakkında konuşurken bundan da bahsedeceğim, ancak şimdilik bilmeniz gereken; bir fonksiyonun içinde tanımlanan bir değişkenin asla **public** olmayacağıdır.

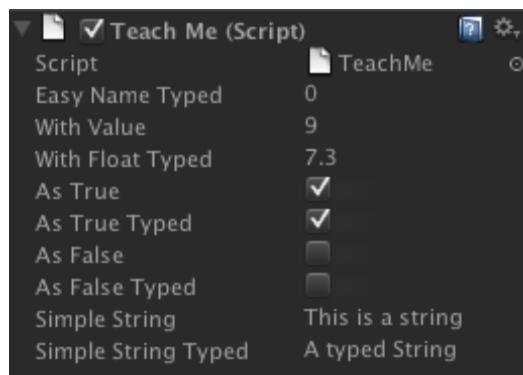
‘Private’ (Kişisel, Özel, Kapalı) değişkenler



```
1 //Declaring variables
2
3 //Simplest
4 var easyName; //Does not show in Inspector
5 easyName = 10;
6 var easyNameTyped : int;
7
8 //With assigned integer
9 var withValue = 9;
10
11 //Specifying the integer type
12 private var withIntTyped : int = 8;
13
14 //Specifying the float type
15 var withFloatTyped : float = 7;
16
17 //Specifying the boolean types
18 var asTrue = true;
19 var asTrueTyped : boolean = true;
20
21 var asFalse = false;
22 var asFalseTyped : boolean = false;
23
24 //Specifying the string type
25 var simpleString = "This is a String";
26 var simpleStringTyped : String = "A typed String";
```

12. satırda küçük bir değişiklik yaptım ve değişken tanımlamasında olması gereken “**var**” takısının başına bir de “**private**” takısı ekledim. Bunun anlamı orada tanımladığım “**withIntTyped**” isimli değişkenin **private** (**özel**) olacağı, yani başka scriptler tarafından erişilemeyeceğidir. Sadece değişkeni tanımladığım script içinde o değişkene ulaşabilir ve onunla işlem yapabilirim.

'Private' değişkenler Unity Inspector'da gözükmmezler



Gördüğünüz gibi; "withIntTyped" değişkenini (**variable**) **private** (**özel**) yaptığımız için değişkenin değeri **Inspector** panelinde gözükmüyor.

Sonuç

Gördüğünüz gibi değişkenlerle ilgili öğrenilmesi gereken pek çok konu var. Bir değişken tanımlarken yapıklarımızı şöyle özetleyebiliriz:

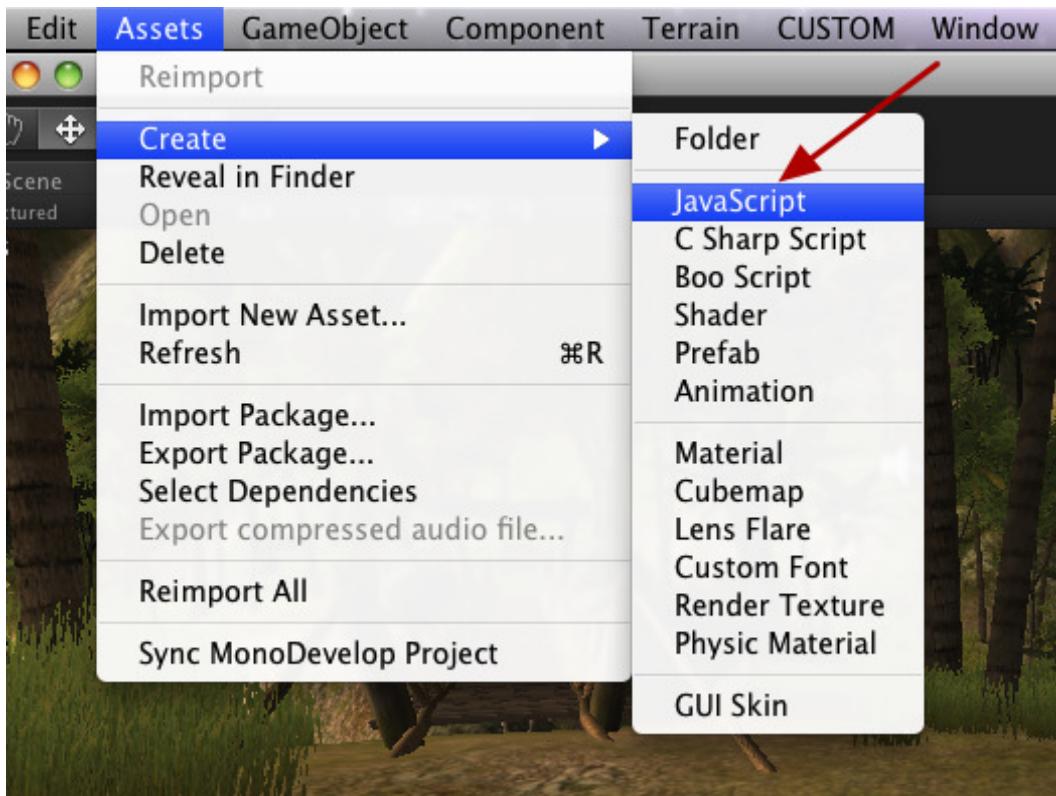
1. Değişkenimin **public** (**genel**) mi yoksa **private** (**özel**) mi olmasını istiyorum?
2. "var" takısı eklenir.
3. Değişken için kurallara uygun bir isim belirlenir. (İsim küçükle başlar ve içinde Türkçe karakter bulunduramaz.)
4. Değişkenin **türü** (**type**) belirlenir.
5. İsteğe bağlı olarak değişkene bir değer atanır.
6. Noktalı virgül işaretiyile işlem sonlandırılır.

Fonksiyonlara Detaylı Bakış

Unity'nin Update() (Güncel, Sürekli yapılan) ve Start() (Başlangıç, Tek seferlik) fonksiyonları

Unity'nin scriptlerimizi çalıştırmak için kullandığımız fonksiyonları bulunmaktadır. En yaygın olarak kullanılan fonksiyon **Update()** fonksiyonudur. Her yeni bir script oluşturduğunuzda bu fonksiyon da otomatik olarak scriptin içerisinde yazılı olarak gelir. Ee, bu da demektir ki bu fonksiyon dehset derecede önemli. Bir bakış atalım.

Yeni bir script oluşturmak



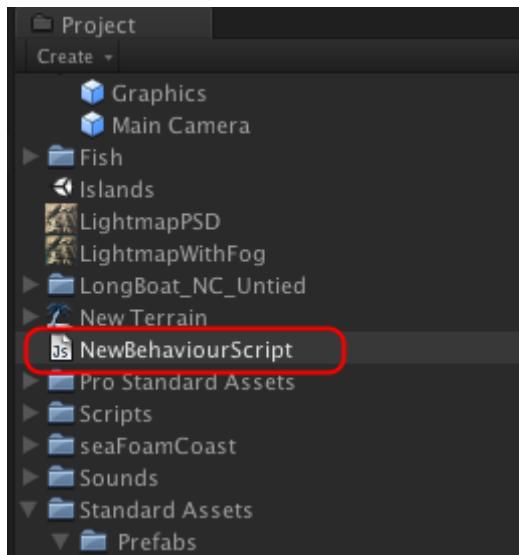
Yeni bir UnityScript (JavaScript) oluşturmak için üst taraftaki menüden şu yolu izleyin:

Assets → Create (Oluştur) → JavaScript

Niçin JavaScript? Unity'nin resmî forumlarında bu konuya ilgili pek çok tartışma konusu açıldı. Çoğu kişi bu dile JavaScript yerine UnityScript demeyi tercih ediyor, çünkü bu dil tam olarak JavaScript değil, ancak çok benzer yapıda. Ben de bu dili UnityScript olarak adlandıranlardanım.

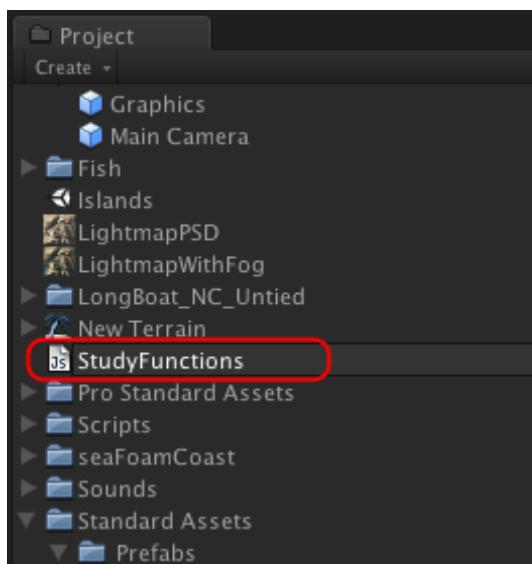
Bir script oluştururken '**Assets**' menüsünü kullandığımıza dikkat ettiniz mi? Hatırlayın, bir scripti **GameObject**'e (Oyun Objesi) atadığınız zaman script sihirli bir şekilde o **GameObject**'in bir **Component'i** (**Parça**) olveriyordu. Eğer sözlüğü açıp da 'asset' kelimesinin anlamına bakarsanız şu tanımla karşılaşırınsız: "*a useful or valuable thing or quality.*" (Türkçesi: **Değerli veya işe yarar, kullanışlı şey**). Ve elbette bir script de bir **GameObject**'i daha kullanışlı (useful) kılar. :)

NewBehaviourScript adında yeni bir script oluşturduk



Bu yeni UnityScript dosyası Unity'nin arayüzünde yer alan Project (Proje) paneline yerleştii. Varsayılan olarak ismi otomatikman '**NewBehaviourScript**' olarak ayarlandı.

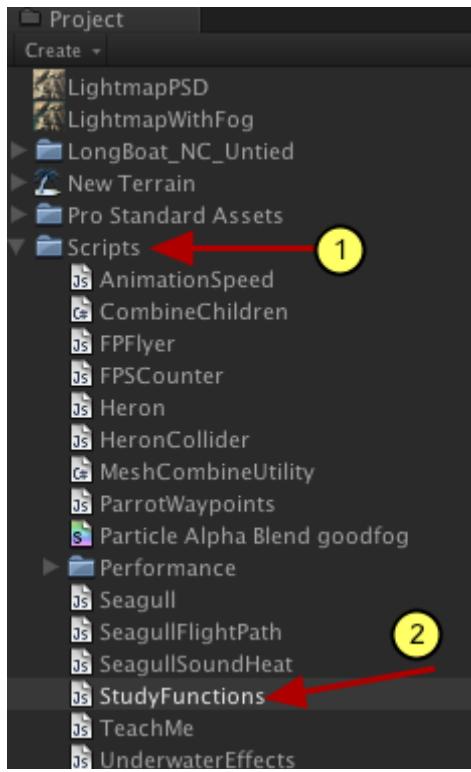
Scriptin ismini değiştirmek



Scriptinizin ismini değiştirmek için önce onu mouse ile seçin ve ardından **F2** tuşuna basın (MAC OS işletim sistemi kullanıyorsanız **Enter** tuşuna basın.). Ardından scriptinize ne işe yarayacağını her gördüğünüzde anlayabileceğiniz, açıklayıcı bir isim verin. Bu bir eğitim kılavuzu olduğu için ben kendi scriptimi '**StudyFunctions**' (EgitimFonksiyonları) olarak adlandırıyorum. İsim değişikliğini hallettikten sonra **Enter** tuşuna basarak isim değişikliğini bitirin.

(Çevirmen Eklemesi): Scriptlerinizi isimlendirirken, tipki değişkenlerde olduğu gibi, Türkçe karakter kullanmayın. Aksi takdirde bir objeye atadığınız scriptinizin Component'ine, başka bir scriptten erişim yapamazsınız, çünkü kodlama dilleri kesinlikle Türkçe karakterleri desteklemez, hata verir.

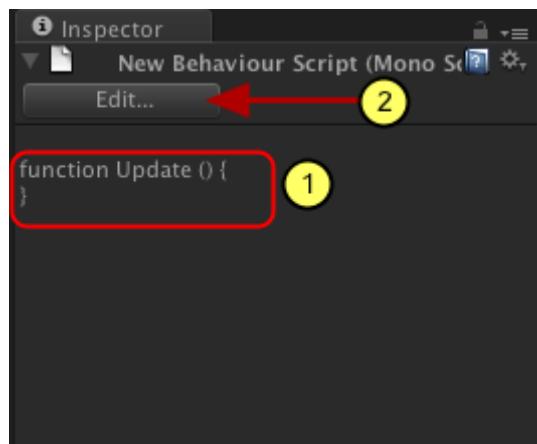
Scriptimizi 'Scripts' klasörüne taşımak



1. Sadece scriptinizi mouse ile sürükleyip 'Scripts' klasörüne atın.
2. İşte benim scriptim.

NOT: Eğer ki sizde '**Scripts**' diye bir klasör yoksa telaş yapmayın, çünkü bu klasör projenize istege bağlı olarak eklediğiniz hazır paketler vasıtasiyla gelir ve Unity'nin hazır scriptlerini içerisinde barındırır. Kendiniz bir klasör oluşturmak içinse yine **Project** panelinde, üst taraftaki '**Create**' (Oluştur) butonuna basın ve ardından '**Folder**' (Klasör) seçeneğini seçin.

Unity'nin Inspector paneline bakın



Project (Proje) panelinde scriptiniz seçiliyken **Inspector** paneline bakarsanız scriptinizin içeriğini görebilirsiniz.

1. Burada scriptinizde otomatik olarak oluşturulan **Update()** fonksiyonu bulunmaktadır.
2. Edit butonuna basarak script editörü vasıtasıyla scriptınızı düzenleyebilirsiniz.

Update(): Sürekli, sürekli ve sürekli "Şunu yap" fonksiyonu



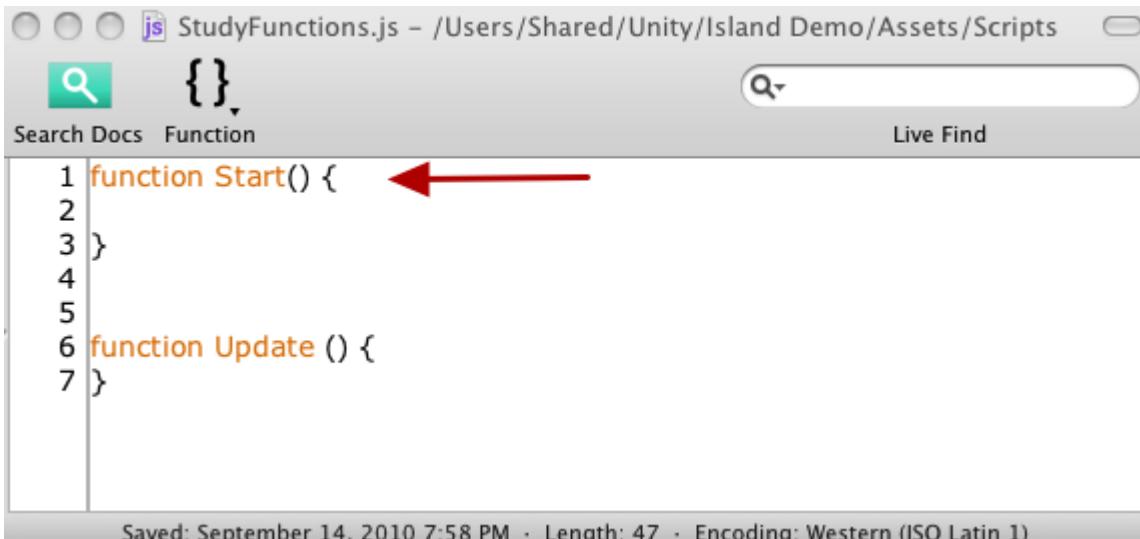
Niçin **Update()** fonksiyonu otomatik olarak yeni scriptlere eklenir?

Unity oyununuz sırasında her bir karede (**frame**) otomatik olarak bu fonksiyonu gerçekleştirir. ([Update\(\) Fonksiyonu İçin Referans Link](#))

(Çevirmen Eklemesi): Bir saniye kesinlikle bir kareye (**frame**) eşit değildir. Bir saniye oyunun hızına göre 30 kare de olabilir, 60 kare de 150 kare de. Bunu unutmayın.

Script Referans Kılavuzu'nu (Scripting Reference Manual) okudukça, çoğu kodun **Update()** fonksiyonu içerisinde yazıldığını görebilirsiniz.

Start(): Tek bir seferlik “Şunu yap” fonksiyonu



```
1 function Start() { ←  
2  
3 }  
4  
5  
6 function Update () {  
7 }
```

Saved: September 14, 2010 7:58 PM · Length: 47 · Encoding: Western (ISO Latin 1)

Unity içerisindeki bir başka çok yaygın kullanılan fonksiyon da **Start()** (Başlangıç) fonksiyonudur. Bu fonksiyon sadece 1 kere çalıştırılır ve uygulanır.

İçerisinde **Start()** fonksiyonu bulunan ve bir **GameObject**'in (**Oyun Objesi Component'i** (**Parça**) olan bir scriptteki **Start()** fonksiyonu, Unity tarafından, o obje oyunun herhangi bir bölümünde oluşturulduğu anda otomatik olarak tek bir seferlik çalıştırılır.

Fonksiyon İsimleri

Kılavuzun ilk bölümlerinde fonksiyonların ne oldukları hakkında ve niçin onlara ihtiyacımızın olduğu hakkında konuşmuştu.

Şimdi ise nasıl fonksiyon oluşturabileceğinizi aşama aşama göstereceğim.

Fonksiyon Tanımlamak (Declare)

```
studyfunctions.js - /Users/Shared/Unity/Bootcamp Demo/Assets/Scripts
function WhereAreYou(anyVariable : String)
{
    Debug.Log(anyVariable);
}

function Start()
{
    var location : String = "I'm down here to your Left";
    var noClue : String = "I don't know, I'm lost";
    WhereAreYou(location);
    WhereAreYou(noClue);
}

function Update () {
```

Kullanmak istediğiniz herhangi bir fonksiyonu (**function**) öncelikle tanımlamalısınız. Bunun anlamı nedir? Bunun anlamı siz Unity'e bir dizi kod yazacağınızı ve bu kod dizisine de bir isim vereceğinizi söylemektedir. Ardından Unity siz bu fonksiyonu çağrıdığınızda tam olarak neyden bahsettiğini bileyeciktir.

Hadi Start() fonksiyonuna bir bakalım. Bu sizin değil, Unity'nin içinde tanımlanmış olan ve otomatik olarak çağrılan bir fonksiyondur. Buna rağmen bu fonksiyonun içindeki kod dizisi tamamen size bağlıdır.

1. Fonksiyonun ismi olan **Start()**'tan hemen önce bir '**function**' takısı getirdik. Başa '**function**' takısını koymam Unity'e yeni bir fonksiyon oluşturmaktı olduğumuzu söyler.

2. Fonksiyon ismi olan **Start**'tan hemen sonra iki parantez "()" açılır. Bu parantezlerin anlamı, işlevi nedir?

Bu parantezler bir fonksiyonun parçası olmakla beraber (Yazması zorunlu parçası) bu parantezler sayesinde fonksiyonun içine özel veri (data) girişleri de yapabiliyoruz.

Şu anda kafanızı parçalayıp ne demek istediğimi kavramaya çalıştığınıza görür gibiyim :) . Madem **Start()** fonksiyonundaki parantezlerin içerisinde hiçbir şey yazmıyor, o zaman bu fonksiyona hangi veri giriyor ki?

Aslında hiçbir şey yok gibi duruyor değil mi, ancak korkmayın, adım adım bunu açıklayacağım.

3. Fonksiyonun başlangıç noktasına bir tırtıklı parantez '{' işaretini koyulur,
4. Fonksiyonun bitim noktasına da bir tırtıklı parantez '}' koyulur. **NOT: Fonksiyon tanımlamaları (declaration) göreceğiniz üzere sonlarına ';' (Noktalı virgül) almazlar.** Derleyici (**compiler**) bu sondaki tırtıklı parantezin '}' fonksiyonun bitiş noktasını olduğunu anlar.
5. Bu iki tırtıklı parantez (3 ve 4) arasındaki her şey fonksiyonun **gövdesini (body)** oluşturur, buraya yazılan kodlar fonksiyon çağrıldığı zaman gerçekleştirilir.

NOT: Biraz kişisel bilgiye kaçacak ama olsun. Ben aylar önce ilk defa fonksiyonları öğrenmeye başladığında bu şeyin nasıl işlediğini anlamak için az daha kendimi paralıyorum. Basitçe belirtmek gerekirse burada neler olup bittiğini bir türlü kafam almıyordu. Sonradan anladım ki fonksiyonlar tekrar tekrar kullanılabilen kod parçalarından ibaretmiş, bu kadar basit. Buna rağmen bir fonksiyonu çağırırmak ve içine özel veri girmek hâlâ kafamın basmadığı konulardı; bunun bir diğer nedeni de kitaplar yazıp bu konuda eğitim veren şahsiyetlerin bu konuda hiç detaylı bilgi vermemesiydı. Bu yüzden bu dersleri yazıyorum, çünkü ben bu konuya anlamaya çalışırken çok zorluklar çektim ve sizin de çekmenizi istemem. (**Bu yazı çevirmen eklemesi değildir, yazarın kaleminden çıkmıştır.**)

Bir sonraki bölüm, yaşadığım sıkıntıyı yaşamamanız için bu bahsettiğim konuya ilgili olacaktır.

Fonksiyonlara Özel Bakış

The screenshot shows a Unity script editor window titled "studyfunctions.js". The code is as follows:

```
1 function WhereAreYou(anyVariable : String)
2 {
3     Debug.Log(anyVariable);
4 }
5
6 function Start()
7 {
8     var location : String = "I'm down here to your Left";
9     var noClue : String = "I don't know, I'm lost";
10
11    WhereAreYou(location); 3
12    WhereAreYou(noClue); 4
13 }
14
15
16 function Update () {
17 }
```

Annotations with yellow circles and numbers:

- Annotation 1: A red box surrounds the first two lines of the `WhereAreYou` function definition.
- Annotation 2: A yellow circle with the number 2 is placed next to the word `anyVariable` in the `Debug.Log` call.
- Annotation 3: A yellow circle with the number 3 is placed next to the `location` variable in the `WhereAreYou(location);` call.
- Annotation 4: A yellow circle with the number 4 is placed next to the `noClue` variable in the `WhereAreYou(noClue);` call.

1. Bu benim oluşturduğum, **WhereAreYou()** ‘Neredesin()’ ismindeki fonksiyonun tanımlandığı kısım (Fonksiyon isimlerinin büyük harfle başladığını dikkat edin.). Gördüğünüz gibi önceki bölümde açıkladığım tüm parçalar bu fonksiyonda da mevcut.

2. Bu sefer fonksiyonun içerisinde ekstra bir şey var. İki parantezin ortasında **parametre (parameter)** denilen özel bir değişken oluşturdum. (Fonksiyona özel veri girmek bu parametreler vasıtasyyla olmaktadır.) Oluşturduğum bu değişkene **anyVariable : String** (herhangiDegisen : String) ismini verdim. Bunun anlamı fonksiyona göndereceğim verinin türünün **String** olacaktır. Bu sayede bu String değişkenle fonksiyonun içerisinde çeşitli işlemler yapabileceğim.

Fonksiyonlar tamamen isteğe bağlı olarak, içerisine veri (**data**) aktarmamıza yarayan onlarca parametreye de sahip olabilir ya da hiçbir parametreye sahip olmayabilir de. Her bir parametre ise „,“ (Virgül) işaretiley birbirinden ayrılır. **Start()** fonksiyonunda hiç parametre bulunmamaktadır, bu yüzden bu fonksiyonun içerisinde hiçbir özel veri girişi yapılmaz. Ancak kendi fonksiyonum olan **WhereAreYou()** fonksiyonunda ise 1 tane parametre bulunmaktadır.

İşte benim zamanında bir türlü anlayamadığım kısma geldik:

Henüz acemi biri gelip de fonksiyonun içerisindeki **anyVariable : String** parametresini görünce şöyle der: “Tamam, bu adam bu fonksiyonun içerisinde özel veri sokacak ve bunu da **anyVariable** isimli, **String** türündeki değişkenini kullanarak yapacak.”

AMA, bir de WhereAreYou() fonksiyonumun çağrıldığı anlara bakacak olursak:

3. WhereAreYou(location);
4. WhereAreYou(noClue);

Bu da nesi yahu? **3** numaralı kısımda fonksiyonu çağrıyoruz çağrımasına da ancak kullandığımız parametrenin adı **location** (konum) ve **4** numaralı kısımda ise parametrenin adı **noClue** (Anlamı: fikrimYok). Bu ne anlama geliyor, nasıl çalışıyor? Ne **location**, ne de **noClue** değişkenlerinden herhangi biri fonksiyonumda tanımlanmadı (**declare**) ?!

Uzun zaman boyunca kitaplarda bu konuyu araştırmama rağmen (O zamanlarda henüz Google diye bir şey yoktu.) basit bir yanıt bile bulamamıştım.

Kimse bu konuyu açıklamıyordu. Çeşitli deneme yanılma işlemleri ve kodla bol bol haşır neşir olma işleminin ardından en sonunda konuyu kendim sökmeyi başardım.

Az önceki örnekte, **anyVariable** isimli fonksiyon parametremizin değerine fonksiyon ilk çağrılığında (**3** numara) **location** değişkeninin değeri neyse o atanıyor, ikinci çağrılığında ise (**4** numara) **noClue** isimli değişkenin değeri parametremize atanıyor (**assign**).

Daha açıklayıcı olursak, fonksiyon ilk kez çağrıldığı sırada oyun motorumuzun arkaplanında şu işlem gerçekleşiyor:

anyVariable = location;

ve sonraki (ikinci) çağrılığında ise:

anyVariable = noClue;

Yani anlayacağınız **location** değişkeninin (**variable**) değeri **anyVariable** değişkenine kopyalandı (atanıyor). Ardından ikinci çağrılığında aynı işlem gerçekleşiyor, ancak bu sefer **noClue** değişkeninin değeri **anyVariable** değişkenine aktarılıyor. Ardından bu **anyVariable** parametremiz de fonksiyonumuzda kullanılıyor.

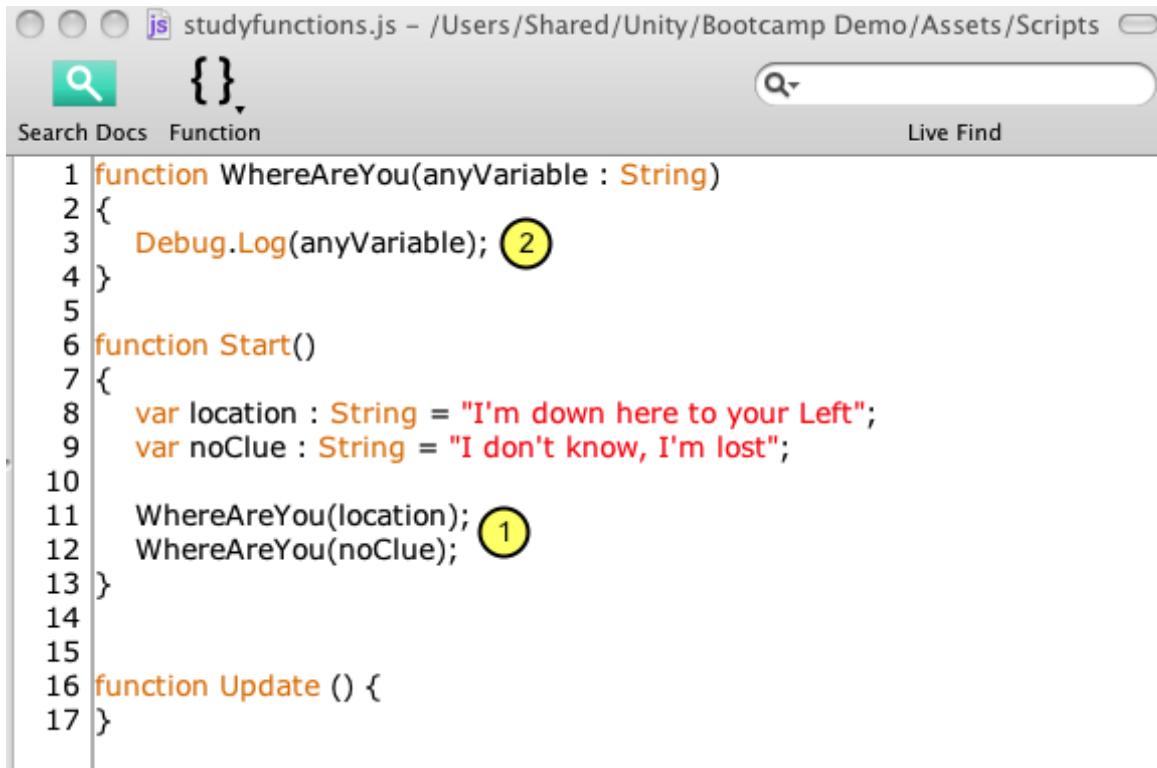
Yani bu yolla, **String** türündeki istediğim herhangi bir değişken ile fonksiyonu çağrıabilirim.

Ayrıca bu yolla **location** ve **noClue** değişkenlerinin değerleri parametreye **kopyalandığı** için; fonksiyon içerisinde **anyVariable** parametresinin değeriyle ilgili herhangi bir oynama yapsam bile **location** veya **noClue** değişkenlerinin değerlerine hiçbir etkisi olmuyor, onlar aynen kalıyor.

Fonksiyonun çağrılması ve akışı

Bir fonksiyon çağrıldığı zaman, başka fonksiyonlarla dolambaçlı etkileşimlerin de olabileceği basit bir akış meydana gelir.

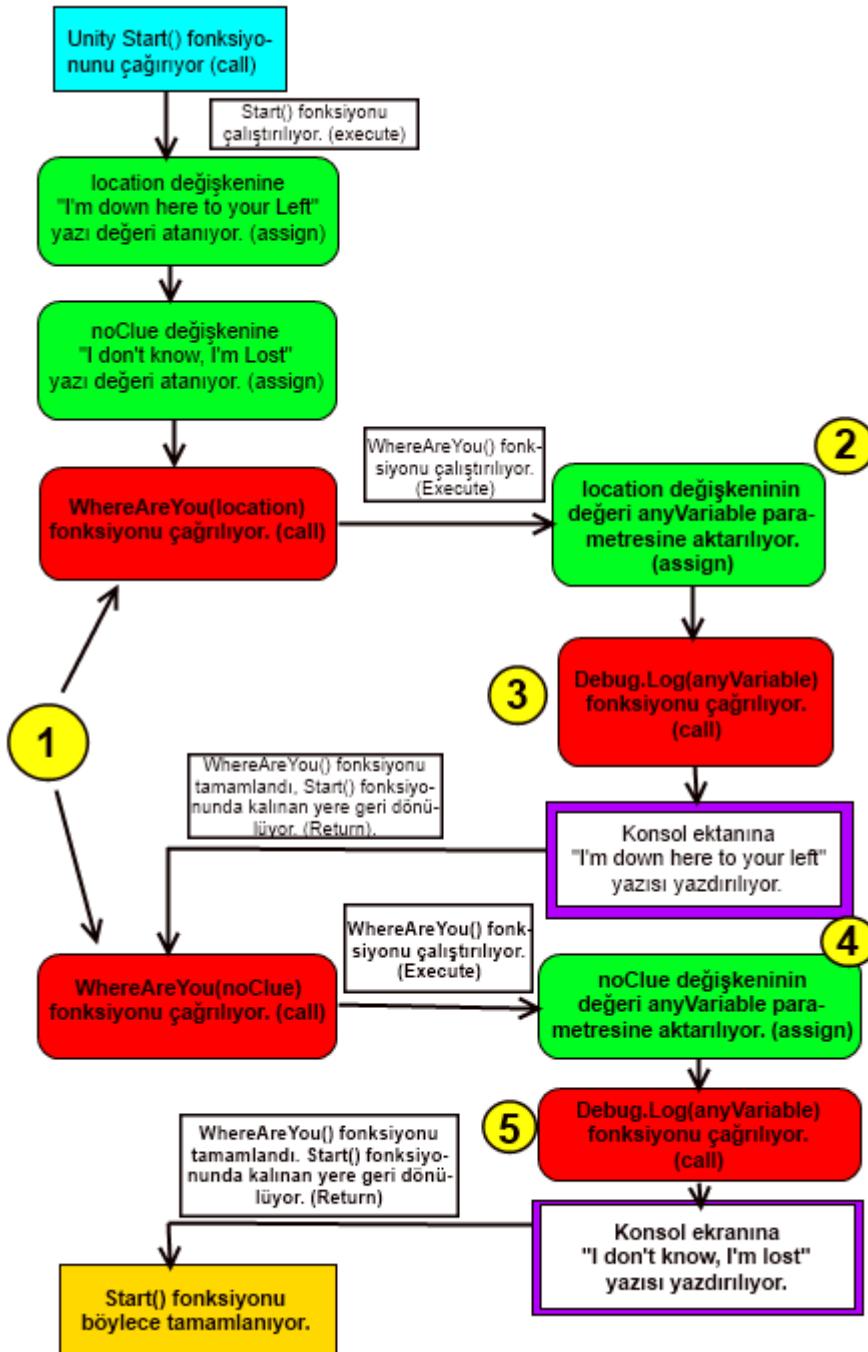
Hadi fonksiyonun çağrılmış aşamasına bakalım



```
1 function WhereAreYou(anyVariable : String)
2 {
3     Debug.Log(anyVariable); (2)
4 }
5
6 function Start()
7 {
8     var location : String = "I'm down here to your Left";
9     var noClue : String = "I don't know, I'm lost";
10
11    WhereAreYou(location); (1)
12    WhereAreYou(noClue);
13 }
14
15
16 function Update () {
17 }
```

1. **Start()** fonksiyonunda **WhereAreYou()** fonksiyonu 2 kere çağrılmıyor;
2. Burası **WhereAreYou()**'nun içeriğinin olduğu kısım; yani **WhereAreYou()** fonksiyonunun ne işe yarayacağı buradan belirleniyor. Buradaki **Debug.Log()** ise Unity'nin hâli hazırda bir hazır fonksiyonu olup yaptığı şey Unity'nin konsol ekranına (console) çıktı vermektedir. (Çıktı derken yazıcı çıktısı değil, içerisinde girilen değeri orada göstermesini kastediyorum.)

Scriptin akış şeması



Resimde sol tarafta Start() fonksiyonunun çalışma aşamaları gözükmeektedir.

Sağ tarafta ise WhereAreYou() fonksiyonunun çalışma aşamaları gözükmeektedir.

1. **Start()** fonksiyonunun içerisinde **WhereAreYou()** fonksiyonu 2 kere çağrılmaktadır. (**call**)
2. Bu **WhereAreYou()** fonksiyonunun ilk defa çalıştırıldığı andır.
3. Konsol ekranına çıktı vermek için **Debug.Log()** fonksiyonu çağrılıyor.
4. **WhereAreYou()** fonksiyonu 2. kez çağrılıyor (**call**) ve çalıştırılıyor (**execute**).
5. Konsol ekranına çıktı vermek için **Debug.Log()** fonksiyonu tekrar çağrılıyor. (**call**)

Konsol Çıktısı (Output)



1. **WhereAreYou()** fonksiyonunun ilk kez çağrılmaması sonucu oluşan çıktı;
2. **WhereAreYou()** fonksiyonunun ikinci kez çağrılmaması sonucu oluşan çıktı.

Bu script'i çalıştırmak için önce scriptin bir **GameObject**'e atanması gereklidir. Ben bunun için boş bir **GameObject** (Oyun Objesi) oluşturduğum scripti seçip sürükleyerek objeme atadım. Ardından '**Play**' (Oyna) butonuyla oyunu çalıştırıldım. Aldığım çıktı da böyle oldu. (Üstteki resim)

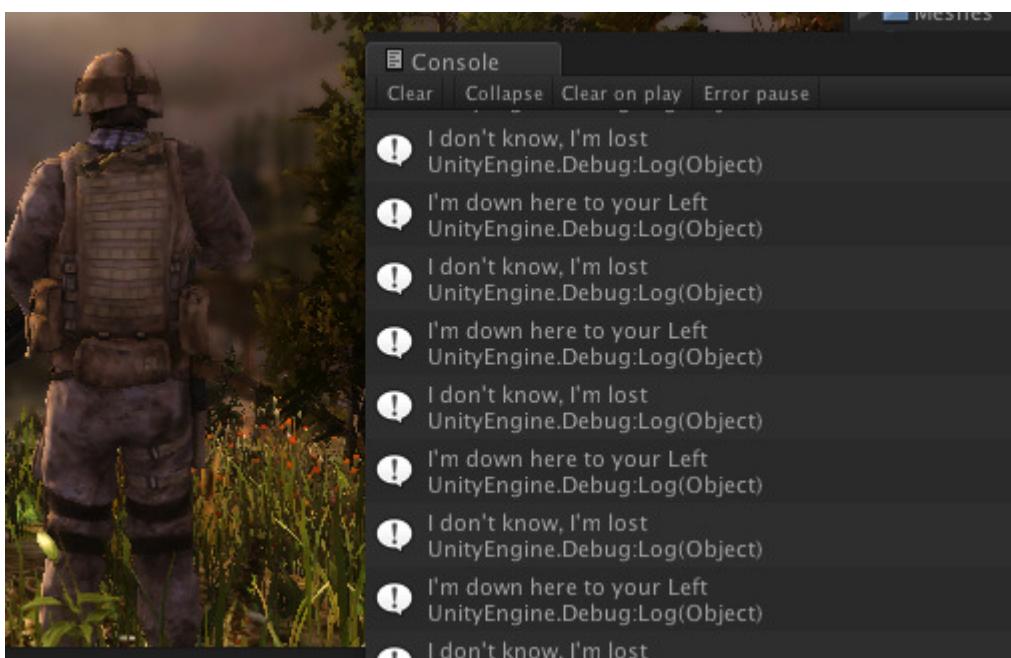
Ya Update() fonksiyonunu kullanmış olsaydık?



```
studyfunctions.js – /Users/Shared/Unity/Bootcamp Demo/Assets/Scripts
Search Docs Function Live Find
1 function WhereAreYou(anyVariable : String)
2 {
3     Debug.Log(anyVariable);
4 }
5
6 function Start()
7 {
8 }
9
10
11 function Update () {
12
13     var location : String = "I'm down here to your Left";
14     var noClue : String = "I don't know, I'm lost";
15
16     WhereAreYou(location);
17     WhereAreYou(noClue);
18 }
19 }
```

Start() fonksiyonunun yerine bu sefer Update() fonksiyonunu kullandım.

Update() fonksiyonunun kullanılmasının sonucu



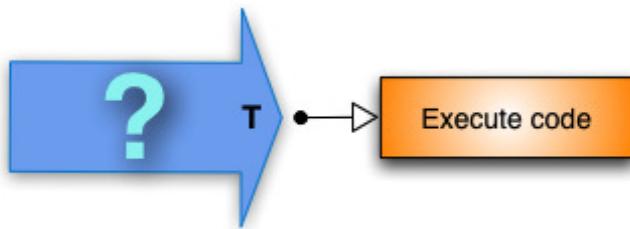
Beklediğimiz gibi, **Update()** fonksiyonu her bir karede çalıştırıldığından dolayı konsol çıktısı oyunu durdurana kadar (stop) sürekli aynı çıktıyi vermeye devam ediyor.

**Seçimler, seçimler, her
zaman seçim yapmak
zorundayız**

if (eğer)?

Eğer, eğer, eğer. Eğer şunu yaparsam... eğer bunu yaparsam... peki eğer şöyle yaparsam ne olur... Kesinlikle 'eğer' (**if**) kelimesini hayatımız boyunca her gün bolca kullanıyoruz. Yani zaten 'eğer' (**if**) kelimesinin ne işe yaradığını biliyorsunuz, bu kelime karar vermemize (seçim yapmamıza) yarar.

if ifadesi (Statement), doğru (true) ya da yanlış (false)



Oyuncu bir butona bastı ve kapı kilitlendi. Oyuncu öldü. $7 > 6$ (7 sayısı 6'dan büyütür.)

Yukarıdaki gibi genel **ifadeler (statement)**, ya **doğru**'dur (**true**) ya da **yansız**'tırılar (**false**). Örneğin: "Ayıcık ormanda kendine bir ev yaptı." ifadesi ya **doğrudur** "Ayı ev yaptı." ya da **yansızdır** "Ayı ev yapmadı.".

Peki burada **if (eğer)** ifadesi nerde? Hadi alttaki 2 örneğe bakalım:

Koşullar (Condition)

```
function Start (){  
    if( 7 > 6){  
        Debug.Log("Yes");  
    }  
}
```

ifStatement.js (egerifadesi.js) adında yeni bir UnityScript (JavaScript) oluşturdum ve bu scripti sahnedeki (scene) boş bir **GameObject**'e atadım.

1. Önce **if** takısı yazdım, ardından iki parantez açıp aralarına cevabının **doğru** (**true**) veya **yanlış** (**false**) olabileceği bir ifade (**statement**) ekledim.

2. Eğer ifade doğruysa yapılacakları (çalıştırılacak kodları); iki tırtıklı parantez arasına '**{ }**' yazdım.

1. parçadaki iki adet parantezin arasında yazdığımız **ifadelere** (**statement**) **koşul** (**condition**) denir. Yani bir **if** (**eğer**) ifadesi şöyle çalışır:

Eğer koşullarım (**condition**) tutuyorsa o zaman içimdeki kodları çalıştırırım (**execute**).

Bu durumda, $7 > 6$ ifadesi daima **doğrudur** (**true**), yani konsol ekranına "**Yes**" (Evet) yazısı yazdırılır.

Not: Unity size böyle bir **koşulda** (**condition**) bir uyarı (**warning**) verir, der ki girdığınız koşul **daima doğrudur** (**true**). Bu bir test olduğu için bu uyarıyı dinlemenize gerek yok, ne de olsa **if** ifadesi görevini yerine getiriyor. Koşulun hep doğru olmasının sebebi ise 7 ve 6 sayılarının birer **değişken** (**variable**) değil de birer **sabit** (**constant**) olmalarıdır.

Bu sayılar asla değişmez, 7 her zaman 7 olarak geçer.

Doğruluğu "true" test etmek

The screenshot shows the Unity Editor's code editor window. The file is named 'ifStatements.js'. The code contains the following JavaScript:

```
1 function Start (){
2     if( 7 > 6){
3         Debug.Log("Yes");
4     }
5     var theBearMadeBigPottyInTheWoods : boolean = true;
6
7     if(theBearMadeBigPottyInTheWoods){
8         Debug.Log("It's stinky, too");
9     }
10 }
```

Annotations with yellow circles and red arrows highlight specific parts of the code:

- Annotation 1 points to the condition `7 > 6`.
- Annotation 2 points to the assignment `var theBearMadeBigPottyInTheWoods : boolean = true;`.
- Annotation 3 points to the variable `theBearMadeBigPottyInTheWoods` in the second `if` statement.
- Annotation 4 points to the first `if` statement.
- Annotation 5 points to the closing brace of the first `if` statement.

O zaman ben nasıl "Ayı ormanda kendine ev yaptı." gibi bir ifadeyi test edebilirim?

Bu cevabı ya **doğru** (**true**) ya da **yanlış** (**false**) olacak şekilde yazılmalıdır. Bunun için **boolean** tipindeki değişkenleri kullanız.

boolean türünde bir değişken kullanırsınız çünkü bir **boolean** değişken sadece 2 değer alabilir: **true** ya da **false**.

1. Peki o zaman **theBearMadeBigPottyInTheWoods** (ayiOrmandaBuyukEvYapti) adında bir değişken oluşturarak başlayalım (Aslında Potty'nin sözlük anlamı 'deli'dir ancak ben bunu cümleye bir türlü uyaramadığımdan onun yerine 'ev' kelimesini kullandım.). Tabi ki çok daha kısa bir isim kullanabilirdim ancak böyle bir ismi daha kolay hatırlayabileceğimi düşündüğüm için bu ismi kullandım.

2. Değişkenin türünü **boolean** olarak ayarladım.

3. Değişkene **true** (doğru) değerini **atadım (assign)**.

4. Bir **if ifadesi (statement)** oluşturdum ve parantezlerin arasına az önce oluşturduğum değişkenin adını yazdım.

Bu örnekte tabi ki ilgili **koşulun (condition)** tutacağını biliyoruz çünkü değişkenimize oluşturma aşamasında **true** değerini vermiştık.

5. Sonuç olarak, **Play** butonuyla oyunu başlattığında konsol ekranında ""It's stinky, too" yazısı belirir.

Hatırlatma yapacak olursak, bir değişken, depoladığı verinin yerine geçer, yani ne zaman **theBearMadeBigPottyInTheWoods** değişkeni bir yerde kullanılsa, değişkenin yerine değeri olan **true** geçer. Bunun anlamı **if** ifadesindeki **koşul(lar) (condition)** sağlanır ve böylece içerisindeki kod çalıştırılarak (**execute**) **Debug.Log()** fonksiyonu çağrılrır (**call**).

Eğer ki **theBearMadeBigPottyInTheWoods** değişkeninin değerini **false** olarak ayarladık o zaman koşul(lar) tutmadı ve **if** ifadesi çalıştırılmazdı; haliyle de **Debug.Log()** fonksiyonu çalıştırılmaz, konsol ekranına çıktı verilmezdi.

"Tersini" Test Etmek?

```
1 function Start (){
2     if( 7 > 6){
3         Debug.Log("Yes");
4     }
5     var theBearMadeBigPottyInTheWoods : boolean = false;
6
7     if(!theBearMadeBigPottyInTheWoods){
8         Debug.Log("It's on the ice");
9     }
10 }
11
```

Tersini mi test etmek? Evet, aynen öyle. Bu biraz hile yapmak gibi bir şey, bu yeni örnekte **false** değerini **true** değerine çevireceğiz ve böylece **if** ifadesi çalıştırılacak.

1. Değişkeni oluştururken bu sefer değerini başta **false** olarak ayarladığımıza dikkat edin. Bunu yaptım çünkü ayımız bir kutup ayısı ve onlar evlerini ormanda yapmazlar.

Şimdi biraz tersine zekâ kullanarak **if** ifadesini yine çalıştırmayı başaracağız.

2. Parantezler içindeki **theBearMadeBigPottyInTheWoods** değişkeninin başına çok dikkatlice bakın. İsminin tam önünde bir **ünlem işareteti** ‘!’ var!

Peki bunun anlamı nedir?

Bunun anlamı ‘**Değilse**’dir (**Not**).

Kafanızı buraya yoğunlaştırmaya çalışın. Bu ifadenin anlamı; **theBearMadeBigPottyInTheWoods** değeri bu “**Değilse**” demektir.

theBearMadeBigPottyInTheWoods değişkeninde depolanan değer **false** idi.

Ve bizim burada söylediğimiz şey de **theBearMadeBigPottyInTheWoods** değeri **değilse**, ya da bir başka deyişle **false değilse**.

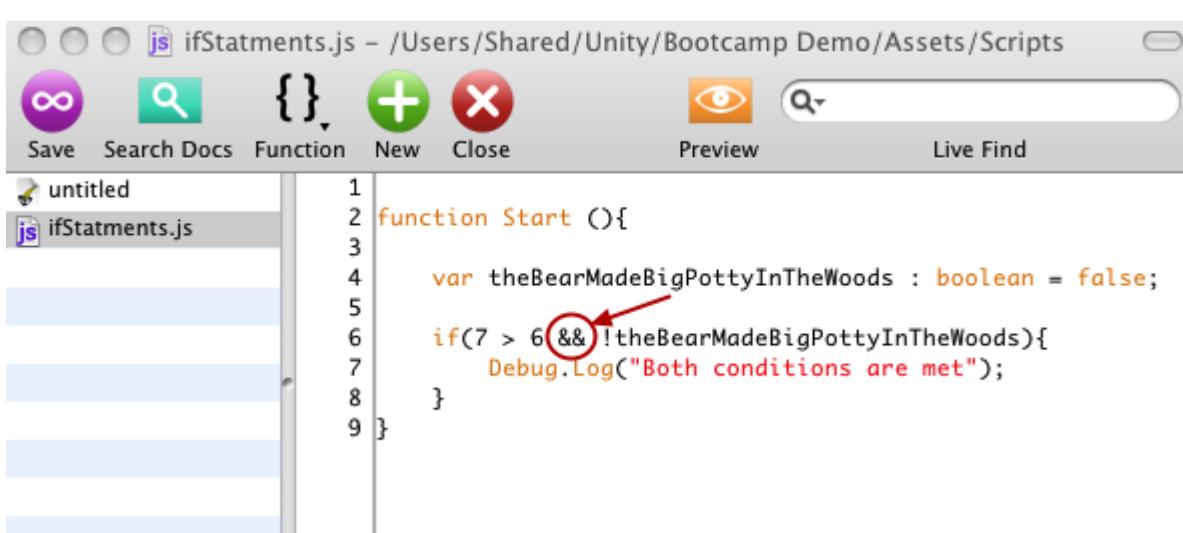
False değil demek de **true** demekle aynı anlama gelmektedir.

Yani anlayacağınız bir koşulun başına **ünlem işareteti** koyarsanız, **koşulun değerinin tam tersi işleme alınır**.

Yani, bu sayede **if** ifadesinin koşulu sağlanmış olur ve konsol ekranında “It’s on the ice” (Buzun üzerinde) çıktısı alınır.

Koşul değişkenlerinin illâ boolean türünde olması gereklidir

Peki ya birden fazla (çoklu) (multiple) koşullu durumlar?

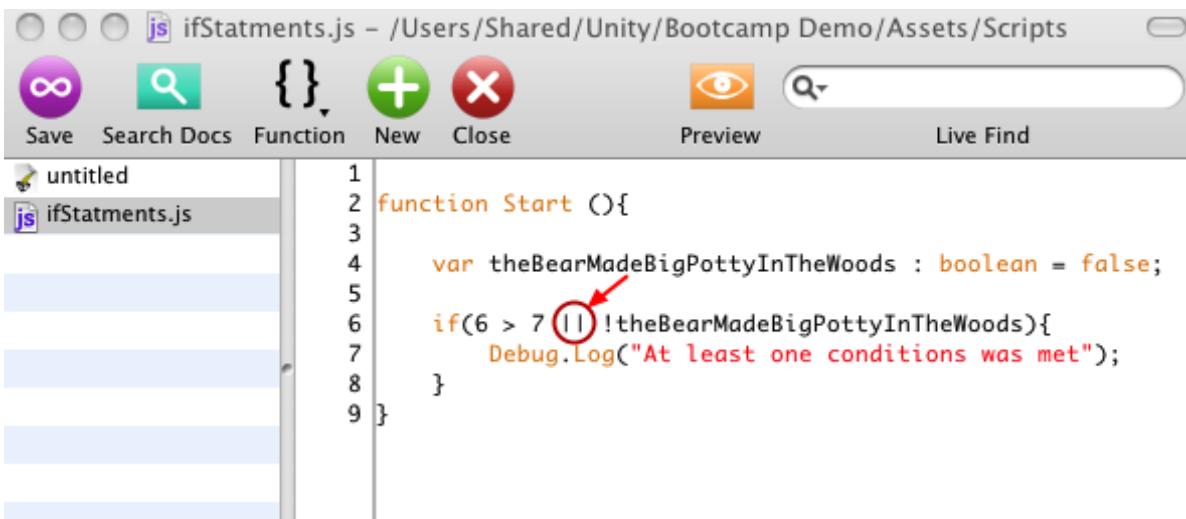


```
1 function Start () {
2     var theBearMadeBigPottyInTheWoods : boolean = false;
3
4     if(7 > 6 && !theBearMadeBigPottyInTheWoods){
5         Debug.Log("Both conditions are met");
6     }
7 }
```

Bir if ifadesinin çalıştırılması için birden çok koşulun da sağlanmasını isteyebilirsiniz. Bunu yapmak için koşulların arasına “**&&**” (VE anlamına gelir.) eklemesi yapmak işinizi görür.

Bunun anlamı tüm koşulların (**condition**) değeri **true** olmalıdır ki **if** ifadesi (**statement**) çalıştırılsın.

Çoklu koşullu durumlar, ancak bu sefer birinin doğru olması yeterli



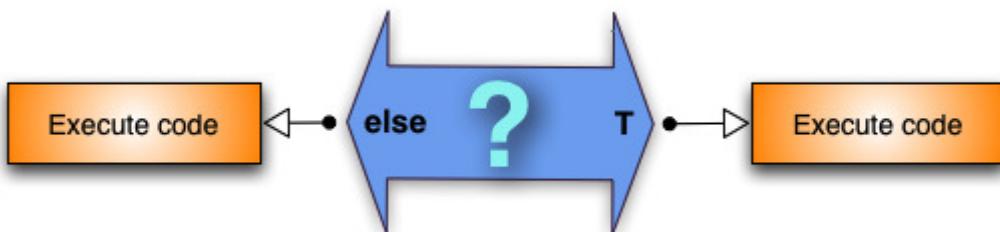
```
function Start () {
    var theBearMadeBigPottyInTheWoods : boolean = false;
    if(6 > 7 || !theBearMadeBigPottyInTheWoods){
        Debug.Log("At least one conditions was met");
    }
}
```

Burada değerinin **false** olduğu bir koşul yazdım ($6 > 7$). Ardından diğer koşulla arasında (**||**) eklemesi yaptım. Bu işaret "**VEYA**" demektir.

Yani bunun anlamı **if** ifadesindeki sadece bir koşulun sağlanması **if** ifadesini çalıştırılmak (**execute**) için yeterlidir. " $6 > 7$ " **VEYA** `!theBearMadeBigPottyInTheWoods`

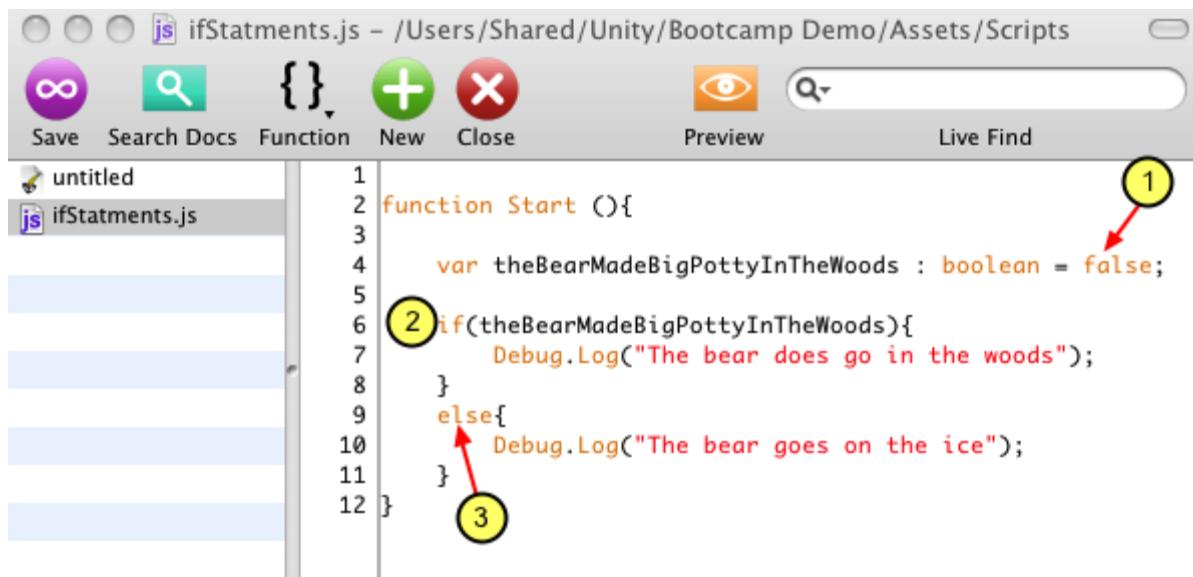
Bu durumda $6 > 7$ koşulunun değeri **false**'dır, ancak `!theBearMadeBigPottyInTheWoods` ifadesinin değeri **true**'dır. Yani **if** ifadesi çalıştırılır.

Çalıştırılacak opsiyonel (alternatif) kod



Az önce gösterdiğim **if** ifadeleri ancak içerisindeki koşullar sağlanırsa çalıştırılıyordu. Şimdi ise bu koşullar sağlanmazsa alternatif bir kod kümесinin çalıştırılmasını göstereceğim.

if-else (eğer-değilse) İfadesi



The screenshot shows the Unity Text Editor window with the file 'ifStatements.js' open. The code is as follows:

```
1 function Start () {
2     var theBearMadeBigPottyInTheWoods : boolean = false;
3
4     if(theBearMadeBigPottyInTheWoods){
5         Debug.Log("The bear does go in the woods");
6     }
7     else{
8         Debug.Log("The bear goes on the ice");
9     }
10 }
11
12 }
```

Annotations with yellow circles and numbers:

- Annotation 1: A red arrow points from the word "false;" to the number 1.
- Annotation 2: A red arrow points from the "if" keyword to the number 2.
- Annotation 3: A red arrow points from the "else" keyword to the number 3.

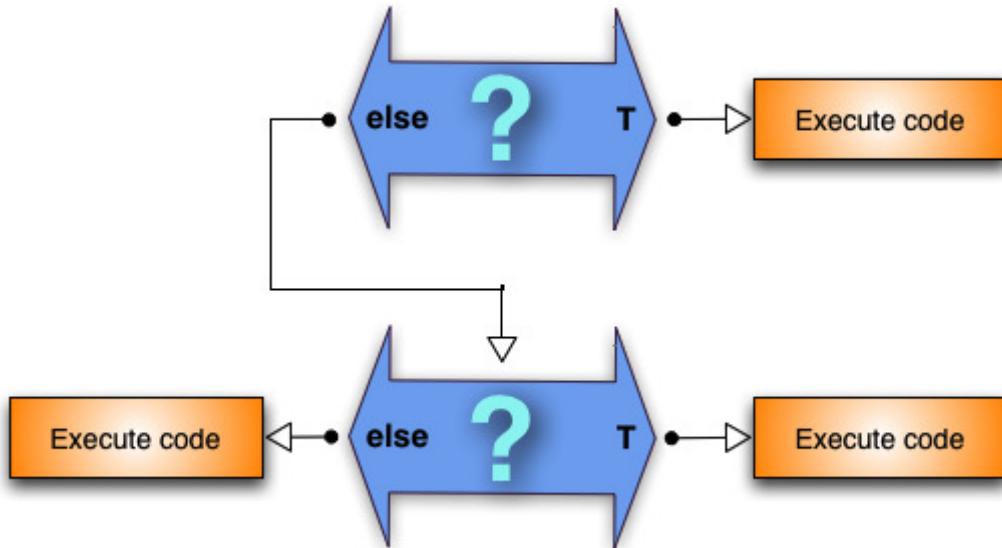
1. Değişkenin değerini false olarak ayarlıyorum.
2. **if** ifadesinin az önce gördüğümüz, normal kısmını yazıyorum.
3. **else** (değilse) takısı ekleyerek iki adet tırtıklı parantez arasına “{ }” opsiyonel kod(lar)ı yazıyorum.

Bu işlem oldukça basit. Eğer ki **if** ifadesindeki koşullar sağlanmazsa o zaman **if** ifadesi yerine **else** ifadesinin içerisindeki kod(lar) çalıştırılır.

Burada if ifadesi şöyle işler:

If (eğer) koşullarım (**condition**) tutuyorsa benim kodlarımı çalıştır, yok eğer tutmuyorsa **else**'nin içindeki, alternatif kodları çalıştır.

Peki ya birden çok seçenek istersek?



Seçimler yaparken fark edeceksiniz ki 1 veya 2 tane değil de daha çok seçeneğe sahip sorgular ortaya çıkacaktır. Böyle durumlarda ise basitçe, **if-else** kodunun **else** parçasının içerisinde yeni bir **if-else** ifadesi koyarak çoklu seçeneğe sahip sorgular oluşturabilirsiniz.

Buna "**nesting**" denmektedir (Sözlük anlamı: Yuva kurmak, yuvalama).

if-else-if-else... (eğer-değilse-eğer-değilse)

```
1 var theBearMadeBigPottyInTheWoods : boolean = false; 5
2 var thePolarBearMadeBigPottyOnTheIce : boolean = false;
3
4 function Start () {
5
6     1 if(theBearMadeBigPottyInTheWoods){
7         Debug.Log("The bear does go in the woods");
8     }
9     2 else{
10         1 if(thePolarBearMadeBigPottyOnTheIce){
11             Debug.Log("The polar bear goes on the ice");
12         }
13         2 else{
14             Debug.Log("The polar bear goes in the ocean");
15         }
16     }
17 }
18
19 }
20 }
```

if-else ifadelerinin (**statement**) sayısını istediğiniz kadar yapabilirsiniz ve her **if** ifadesi için istediğiniz **koşulları** (**condition**) kullanabilirsiniz.

1. İlk **if** ifadesi

2. İlk **else** ifadesi

3. İkinci **if** ifadesi

Bu ikinci **if** ifadesinin; ilk **else** ifadesinin icerisinde yer aldığına dikkat edin. (#2)

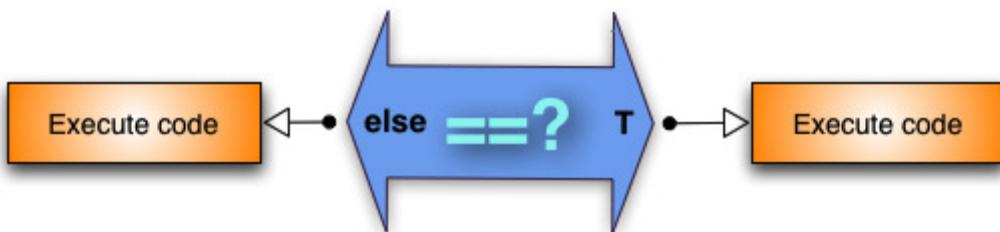
4. İkinci **else** ifadesi

Bu örneği çalıştırduğumızda alacağımız konsol çıktısı "*The polar bear goes in the ocean*" olacaktır çünkü iki **if** ifadesinin ikisinin de içerisindeki koşullar tutmamaktadır, ikisinin de değeri **false**edir (yanlış). Peki niçin? Çünkü hem **theBearMadeBigPottyInTheWoods**, hem de **thePolarBearMadeBigPottyOnTheIce** değişkenlerinin ikisinin de değeri **false** olarak atanmıştır (**assign**).

Not:

5. Gördüğünüz gibi iki değişkenin de oluşum aşaması kodlarını **Start()** fonksiyonunun içerisinde çikardım. Böylece değişkenler **public** (genel, görünür) hâle geldiler ve artık Unity'nin **Inspector** panelinde değerlerini elle değiştirebilirim. Bu yolla **Inspector**'dan değişkenlerin değerlerini istediğiniz gibi değiştirerek konsol çıktısının nasıl değiştiğine bakmanızı tavsiye ederim.

boolean'ın dışındaki türlerde değişkenlerle koşullama işlemi



Size gösterdiğim tüm örneklerdeki değişkenlerin türleri **boolean** idi, değerleri de sadece **true** veya **false** olabiliyordu. Bu da **if** ifadesinde koşulların **true** veya **false** olarak sınamasını epey kolaylaştıryordu.

if ifadesinin içerisinde istedigimiz türde değişkenleri koşul olarak kullanabiliriz. Peki bunu nasıl yaparız?

Hayatınızda hiç herhangi bir insana cevabının doğru (**true**) veya yanlış (**false**) olabileceği bir soru sordunuz mu? Elbette sordunuz. Şimdi yapacağımız şey de tam olarak bu.

"== " eşitse işaretti

```
1 var theBearMadeBigPottyInTheWoods : boolean = false;
2 var thePolarBearMadeBigPottyOnTheIce : boolean = false;
3
4 var theSeasonOfTheYear : String = "summer";
5
6 function Start () {
7
8     if(theSeasonOfTheYear == "summer"){
9
10         if(theBearMadeBigPottyInTheWoods){
11             Debug.Log("The bear does go in the woods");
12         }
13     }
14     else{
15
16         if(thePolarBearMadeBigPottyOnTheIce){
17             Debug.Log("The polar bear goes on the ice");
18         }
19     }
20     else{
21         Debug.Log("The polar bear goes in the ocean");
22     }
23 }
24
25 }
```

1. **theSeasonOfTheYear** adında yeni bir değişken oluşturup türünü **String** (Yazı) olarak ayarladım ve değerini de “**summer**” (yaz mevsimi) olarak **atadım (assign)**.

2. **theSeasonOfTheYear** değişkenini koşul (**condition**) olarak sırayan yeni bir **if ifadesi (statement)** oluşturdum.

3. İki adet eşittir işareti (**==**) kullandığımı dikkat edin. İşte bu sorgularda “**Eşitse**” anlamında kullanılır.

Peki nasıl **theSeasonOfTheYear == "summer"** ifadesinin sonucu doğru (**true**) oldu? Bunu basit bir örnekle açıklayayım:

6 == 6

Bunun anlamı 6 “**eşitse**” 6. Peki sizce bu **true** (doğru) bir ifade midir? Elbette. Peki bir de şuna bakın:

“**summer**” == “**summer**”

Bu da eşittir, değil mi? Peki, söyle misiniz **theSeasonOfTheYear** değişkeninde hangi veriyi (değer) depolamıştık? 1. maddeye bakarsak bunun “**summer**” yazısı olduğunu görebiliriz. O zaman;

theSeasonOfTheYear == "summer"

ifadesi de doğru (**true**) bir ifadedir.

Yani anlayacağınız üzere **if** ifadesi içerisinde herhangi türde değişken koşul olarak kullanılabilir ve bu koşulun değerini test etmek için de (**==**) işaretini kullanılır.

Not:

theSeasonOfTheYear değişkeninin değeri ayrıca **Inspector** panelinde de gözükmektedir. Eğer ki bu değeri değiştirip “*summer*” haricinde bir şey yaparsanız **if** ifadesindeki koşul tutmaz ve sonuç olarak kodların hiç biri çalıştırılmaz ve konsol ekranında da hiçbir çıktı alınmaz. Neden peki?

Çünkü 2. maddede oluşturduğumuz **if** ifadesi tek başına bir ifadedir, bir **if-else** ifadesi değildir. Bu yüzden koşullar tutmazsa çalıştırılacak alternatif bir kod da belirtilememiştir.

Başka Bir Not:

Tüm bu örnekler aslında gayet basitti, ancak gördüğünüz üzere eğer çok fazla sayıda **if-else** kullanmaya başlarsanız epey karmaşık görünen kodlar elde edersiniz, oysaki kodlarınız oldukça basit olmasına rağmen. Ben script dosyamın içerisinde herhangi bir yorum (comment) ifadesi kullanmadım (İki adet Slash ‘//’ işaretiyile başlayıp ardından script içerisinde çalıştırılmayan, sadece yorum eklenmesini sağlayan yazılar.). Ancak sizler kendi kodlarınızda bolca yorum eklemesi yapın, böylece uzunca bir süre sonra kodu açtığınızda kodun içerisinde nelerin döndüğünü anlamak için kafanızı yerden yere vurmazsınız.

Dot Syntax (Noktasal Yapı): Component'ler (Parça) Arası İletişim

Scriptten dışarıyıla iletişim

Scriptler pek çok şey yapabilir ve bir scriptin asıl gücü, oyuncuya ya da başka **GameObject**'lerle teması geçince bir şeyler yaptırımla ortaya çıkar. Bu yüzden **GameObject**'lerin kendi aralarında iletişim kurmalarının bir yolu olmalı, değil mi? İşte burada devreye **Dot Syntax (Noktasal Yapı)** girer.

Dot Syntax bir adresi temsil eder

Konuya devam etmeden önce kılavuzun başlarındaki, bu konuya ilgili yaptığımız ilk bakış dosyasına bakmanızı tavsiye ederim.

Dot Syntax İşleminin Basitliği (Yalınlığı)

Terry Norton
22 myStreet
Essex, VT

Post Office Syntax



Dot Syntax



Postaneden gelen mektuplarınızda yazılması zorunlu olan adres formatına epey alışıkınızdır heralde.

UnityScript'in de uyulması gereken bir adres formatı vardır. Bu formatla başka **component**'lerdeki (parça) **değişkenlere (variable)**, **fonsiyonlara (function)** kolayca erişebilirsiniz. Ne zaman ki **Dot Syntax**'ı kullanıyorsunuz, o zaman başka bir scriptteki bir değişkene ya da fonksiyona ulaşmaya çalışıyorsunuz demektir.

Bir script'te Dot Syntax'ın kullanımı

```
private var spin : boolean = true;
private var otherObject : GameObject;
otherObject = GameObject.Find("Capsule");

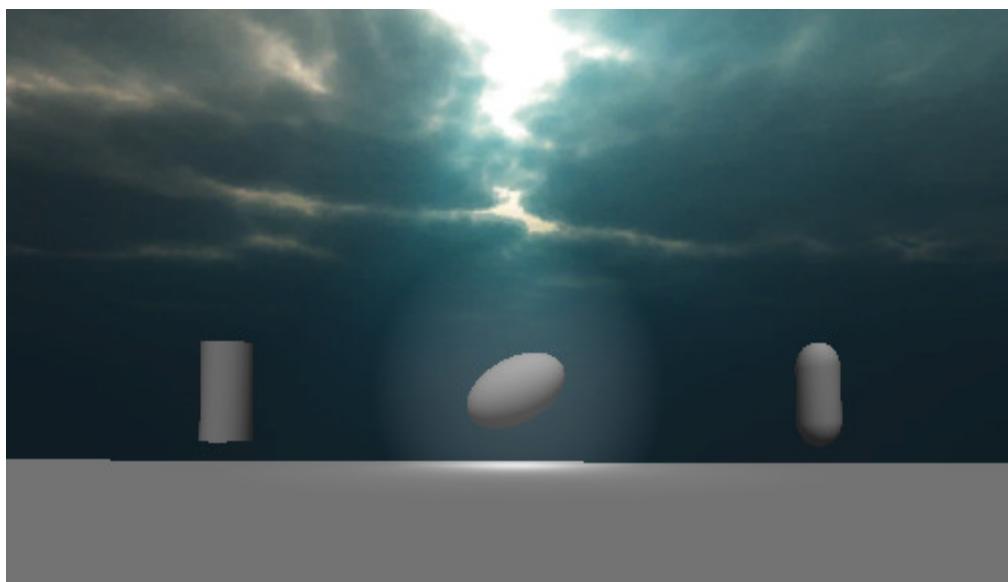
function Update () {
    if(spin){
        light.enabled = true;
        transform.Rotate(0,0,60 * Time.deltaTime);
        if(transform.eulerAngles.z >= 358.5) {
            spin = false;
            light.enabled = false;
            transform.eulerAngles = Vector3(0,0,0);
            otherObject.GetComponent(TwirlCapsule).spin = true;
        }
    }
}
```

1. Burada gördüğünüz script, bir **GameObject**'i kendi ekseni etrafında bir tam tur döndürmeye yarayan 3 scriptimden birisidir. Script tamamlandığında, bu sefer başka bir **GameObject**'in kendi etrafında dönmesini sağlar. Bunu da diğer **GameObject**'teki bir değişkene ulaşıp onun değerini değiştirmektedir.

2. Burası da az önce bahsettiğim 3 scriptin olduğu kısım. Her bir script farklı bir **GameObject**'in dönmesini sağlıyor. Bu objeler ise: bir yumurta (egg), bir kapsül (capsule) ve bir silindir (Cylinder). Bunlar; sırı eğlence için, kullandığım geometrik şekillerin isimleri kullanılarak isimlendirildi.

Sizin de gördüğünüz üzere bu scriptin büyük bir kısmında **Dot Syntax** iletişimini kullanılarak çeşitli değişkenler okunmakta (**read**), değişmekte ve fonksiyonlar çağrılmaktadır (**call**). Benim yazdığım 3 script de birbirine oldukça benzer, aralarında sadece birkaç basit farklılık bulunmaktadır.

Projeden örnek bir görüntü



Bu görüntü Game (Oyun) penceresinde, oyunum çalışırken alınmıştır. Şu an yumurta (egg) dönmektedir. Her bir GameObject bir kere sırayla döner ve ardından işlem tekrar başlar. Ayrıca dönmeye olan objenin etrafında bir ışık halesi oluşur.

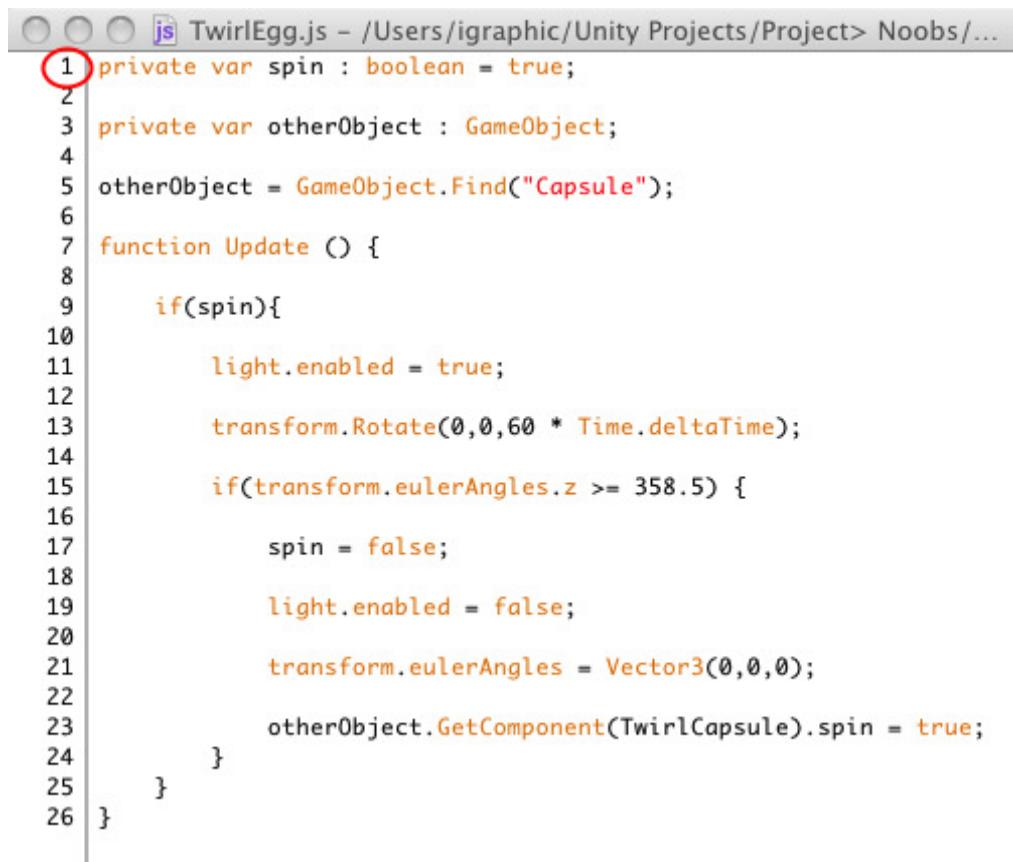
Eğer bu işlemi bir video olarak görmek isterseniz şu linke tıklayın:

http://www.youtube.com/watch?v=aNfz1ID_v2w

Yumurta (Egg) scripti, 1 - 5 arası satırlar

Şimdi scriptimdeki tüm satırları tek tek, nasıl işlediğini açıklayarak anlatacağım.

1. Satır: Kontrol değişkenini ayarlamak



```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10
11         light.enabled = true;
12
13         transform.Rotate(0,0,60 * Time.deltaTime);
14
15         if(transform.eulerAngles.z >= 358.5) {
16
17             spin = false;
18
19             light.enabled = false;
20
21             transform.eulerAngles = Vector3(0,0,0);
22
23             otherObject.GetComponent(TwirlCapsule).spin = true;
24         }
25     }
26 }
```

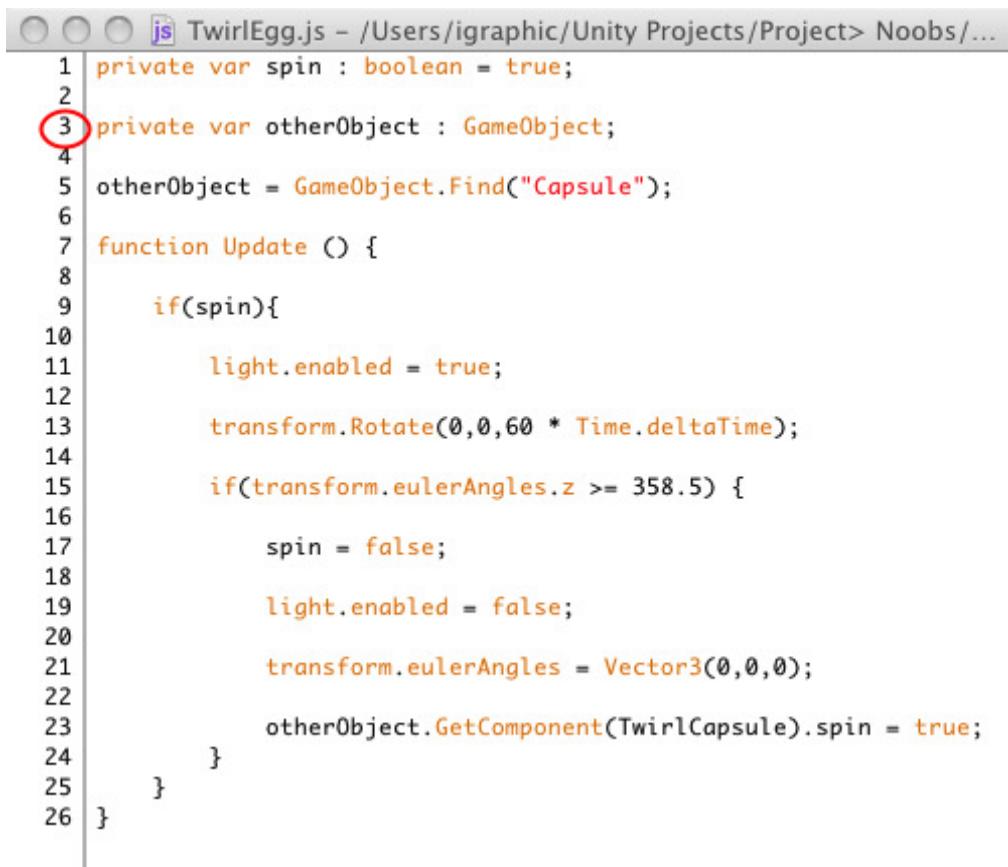
Bu **spin** (donme) adındaki **private** (özel) bir değişkendir. Türü **boolean**'dır, yani değeri ya **true** ya da **false** olabilir.

spin değişkeninin yaptığı şey Egg ismindeki yumurta objesinin dönmesinin müsait olup olmadığını kontrol etmektir. Eğer değeri **true** ise yumurta dönecektir, **false** ise dönmeyecektir.

spin'in değerini **true** olarak ayarladım çünkü oyun başladığında ilk dönen objenin yumurta olmasını istiyorum.

Bu bir **private** (özel) değişkendir, yani değeri **Inspector** panelinde gözükmekz. İsterseniz başındaki '**private**' takısını silerek değerini **Inspector**'dan izleyebilirsiniz.

3. Satır: Sonraki dönecek GameObject'i depolayacak bir yer



```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10
11         light.enabled = true;
12
13         transform.Rotate(0,0,60 * Time.deltaTime);
14
15         if(transform.eulerAngles.z >= 358.5) {
16
17             spin = false;
18
19             light.enabled = false;
20
21             transform.eulerAngles = Vector3(0,0,0);
22
23             otherObject.GetComponent(TwirlCapsule).spin = true;
24         }
25     }
26 }
```

Bu **otherObject** (digerObj) adındaki **private** bir değişken. Bu değişkenin türü gördüğünüz üzere **GameObject**. Bunun anlamı bu değişken, yumurtadan sonra dönecek olan **GameObject**'in ismini depolayacaktır.

Bu **private** bir değişken olduğu için değeri **Inspector**'da gözükmeyecektir, ancak diğer 2 scriptte bu değişken **private** değildir. Bunu sonra açıklayacağım.

5. Satır: sonraki dönecek GameObject'i ayarlamak

```

1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10
11         light.enabled = true;
12
13         transform.Rotate(0,0,60 * Time.deltaTime);
14
15         if(transform.eulerAngles.z >= 358.5) {
16
17             spin = false;
18
19             light.enabled = false;
20
21             transform.eulerAngles = Vector3(0,0,0);
22
23             otherObject.GetComponent(TwirlCapsule).spin = true;
24         }
25     }
26 }

```

Burada otherObject değişkenimize **Capsule** GameObject'i atıyor.

Ayrıca ilk **Dot Syntax**'ımızı da burada görüyoruz: **GameObject.Find()**

([Buraya tıklayarak](#)) Unity'nin web sitesinde yer alan Script referans dosyasından, bununla ilgili bilgi bulabilirsiniz.

GameObject.Find()

The screenshot shows the Unity documentation interface. The top navigation bar includes links for Unity, Gallery, Store, Support, and Company. Below the navigation bar, there's a secondary menu with Documentation, Resources, and Community. Under the Documentation link, it says "Scripting > Runtime Classes > GameObject". On the left side, there's a "Search" bar and a "Menu" section with Overview, Runtime Classes, and Attributes. The main content area is titled "GameObject.Find" and describes it as a static function that takes a string parameter and returns a GameObject. The description states: "Finds a game object by name and returns it."

GameObject.Find
static function Find (name : string) : GameObject
Description
Finds a game object by name and returns it.

Peki bu şey ne işe yarar? Eğer referans dosyasına bakarsanız göreceğiniz üzere burada kullandığımız fonksiyon; içerisinde yazdığımız isme sahip bir **Game Object** bulup onu geri döndürmeye (**return**) (Rotate değil! Programlama dilinde yer alan döndürmek) yarar. Bizim de istediğimiz tam olarak bu. Yumurtadan sonra donecek GameObject'i belirlemek zorundaydık ve de belirlemiş olduk.

Peki niçin **Find()** fonksiyonunun başında bir '**GameObject**' takısı ve bir **nokta '.'** olması gereklidir?

Çünkü **Find()** fonksiyonu TwirlEgg scriptimizde tanımladığımız bir fonksiyon değildir, bu yüzden bu fonksiyonun tanımlandığı konumu bulmak zorundayız. Bu fonksiyon da otomatik olarak Unity tarafından bir yerlerde kodlanmıştır. Hatırlayın, **Dot Syntax** bir adres formatıdır, yani bize **Find()** fonksiyonunun nerede bulunduğu söylmektedir.

Pekâlâ, Unity'nin referans dosyasının dediğine göre bu fonksiyon **GameObject class**'ının (sınıf) içerisinde bulunuyor. **Bu, Find() fonksiyonu GameObject isimli scriptin içerisinde tanımlanmıştır (declare) demekle aynı anlama gelmektedir.** Peki bu GameObject bir script mi? Bilmiyorum, umursamıyorum da. Bana tek gereken bu fonksiyonun hangi class'ta yer aldığıydı ve referans dosyası da bunu bana söylüyor.

Elbette akınıza takılmıştır, **Find()** fonksiyonu için ilk önce **GameObject** classının içerisinde bakacağımızı nereden bileyceğiz? TwirlEgg scriptimin 23. satırındaki olayla aynı sebepten. Bir başka **GameObject**'te yer alan bir başka **Component**'teki bir değişkeni değiştirmeliyim. Bunun anlamı bu fonksiyon için **GameObject** class'ına bakmalıyım.

(Çevirmen Eklemesi) : Bir önceki paragrafi anlamazsanız dert etmeyin çünkü o paragrafta yazar çok iyi bir anlatım sergileyememiş. Çok anlaşılır değil, ben de tam anlamadım ne dediğini.

Peki o zaman, nasıl **Find()** fonksiyonunu kullanacağımı bildim? Elbette ki **GameObject** class'ı içerisinde yer alan çeşitli kodların açıklamalarını okuyarak.

Kullanabileceğim **GameObject** değişkenleri ve fonksiyonları

Class Functions

CreatePrimitive	Creates a game object with a primitive mesh renderer and appropriate collider.
FindWithTag	Returns one active GameObject tagged tag. Returns null if no GameObject was found.
FindGameObjectsWithTag	Returns a list of active GameObjects tagged tag. Returns null if no GameObject was found.
Find	Finds a game object by name and returns it.

[\(Buraya tıklayarak\)](#) referans dosyasındaki **GameObject** sayfasına gidebilirsiniz. Sayfanın ortalarında **Class Functions** (Sınıf Fonksiyonları) kısmını göreceksiniz. İşte **Find()** fonksiyonu burada ve yanında da kısa bir açıklaması var. Tek yapmam gereken bu referans dosyasına bakıp istediğime uygun bir şey bulana kadar araştırmaktı.

GameObject.Find() fonksiyonunu nasıl kullanırıım?

The screenshot shows the Unity documentation website. At the top, there's a navigation bar with links for Unity, Gallery, Store, Support, and Company. Below that is a secondary navigation bar with Documentation, Resources, and Community. Underneath those, a breadcrumb trail shows Scripting > Runtime Classes > GameObject. On the left, there's a sidebar with a search bar and links for Menu, Overview, Runtime Classes, Attributes, and Examples. The main content area has a title "GameObject.Find" and a subtitle "static function Find (name : string) : GameObject". A red box highlights this subtitle. Below it is a "Description" section with the text "Finds a game object by name and returns it."

Burada ilk gördüğüm şey bu fonksiyonun **static** (durgun, değişmez) bir fonksiyon olduğunu. Bunun anlamı bu bir **global** (evrensel) fonksiyondur ve kendi oluşturduğum herhangi bir scriptten bu fonksiyona erişim sağlayabilirim.

Ardından fonksiyonun adını görüyorum: **Find()** “Bul()”. Parantezlerin içerisinde de bulmak istediğimiz **GameObject**’in ismini (name) yazıyoruz. Bizim istediğim **GameObject**’in adı **Capsule**.

Ayrıca **name** değişkeninin bir **String** olduğuna dikkat edin. Yani objemizin ismini yazarken iki adet tırnak işaretini (“ ”) arasına yazmalıyız.

Son olarak görüyorum ki bir ":" (iki nokta) işaret var ve ardında da **GameObject** yazıyor. UnityScript’ın size ne tür bir veri döndüreceğini buradan görürsünüz. Yani bu fonksiyon bir **GameObject**’i döndürüyor (**return**). **GameObject.Find()** fonksiyonunu çağrılmamızın bütün anlamı burada, **Capsule** isimli **GameObject**’i bulup **otherObject** isimli değişkenin içerisinde depolamak için.

Referans dosyasında GameObject.Find() ile ilgili daha çok bilgi

For performance reasons it is recommended to not use this function every frame instead cache the result in a member variable at startup or use **GameObject.FindWithTag**.

JavaScript ▾

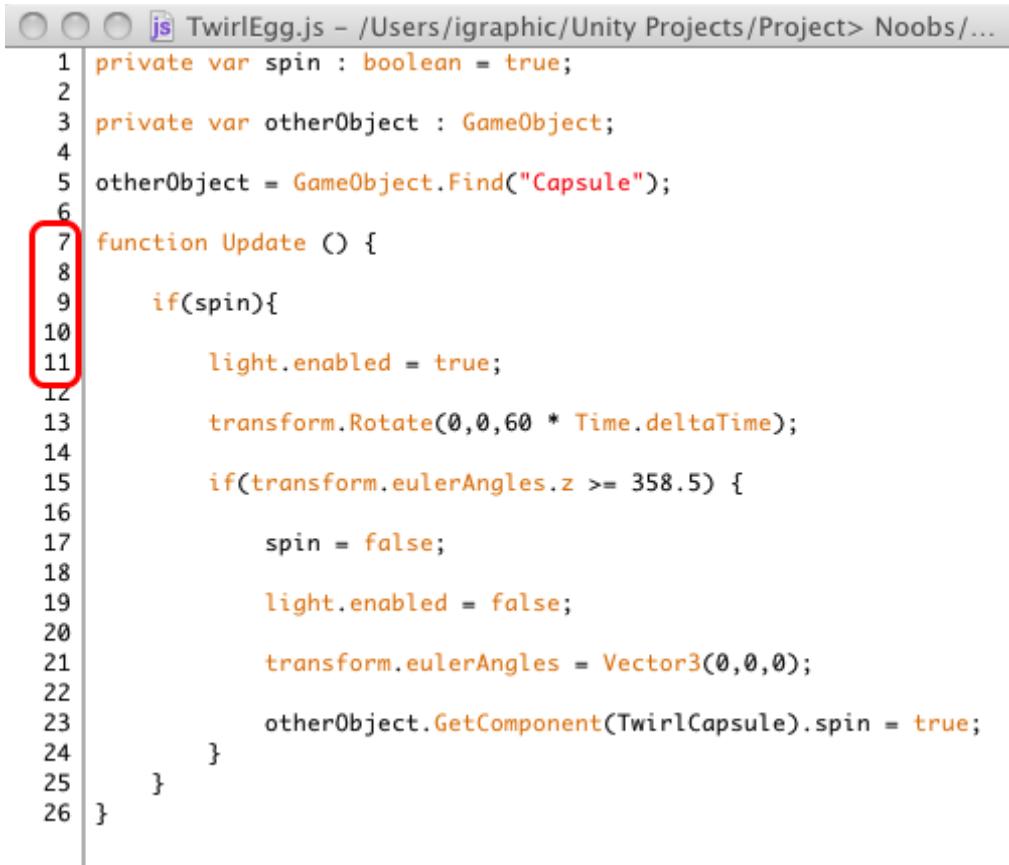
```
var hand : GameObject;
// This will return the game object named Hand in the scene.
hand = GameObject.Find("Hand");
```

Burada referans dosyası bize, bu **GameObject.Find()** fonksiyonunu **Update()** fonksiyonu içerisinde kullanmamamızı tavsiye ediyor. Böylece oyunumuzdan daha çok performans alacağımızı söylüyor.

Sonrasında da bir örnek gösteriyor. Bu örnek tam da bizim kullandığımız kodlarla aynı, bir tek değişken ismi ve **GameObject**’in ismi farklı.

Yumurta (Egg) scripti, 7 – 11 arası satırlar, Update() fonksiyonu

Update() fonksiyonu asıl aksiyonun yapıldığı kısımdır



```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10
11         light.enabled = true;
12
13         transform.Rotate(0,0,60 * Time.deltaTime);
14
15         if(transform.eulerAngles.z >= 358.5) {
16
17             spin = false;
18
19             light.enabled = false;
20
21             transform.eulerAngles = Vector3(0,0,0);
22
23             otherObject.GetComponent(TwirlCapsule).spin = true;
24         }
25     }
26 }
```

7. Satır:

Buraya yazdığımız **Update()** fonksiyonunun içerişine yazdığımız kodlar oyun sırasında Unity tarafından her bir karede otomatik olarak çalıştırılmaktadır.

9. Satır:

Bu **if** ifadesi **spin** değişkeninin değerinin **true** (doğru) ya da **false** (yanlış) olduğunu test ediyor. Bu değişken 1. satırda **true** olarak tanımlanmıştı (**declare**), bunun anlamı biz **spin** değişkeninin değerini **false** olarak ayarlayana kadar da oyun sırasında her karede buraya yazacağımız kodlar tekrar ve tekrar çalıştırılacaktır (**execute**). (Değişkeni 17. satırda **false** yapmaktadır.)

Başlangıçta **spin** değişkeninin değerini **true** olarak ayarlamamın sebebi, oyun başladığında elimdeki 3 adet **GameObject**'ten tekinin ilk başta dönmesinin gereklisiydi ve bu obje de yumurta (egg) olacak. Eğer tüm 3 scriptimdeki bu değişkenin değerini **false** olarak ayarlasaydım o zaman oyunumu çalıştırıldığında hiçbir şey gerçekleşmezdi.

spin bir mekanizmanın düğmesi gibi. Açık olduğunda (**true**) yumurta döner (**rotate**), kapattığımızdaysa (**false**) dönmeyi durdurur.

11. Satır:

Burada bir başka **Dot Syntax**'la karşılaştık.

11. Satır: Light (Işık) component'inin (parça) açılması (aktive edilmesi)

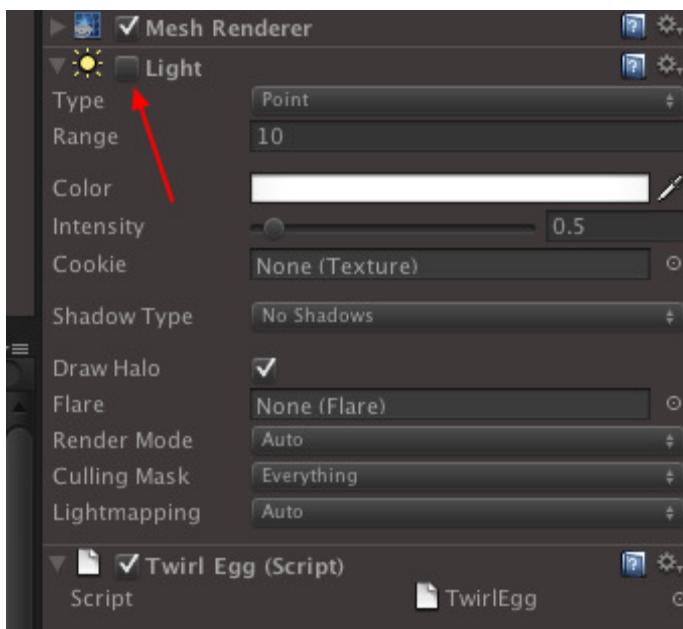
Light Inherits from **Behaviour**
Script interface for [light components](#).

Use this to control all aspects of Unity's lights. The properties are an exact match for the values shown in the Inspector.

Usually lights are just created in the editor but sometimes you want to create a light from a script:

(Buraya tıklayarak) Unity referans dosyasındaki Light classını görebilirsiniz. Sayfanın yarısına doğru **Inherited Variables** (Kalıtım (Miras) Değişkenler) kısmını görebilirsiniz. Buradaki **enabled** değişkeni TwirlEgg scriptindeki tüm ışık faaliyetlerini açmamıza ya da kapamamıza yaramaktadır.

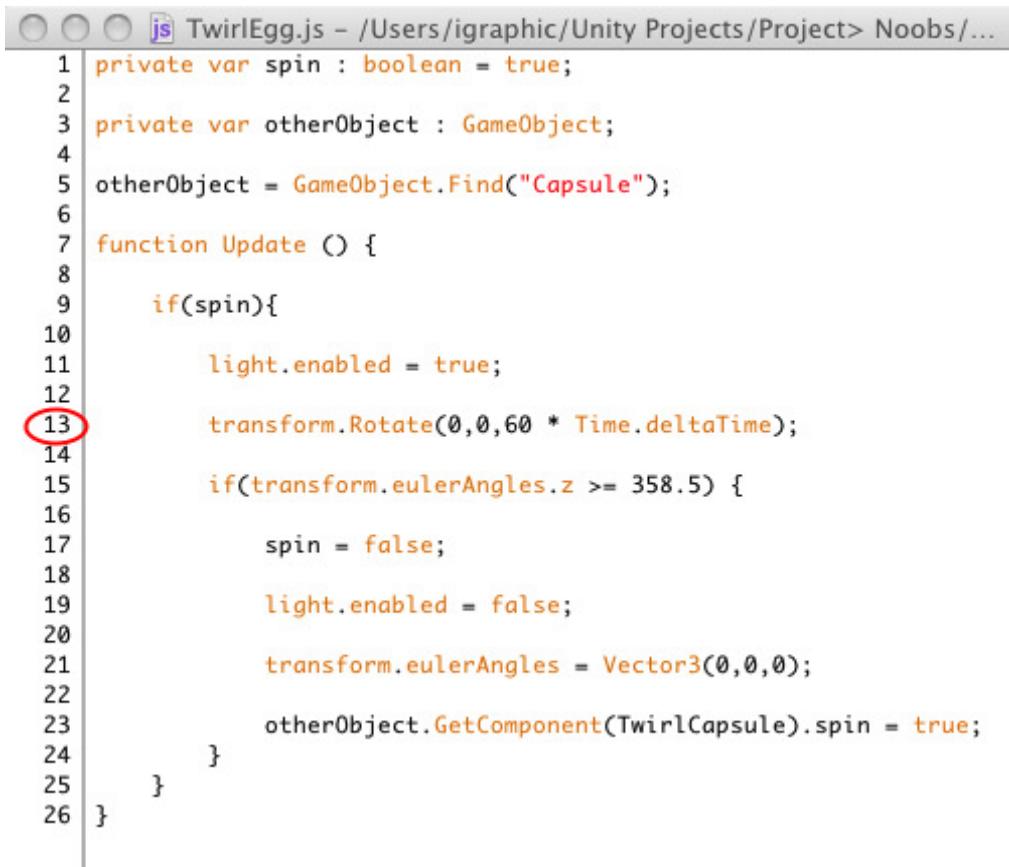
Light (Işık) işaret kutucuğu (checkbox)



Bu, 11. satırda **enable** (açık) hâle getirdiğimiz **Light** checkbox'ı. Oyun boyunca yumurta dönerken ve dururken bu kutucuğun işaretinin sürekli değiştiğini görürsünüz.

Yumurta (Egg) scripti, 13. satır, Yumurtayı Döndürmek

Yumurtayı Rotate() fonksiyonu ile döndürmek (Rotate)



```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10
11         light.enabled = true;
12
13         transform.Rotate(0,0,60 * Time.deltaTime);
14
15         if(transform.eulerAngles.z >= 358.5) {
16
17             spin = false;
18
19             light.enabled = false;
20
21             transform.eulerAngles = Vector3(0,0,0);
22
23             otherObject.GetComponent(TwirlCapsule).spin = true;
24         }
25     }
26 }
```

Tekrar **Dot Syntax** kullanıyorum, bu sefer **Transform class**'ında (Buna uygun bir Türkçe karşılık bulamıyorum. Ancak bu çok temel bir class'tır ve bununla objenin konumunu, eğimini (açısını)(rotation) değiştirebiliriz.) bir fonksiyona erişmek için kullanıyorum.

Dot Syntax'ı kullandık, çünkü **Rotate()** fonksiyonu TwirlEgg scriptimiz içerisinde tanımlanmış bir şey değil.

Unity bunu da başka bir yerde kodlamış, kullanımına hazır olarak bulundurmaktadır.

Egg (Yumurta) GameObject'ının Transform componenti (Parça)

Transform.Rotate

```
function Rotate (eulerAngles : Vector3, relativeTo : Space = Space.Self) : void
```

Description

Applies a rotation of eulerAngles.z degrees around the z axis, eulerAngles.x degrees around the x axis, and eulerAngles.y degrees around the y axis (in that order).

If relativeTo is left out or set to **Space.Self** the rotation is applied around the transform's local axes. (The x, y and z axes shown when selecting the object inside the Scene View.) If relativeTo is **Space.World** the rotation is applied around the world x, y, z axes.

JavaScript ▾

```
function Update() {
    // Slowly rotate the object around its X axis at 1 degree/second.
    transform.Rotate(Vector3.right * Time.deltaTime);

    // ... at the same time as spinning relative to the global
    // Y axis at the same speed.
    transform.Rotate(Vector3.up * Time.deltaTime, Space.World);
}
```

1

```
function Rotate (xAngle : float, yAngle : float, zAngle : float, relativeTo : Space = Space.Self) : void
```

Description

Applies a rotation of zAngle degrees around the z axis, xAngle degrees around the x axis, and yAngle degrees around the y axis (in that order).

If relativeTo is left out or set to **Space.Self** the rotation is applied around the transform's local axes. (The x, y and z axes shown when selecting the object inside the Scene View.) If relativeTo is **Space.World** the rotation is applied around the world x, y, z axes.

2

JavaScript ▾

```
function Update() {
    // Slowly rotate the object around its X axis at 1 degree/second.
    transform.Rotate(Time.deltaTime, 0, 0);
```

Her **GameObject**'in bir **Transform** componenti vardır. Bu component onların sahnede nerede bulunduklarını ve hangi eğimle durduklarını belirler. Bizim tek yapmamız gereken **Transform** componentinde yer alan **Rotate()** fonksiyonunu çağırırmak (call). Bunun için de **Rotate()** fonksiyonunun başına “**transform**” takısı ve bir nokta “.” ekliyoruz. Böylece kodumuz **Dot Syntax**'a uyumlu olarak, sorunsuz bir şekilde çalışacak.

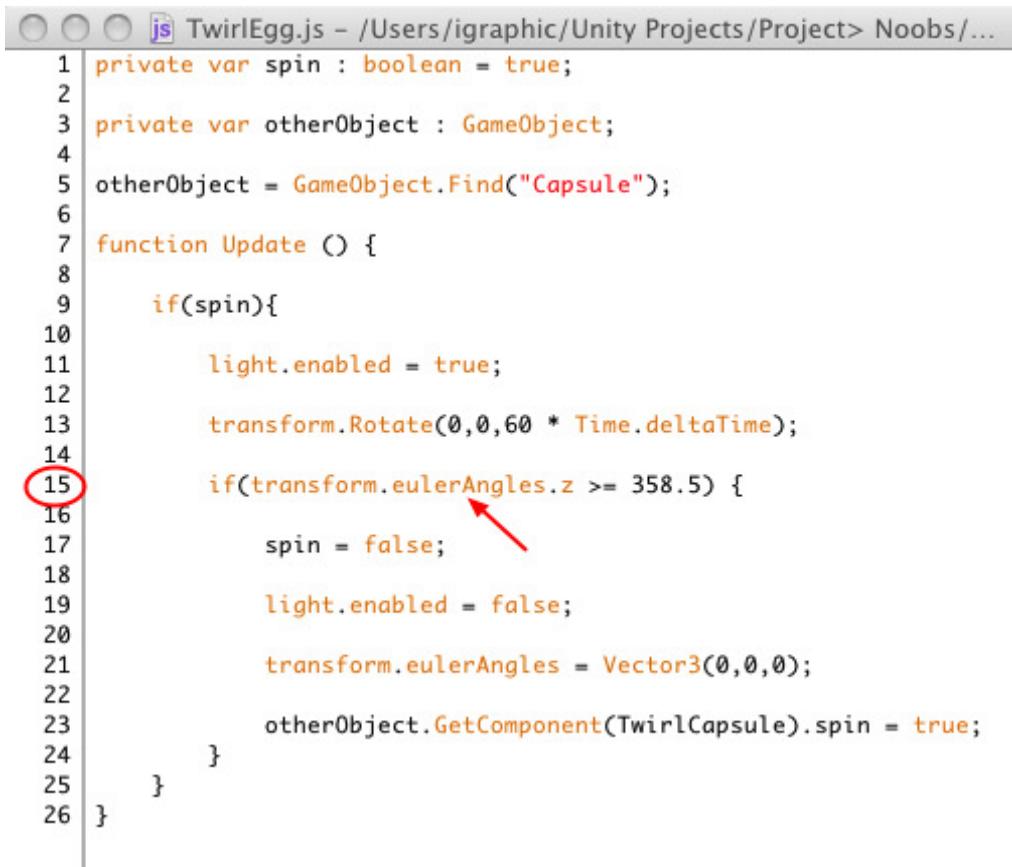
1. `Rotate()` fonksiyonu için parametreleri (**parameter**) gösterir.

2. Fonksiyonun kullanımına bir örnek verilmiş burada. Benim yaptığıma oldukça yakın bir işlem yapılmış bu örnekte de. 13. satırda, **Z** ekseni (**axis**) etrafında saniyede 60 derecelik bir dönüş (**rotate**) sağlıyorum.

Buradaki **Time.deltaTime**'ın anlamı ise "1 saniye"dir. Yani 13. satırda söylediğim şey yumurtanın **saniyede** 60 derece dönmESİdir (**Her bir karede değil!**). Yani yumurta 6 saniyede 360 derecelik tam bir tur atmış olacak.

Yumurta (Egg) scripti, 15. satır, Eğimi (Rotation) İzlemek

15. satır yumurtanın (Egg) kaç derece döndüğünü göstermektedir



```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10
11         light.enabled = true;
12
13         transform.Rotate(0,0,60 * Time.deltaTime);
14
15     if(transform.eulerAngles.z >= 358.5) {
16
17         spin = false;
18
19         light.enabled = false;
20
21         transform.eulerAngles = Vector3(0,0,0);
22
23         otherObject.GetComponent(TwirlCapsule).spin = true;
24     }
25 }
26 }
```

Bu yumurtanın kaç derece döndüğüne bakan bir **if** ifadesidir (**statement**).

Elbette ki eğimin depolandığı yer de yine **Transform** component'ıdır. **eulerAngles**'e gelirsek... Nedir bu **eulerAngles**??

Bunun için Wikipedia'ya bakmak zorunda kaldım. Aldığım bilgiye göre diyor ki “**Euler angles** (Euler açıları??) bir **rigidbody**'nin (sert gövde, yani bir objeye ağırlık veren parça) **eğimini** (**orientation**) açıklamak için **Leonhard Euler** tarafından keşfedilmiştir.”. Bu bilgi oldukça açık, değil mi? Ardından Unity kılavuzundaki birkaç örneğe baktım ve bu ‘şey’in (Değişken? Evet.) tam istediğim görevi yerine getirdiğini anladım.

eulerAngles her eksendeki (axis) eğimi depolar.

Transform.eulerAngles

```
var eulerAngles : Vector3
```

Description

The rotation as Euler angles in degrees.

1

The x, y, and z angles represent a rotation z degrees around the z axis, x degrees around the x axis, and y degrees around the y axis (in that order).

2

Only use this variable to read and set the angles to absolute values. Don't increment them, as it will fail when the angle exceeds 360 degrees. Use **Transform.Rotate** instead.

JavaScript ▾

```
// Print the rotation around the global X Axis  
print (transform.eulerAngles.x);  
// Print the rotation around the global Y Axis  
print (transform.eulerAngles.y);  
// Print the rotation around the global Z Axis  
print (transform.eulerAngles.z);  
  
// Assign an absolute rotation using eulerAngles  
var yRotation : float = 5.0;  
function Update () {  
    yRotation += Input.GetAxis("Horizontal");  
    transform.eulerAngles = Vector3(10, yRotation, 0);  
}
```

3

Do not set one of the eulerAngles axis separately (eg. `eulerAngles.x = 10;`) since this will lead to drift and undesired rotations. When setting them to a new value set them all at once as shown above. Unity will convert the angles to and from the rotation stored in **Transform.rotation**.

1. Tam bilmek istediğim şey, tüm eksenlerdeki (Benim istediğim **Z** ekseni de dahil) yapılan açı depolanmış.
2. Bu değişken sadece değeri okumak içinmiş (read-only), yani değerini değiştirmek için değil. Ben de böyle bir şey düşünmüyorum zaten.
3. Bu örnek benim **if** ifademde yapmaya çalıştığım şeye pek benzemiyor, ancak madem ki bu değişkenin değerini okuyabiliyorum, o zaman ayrıca bunu **Z** eksenindeki açının 358.5 dereceden fazla olup olmadığını test etmek için de kullanabilirim.

Peki niçin 360 derece değil de 358.5 derece? Çünkü 360 derece 0 dereceye eşittir, o yüzden 359 dereceyle test yapabilirdim. Ancak sonradan fark ettim ki rotasyon tam sayılar hâlinde gerçekleşmiyor. Bir miktar kesirli olarak ilerliyor. Bunun nedeni eminim **Time.deltaTime** ifadesini kullanmadır. Bu yüzden de neredeyse 359 derece olan bir değer seçtim. Bu çok bilimsel bir şey olmadığı için 358.5 dereceyi seçtim ve işe yaradı.

Eğim (Rotation) 358.5 dereceye veya daha yukarısına gelince **if** ifadesi (**statement**) **true** (doğru) olmakta ve içerisindeki kod(lar) gerçekleştirilmektedir (**execute**).

(Çevirmen Eklemesi) : Yazar sağolsun burada da çok açıklayıcı bir anlatım sergilemiyor. Burada anlatmaya çalıştığı şey şu: Objemizin Z eksenindeki eğimi 360 dereceye veya yukarısına çıkarsa Unity bu değerden otomatik olarak 360 çıkarır, mesela açımız 363 ise Unity hemen bunu 3 olarak ayarlar. Anlayacağınız bir objenin açısı hiçbir zaman 360 derece veya yukarısı olamaz. Bu yüzden 359 dereceyi deneyebilirdik demiş ama olmadı demiş. Burada olmamasının sebebi ise şu: Obje saniyede 60 derece dönecek şekilde ayarlanmış, bunu her karede hesaplayan Unity; bir karede döndürülecek açı değerini 1 dereceden fazla buluyor. Bu yüzden de çoğu zaman 359 derece tutmaz. 358.5 dereceyi denediğinde ise tutmuş, bu bir deneme yanılma meselesi. Ancak örneğin objemiz saniyede 180 derece dönseydi o zaman 358.5'de büyük ihtimalle tutmazdı, 355 gibi bir değerle test etmek zorunda kalırdık. Umarım bu konu anlaşılmıştır.

Yumurta (Egg) scripti, 17 – 23 arası satırlar, Döndürmeyi Durdurmak

Artık Egg (yumurta) objesi 1 tam tur attığına göre onu döndürmeyi durdurmanın ve sonraki objeyi döndürmeye başlamadanın vakti geldi.

17 – 19 arası satırlar: Döndürmeyi durdur ve ışığı kapat



```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10
11         light.enabled = true;
12
13         transform.Rotate(0,0,60 * Time.deltaTime);
14
15         if(transform.eulerAngles.z >= 358.5) {
16
17             spin = false;
18
19             light.enabled = false;
20
21             transform.eulerAngles = Vector3(0,0,0);
22
23             otherObject.GetComponent(TwirlCapsule).spin = true;
24         }
25     }
26 }
```

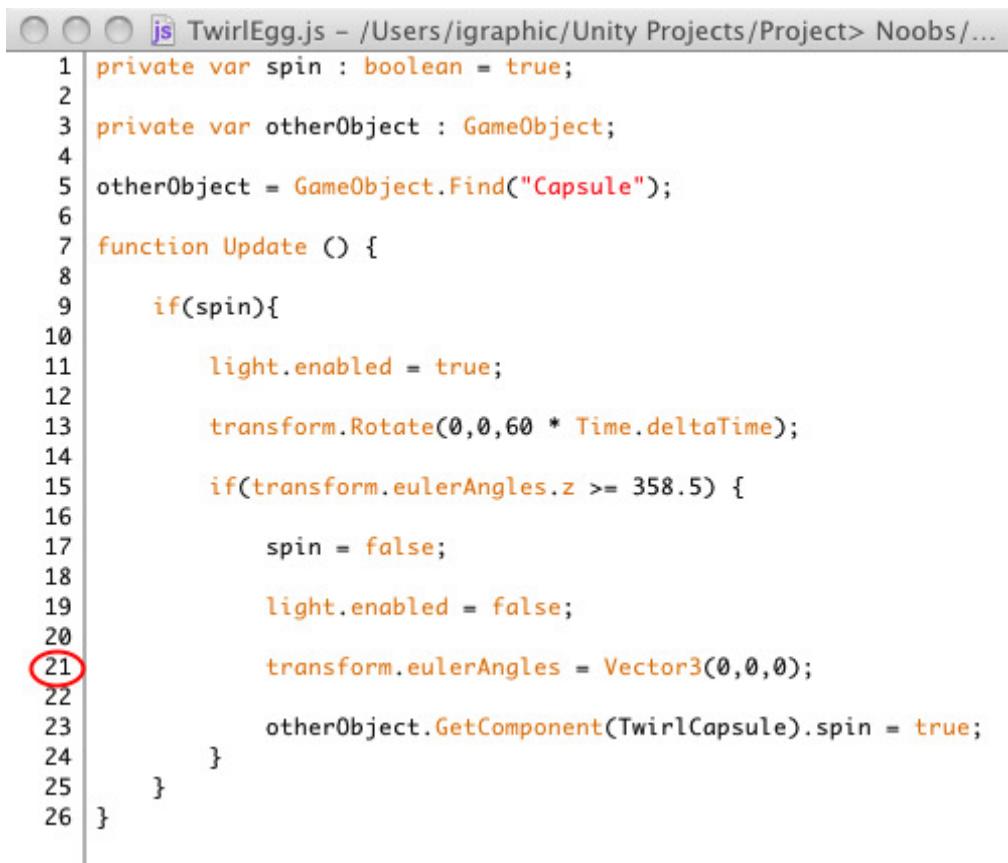
17. Satır:

9. satırda **spin** değişkeninin değeri **true** idi. Böylece yumurta objesi dönüyordu ve etrafında bir ışık halesi (**halo**) bulunuyordu. Şimdi bu **spin** değişkeninin değerini **false** olarak ayarladım.

19. Satır:

11. satırda ışığı aktif etmiştık. Şimdi bu değerini **false** olarak ayarlayarak ışığı kapatıyoruz, böylece objenin etrafındaki hale yok olacak.

21. Satır



```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10
11         light.enabled = true;
12
13         transform.Rotate(0,0,60 * Time.deltaTime);
14
15         if(transform.eulerAngles.z >= 358.5) {
16
17             spin = false;
18
19             light.enabled = false;
20
21             transform.eulerAngles = Vector3(0,0,0);
22
23             otherObject.GetComponent(TwirlCapsule).spin = true;
24         }
25     }
26 }
```

İşte tekrar **eulerAngles** ile karşılaştık. Bu kısımda; **Z** eksenindeki (**axis**) eğimi başlangıç açısı olan 0 dereceye resetliyoruz.

Açayı 0 dereceye resetlemek

```
// Assign an absolute rotation using eulerAngles
var yRotation : float = 5.0;
function Update () {
    vRotation += Input.GetAxis("Horizontal");
    transform.eulerAngles = Vector3(10, yRotation, 0);
}
```

1

2

Do not set one of the eulerAngles axis separately (eg. eulerAngles.x = 10;) since this will lead to drift and undesired rotations. When setting them to a new value set them all at once as shown above. Unity will convert the angles to and from the rotation stored in `Transform.rotation`

1. Bu az önceki gösterdiğim dökümanla aynı yer. Bu sefer sadece ihtiyacımız olan kısmını kesip gösteriyorum. Bu tüm 3 eksendeki (x,y,z) açıları da belli birer değere ayarlamamızı sağlayan kodun örnek bir kullanımı. Tümünün de 0'a ayarlandığına emin oluyorum.
2. Burada açıları ayarlamadan önce bir uyarı verilmiş. Dediğine göre açıları tek tek değiştirmeye kalkarsak istenmeyen sonuçlar alabilirmiştir. Ben de bu bölümü okumadan önce sırf Z eksenindeki açıyı değiştirmiştim, aslında işe yaramıştı ancak sonradan kodu burada tavsiye edildiği gibi düzelttim. (Kodu 21. satırda görebilirsiniz.)

23. Satır: Sonraki GameObject'e dönmeye başlamasını söylemek

```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10
11         light.enabled = true;
12
13         transform.Rotate(0,0,60 * Time.deltaTime);
14
15         if(transform.eulerAngles.z >= 358.5) {
16
17             spin = false;
18
19             light.enabled = false;
20
21             transform.eulerAngles = Vector3(0,0,0);
22
23             otherObject.GetComponent(TwirlCapsule).spin = true;
24
25         }
26     }
}
```

The code shows a conditional block where `spin` is checked. If true, it sets `light.enabled` to true, rotates the transform, and then checks if the z-euler angle is greater than or equal to 358.5. If so, it sets `spin` to false, turns off the light, and resets the transform's euler angles. Finally, at line 23, it uses dot syntax to set the `spin` property of the `otherObject`'s `TwirlCapsule` component to true. The line 23 is circled in red, and four numbered circles (1, 2, 3, 4) with arrows point to the circled line: circle 1 points to the component name, circle 2 to the property name, circle 3 to the assignment operator, and circle 4 to the value being assigned.

Bu kodda biraz uzun bir Dot Syntax kullanımı vardır.

4. Capsule GameObject’indeki (Değişkeni 5. satırda Capsule’yi atamıştık.),

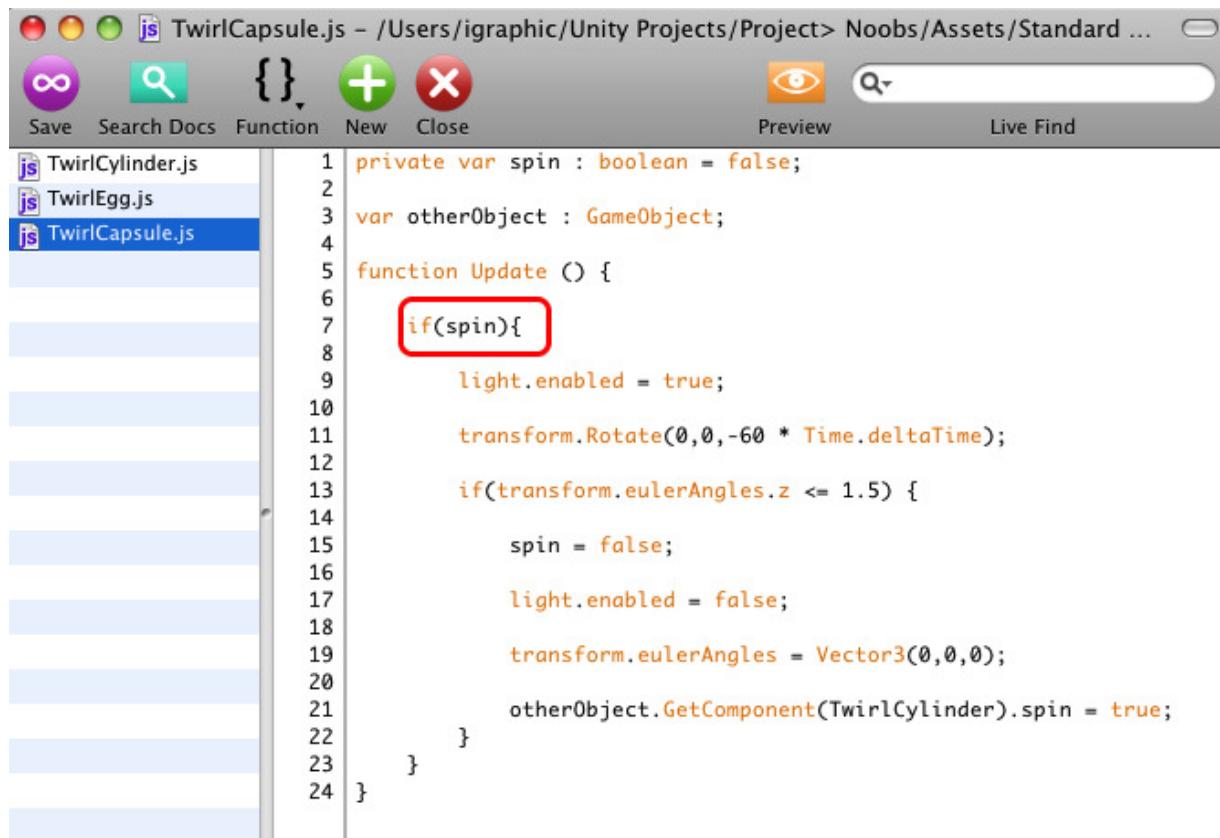
3. TwirlCapsule scriptinde yer alan

2. spin değişkeninin değerini

1. true olarak değiştiriyorum. (**assign**)

Burada **Dot Syntax**’ı (**Noktasal Yapı**) başka bir **GameObject**’in içindeki bir scriptteki bir değişkeni bulup değerini değiştirmek için kullanıyoruz.

TwirlCapsule scripti ve spin değişkeni



The screenshot shows the Unity Editor's Text Editor window. The title bar says "TwirlCapsule.js – /Users/igraphic/Unity Projects/Project> Noobs/Assets/Standard ...". The menu bar includes Save, Search Docs, Function, New, Close, Preview, and Live Find. The left sidebar lists other scripts: TwirlCylinder.js, TwirlEgg.js, and TwirlCapsule.js (which is selected). The main code area contains the following JavaScript code:

```
1 private var spin : boolean = false;
2
3 var otherObject : GameObject;
4
5 function Update () {
6
7     if(spin){
8
9         light.enabled = true;
10
11        transform.Rotate(0,0,-60 * Time.deltaTime);
12
13        if(transform.eulerAngles.z <= 1.5) {
14
15            spin = false;
16
17            light.enabled = false;
18
19            transform.eulerAngles = Vector3(0,0,0);
20
21            otherObject.GetComponent(TwirlCylinder).spin = true;
22        }
23    }
24 }
```

The line "if(spin){" is highlighted with a red rectangle.

Burada **TwirlCapsule** scriptini görüyorsunuz. Burada gördüğünüz **spin** değişkeninin değerini az önce **true** olarak ayarladık. Bu script **TwirlEgg** scriptine oldukça benzemektedir ve aynı işi yapmaktadır.

TwirlEgg scriptine geri dönelim

```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
10         light.enabled = true;
11
12         transform.Rotate(0,0,60 * Time.deltaTime);
13
14         if(transform.eulerAngles.z >= 358.5) {
15
16             spin = false;
17
18             light.enabled = false;
19
20             transform.eulerAngles = Vector3(0,0,0);
21
22             otherObject.GetComponent(TwirlCapsule).spin = true;
23
24         }
25     }
26 }
```

1. Dot Syntax'ta yapmış olduğumuz ilk iş Capsule GameObject'ini hedeflemekti (**address**). Bunu depolayan değişken ise otherObject'tır.

2. Bu değişkene burada atamıştık Capsule GameObject'ini.

Bunun anlamı ne zaman otherObject değişkenini kullandığımızda Capsule GameObject'yle iş yapmış oluyoruz.

3. Dot Syntax'ın sonraki aşamasında ise Capsule objesindeki TwirlCapsule component'ine (parça) geçiş yapıyoruz (**address**).

GameObject.GetComponent

(2)

```
function GetComponent (type : Type) : Component
```

Description

Returns the component of Type type if the game object has one attached, null if it doesn't. You can access both builtin components or scripts with this function.

GetComponent is the primary way of accessing other components. From javascript the type of a script is always the name of the script as seen in the project view.
Example:

JavaScript ▾

```
function Start () {
    var curTransform : Transform;

    curTransform = gameObject.GetComponent(Transform);
    // This is equivalent to:
    curTransform = gameObject.transform;
}

function Update () {
    // To access public variables and functions
    // in another script attached to the same game object.
    // (ScriptName is the name of the javascript file)
    var other : ScriptName = gameObject.GetComponent(ScriptName); 1
    // Call the function DoSomething on the script
    other.DoSomething ();
    // set another variable in the other script instance
    other.someVariable = 5;
}
```

[\(Buraya tıklayarak\)](#) GetComponent hakkında bilgi edinebilirsiniz.

1. Burada 23. satırda yaptığımıza benzer bir örnek var. Bu örnekte **other** isimli bir değişkene, bir objedeki bir script depolanmaktadır.

Biz ise 23. satırda bu scripti herhangi bir değişkene atamıyoruz.

2. **GetComponent(TwirlCapsule)** bir **Component** döndürmeye (**return**) yarar, bizim örneğimizde döndürdüğümüz **component** **TwirlCapsule** scripti. Bizim yaptığımız şey bu **component**'i bir değişkene atamak yerine direk **Dot Syntax**'ın bir parçası olarak kullanmak. Bu componenti önce bir değişkene depolayarak fazladan kod yazmayı gerektirmenin bir anlamı yok.

Inspector Panelini Kullanarak Atama Yapmak (Assign)

Ustaca düşünülmüş bir Unity özelliği (feature)

TwirlEgg scriptinin 5. satırında **Capsule GameObject**'ini **GameObject.Find()** fonksiyonuyla buluyorduk. Ancak diğer iki script (**TwirlCapsule** ve **TwirlCylinder**) **GameObject.Find()** fonksiyonunu kullanmamaktadırlar. Diğer objeyi bulmak için başka bir kod da kullanmıyorlar, ancak buna rağmen uygulamam mükemmel bir şekilde çalışıyor. Çünkü o scriptlerdeki **atama (assign)** işlemini script kullanarak değil de Unity'nin **Inspector** panelini kullanarak yaptım.

Bu öteki 2 scripti oluşturmamın sebebi **Inspector** vasıtasiyla sürükle-bırak (drag-drop) yoluyla atama yapmayı göstermekti.

Not:

Normalde bu 3 **GameObject** için de tek bir script yazmam yeterliydi, değerleri de **Inspector**'dan atama yaparak halledebilirdim. Ancak bu bir eğitim kılavuzu olduğu için 2 yolu da göstermek istedim ve onun için bu uzun yolu seçtim.

Scriptlerdeki farklılıklar

```
1 private var spin : boolean = true; ← 3
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule"); ← 5
6
7 function Update () {
8
9     if(spin){
10
11         if(spin){ ← 4
12
13             var otherObject : GameObject;
14
15             function Update () {
16
17                 if(spin){ ← 2
18
19                     if(spin){ ← 1
20
21                         if(spin){ ← 2
22
23                             if(spin){ ← 1
24
25                             if(spin){ ← 2
26
27                             if(spin){ ← 1
28
29                             if(spin){ ← 2
30
31                             if(spin){ ← 1
32
33                             if(spin){ ← 2
34
35                             if(spin){ ← 1
36
37                             if(spin){ ← 2
38
39                             if(spin){ ← 1
40
41                             if(spin){ ← 2
42
43                             if(spin){ ← 1
44
45                             if(spin){ ← 2
46
47                             if(spin){ ← 1
48
49                             if(spin){ ← 2
50
51                             if(spin){ ← 1
52
53                             if(spin){ ← 2
54
55                             if(spin){ ← 1
56
57                             if(spin){ ← 2
58
59                             if(spin){ ← 1
60
61                             if(spin){ ← 2
62
63                             if(spin){ ← 1
64
65                             if(spin){ ← 2
66
67                             if(spin){ ← 1
68
69                             if(spin){ ← 2
70
71                             if(spin){ ← 1
72
73                             if(spin){ ← 2
74
75                             if(spin){ ← 1
76
77                             if(spin){ ← 2
78
79                             if(spin){ ← 1
80
81                             if(spin){ ← 2
82
83                             if(spin){ ← 1
84
85                             if(spin){ ← 2
86
87                             if(spin){ ← 1
88
89                             if(spin){ ← 2
90
91                             if(spin){ ← 1
92
93                             if(spin){ ← 2
94
95                             if(spin){ ← 1
96
97                             if(spin){ ← 2
98
99                             if(spin){ ← 1
100
101                             if(spin){ ← 2
102
103                             if(spin){ ← 1
104
105                             if(spin){ ← 2
106
107                             if(spin){ ← 1
108
109                             if(spin){ ← 2
110
111                             if(spin){ ← 1
112
113                             if(spin){ ← 2
114
115                             if(spin){ ← 1
116
117                             if(spin){ ← 2
118
119                             if(spin){ ← 1
120
121                             if(spin){ ← 2
122
123                             if(spin){ ← 1
124
125                             if(spin){ ← 2
126
127                             if(spin){ ← 1
128
129                             if(spin){ ← 2
130
131                             if(spin){ ← 1
132
133                             if(spin){ ← 2
134
135                             if(spin){ ← 1
136
137                             if(spin){ ← 2
138
139                             if(spin){ ← 1
140
141                             if(spin){ ← 2
142
143                             if(spin){ ← 1
144
145                             if(spin){ ← 2
146
147                             if(spin){ ← 1
148
149                             if(spin){ ← 2
150
151                             if(spin){ ← 1
152
153                             if(spin){ ← 2
154
155                             if(spin){ ← 1
156
157                             if(spin){ ← 2
158
159                             if(spin){ ← 1
160
161                             if(spin){ ← 2
162
163                             if(spin){ ← 1
164
165                             if(spin){ ← 2
166
167                             if(spin){ ← 1
168
169                             if(spin){ ← 2
170
171                             if(spin){ ← 1
172
173                             if(spin){ ← 2
174
175                             if(spin){ ← 1
176
177                             if(spin){ ← 2
178
179                             if(spin){ ← 1
180
181                             if(spin){ ← 2
182
183                             if(spin){ ← 1
184
185                             if(spin){ ← 2
186
187                             if(spin){ ← 1
188
189                             if(spin){ ← 2
190
191                             if(spin){ ← 1
192
193                             if(spin){ ← 2
194
195                             if(spin){ ← 1
196
197                             if(spin){ ← 2
198
199                             if(spin){ ← 1
200
201                             if(spin){ ← 2
202
203                             if(spin){ ← 1
204
205                             if(spin){ ← 2
206
207                             if(spin){ ← 1
208
209                             if(spin){ ← 2
210
211                             if(spin){ ← 1
212
213                             if(spin){ ← 2
214
215                             if(spin){ ← 1
216
217                             if(spin){ ← 2
218
219                             if(spin){ ← 1
220
221                             if(spin){ ← 2
222
223                             if(spin){ ← 1
224
225                             if(spin){ ← 2
226
227                             if(spin){ ← 1
228
229                             if(spin){ ← 2
230
231                             if(spin){ ← 1
232
233                             if(spin){ ← 2
234
235                             if(spin){ ← 1
236
237                             if(spin){ ← 2
238
239                             if(spin){ ← 1
240
241                             if(spin){ ← 2
242
243                             if(spin){ ← 1
244
245                             if(spin){ ← 2
246
247                             if(spin){ ← 1
248
249                             if(spin){ ← 2
250
251                             if(spin){ ← 1
252
253                             if(spin){ ← 2
254
255                             if(spin){ ← 1
256
257                             if(spin){ ← 2
258
259                             if(spin){ ← 1
260
261                             if(spin){ ← 2
262
263                             if(spin){ ← 1
264
265                             if(spin){ ← 2
266
267                             if(spin){ ← 1
268
269                             if(spin){ ← 2
270
271                             if(spin){ ← 1
272
273                             if(spin){ ← 2
274
275                             if(spin){ ← 1
276
277                             if(spin){ ← 2
278
279                             if(spin){ ← 1
280
281                             if(spin){ ← 2
282
283                             if(spin){ ← 1
284
285                             if(spin){ ← 2
286
287                             if(spin){ ← 1
288
289                             if(spin){ ← 2
290
291                             if(spin){ ← 1
292
293                             if(spin){ ← 2
294
295                             if(spin){ ← 1
296
297                             if(spin){ ← 2
298
299                             if(spin){ ← 1
300
301                             if(spin){ ← 2
302
303                             if(spin){ ← 1
304
305                             if(spin){ ← 2
306
307                             if(spin){ ← 1
308
309                             if(spin){ ← 2
310
311                             if(spin){ ← 1
312
313                             if(spin){ ← 2
314
315                             if(spin){ ← 1
316
317                             if(spin){ ← 2
318
319                             if(spin){ ← 1
320
321                             if(spin){ ← 2
322
323                             if(spin){ ← 1
324
325                             if(spin){ ← 2
326
327                             if(spin){ ← 1
328
329                             if(spin){ ← 2
330
331                             if(spin){ ← 1
332
333                             if(spin){ ← 2
334
335                             if(spin){ ← 1
336
337                             if(spin){ ← 2
338
339                             if(spin){ ← 1
340
341                             if(spin){ ← 2
342
343                             if(spin){ ← 1
344
345                             if(spin){ ← 2
346
347                             if(spin){ ← 1
348
349                             if(spin){ ← 2
350
351                             if(spin){ ← 1
352
353                             if(spin){ ← 2
354
355                             if(spin){ ← 1
356
357                             if(spin){ ← 2
358
359                             if(spin){ ← 1
360
361                             if(spin){ ← 2
362
363                             if(spin){ ← 1
364
365                             if(spin){ ← 2
366
367                             if(spin){ ← 1
368
369                             if(spin){ ← 2
370
371                             if(spin){ ← 1
372
373                             if(spin){ ← 2
374
375                             if(spin){ ← 1
376
377                             if(spin){ ← 2
378
379                             if(spin){ ← 1
380
381                             if(spin){ ← 2
382
383                             if(spin){ ← 1
384
385                             if(spin){ ← 2
386
387                             if(spin){ ← 1
388
389                             if(spin){ ← 2
390
391                             if(spin){ ← 1
392
393                             if(spin){ ← 2
394
395                             if(spin){ ← 1
396
397                             if(spin){ ← 2
398
399                             if(spin){ ← 1
400
401                             if(spin){ ← 2
402
403                             if(spin){ ← 1
404
405                             if(spin){ ← 2
406
407                             if(spin){ ← 1
408
409                             if(spin){ ← 2
410
411                             if(spin){ ← 1
412
413                             if(spin){ ← 2
414
415                             if(spin){ ← 1
416
417                             if(spin){ ← 2
418
419                             if(spin){ ← 1
420
421                             if(spin){ ← 2
422
423                             if(spin){ ← 1
424
425                             if(spin){ ← 2
426
427                             if(spin){ ← 1
428
429                             if(spin){ ← 2
430
431                             if(spin){ ← 1
432
433                             if(spin){ ← 2
434
435                             if(spin){ ← 1
436
437                             if(spin){ ← 2
438
439                             if(spin){ ← 1
440
441                             if(spin){ ← 2
442
443                             if(spin){ ← 1
444
445                             if(spin){ ← 2
446
447                             if(spin){ ← 1
448
449                             if(spin){ ← 2
450
451                             if(spin){ ← 1
452
453                             if(spin){ ← 2
454
455                             if(spin){ ← 1
456
457                             if(spin){ ← 2
458
459                             if(spin){ ← 1
460
461                             if(spin){ ← 2
462
463                             if(spin){ ← 1
464
465                             if(spin){ ← 2
466
467                             if(spin){ ← 1
468
469                             if(spin){ ← 2
470
471                             if(spin){ ← 1
472
473                             if(spin){ ← 2
474
475                             if(spin){ ← 1
476
477                             if(spin){ ← 2
478
479                             if(spin){ ← 1
480
481                             if(spin){ ← 2
482
483                             if(spin){ ← 1
484
485                             if(spin){ ← 2
486
487                             if(spin){ ← 1
488
489                             if(spin){ ← 2
490
491                             if(spin){ ← 1
492
493                             if(spin){ ← 2
494
495                             if(spin){ ← 1
496
497                             if(spin){ ← 2
498
499                             if(spin){ ← 1
500
501                             if(spin){ ← 2
502
503                             if(spin){ ← 1
504
505                             if(spin){ ← 2
506
507                             if(spin){ ← 1
508
509                             if(spin){ ← 2
510
511                             if(spin){ ← 1
512
513                             if(spin){ ← 2
514
515                             if(spin){ ← 1
516
517                             if(spin){ ← 2
518
519                             if(spin){ ← 1
520
521                             if(spin){ ← 2
522
523                             if(spin){ ← 1
524
525                             if(spin){ ← 2
526
527                             if(spin){ ← 1
528
529                             if(spin){ ← 2
530
531                             if(spin){ ← 1
532
533                             if(spin){ ← 2
534
535                             if(spin){ ← 1
536
537                             if(spin){ ← 2
538
539                             if(spin){ ← 1
540
541                             if(spin){ ← 2
542
543                             if(spin){ ← 1
544
545                             if(spin){ ← 2
546
547                             if(spin){ ← 1
548
549                             if(spin){ ← 2
550
551                             if(spin){ ← 1
552
553                             if(spin){ ← 2
554
555                             if(spin){ ← 1
556
557                             if(spin){ ← 2
558
559                             if(spin){ ← 1
560
561                             if(spin){ ← 2
562
563                             if(spin){ ← 1
564
565                             if(spin){ ← 2
566
567                             if(spin){ ← 1
568
569                             if(spin){ ← 2
570
571                             if(spin){ ← 1
572
573                             if(spin){ ← 2
574
575                             if(spin){ ← 1
576
577                             if(spin){ ← 2
578
579                             if(spin){ ← 1
580
581                             if(spin){ ← 2
582
583                             if(spin){ ← 1
584
585                             if(spin){ ← 2
586
587                             if(spin){ ← 1
588
589                             if(spin){ ← 2
590
591                             if(spin){ ← 1
592
593                             if(spin){ ← 2
594
595                             if(spin){ ← 1
596
597                             if(spin){ ← 2
598
599                             if(spin){ ← 1
600
601                             if(spin){ ← 2
602
603                             if(spin){ ← 1
604
605                             if(spin){ ← 2
606
607                             if(spin){ ← 1
608
609                             if(spin){ ← 2
610
611                             if(spin){ ← 1
612
613                             if(spin){ ← 2
614
615                             if(spin){ ← 1
616
617                             if(spin){ ← 2
618
619                             if(spin){ ← 1
620
621                             if(spin){ ← 2
622
623                             if(spin){ ← 1
624
625                             if(spin){ ← 2
626
627                             if(spin){ ← 1
628
629                             if(spin){ ← 2
630
631                             if(spin){ ← 1
632
633                             if(spin){ ← 2
634
635                             if(spin){ ← 1
636
637                             if(spin){ ← 2
638
639                             if(spin){ ← 1
640
641                             if(spin){ ← 2
642
643                             if(spin){ ← 1
644
645                             if(spin){ ← 2
646
647                             if(spin){ ← 1
648
649                             if(spin){ ← 2
650
651                             if(spin){ ← 1
652
653                             if(spin){ ← 2
654
655                             if(spin){ ← 1
656
657                             if(spin){ ← 2
658
659                             if(spin){ ← 1
660
661                             if(spin){ ← 2
662
663                             if(spin){ ← 1
664
665                             if(spin){ ← 2
666
667                             if(spin){ ← 1
668
669                             if(spin){ ← 2
670
671                             if(spin){ ← 1
672
673                             if(spin){ ← 2
674
675                             if(spin){ ← 1
676
677                             if(spin){ ← 2
678
679                             if(spin){ ← 1
680
681                             if(spin){ ← 2
682
683                             if(spin){ ← 1
684
685                             if(spin){ ← 2
686
687                             if(spin){ ← 1
688
689                             if(spin){ ← 2
690
691                             if(spin){ ← 1
692
693                             if(spin){ ← 2
694
695                             if(spin){ ← 1
696
697                             if(spin){ ← 2
698
699                             if(spin){ ← 1
700
701                             if(spin){ ← 2
702
703                             if(spin){ ← 1
704
705                             if(spin){ ← 2
706
707                             if(spin){ ← 1
708
709                             if(spin){ ← 2
710
711                             if(spin){ ← 1
712
713                             if(spin){ ← 2
714
715                             if(spin){ ← 1
716
717                             if(spin){ ← 2
718
719                             if(spin){ ← 1
720
721                             if(spin){ ← 2
722
723                             if(spin){ ← 1
724
725                             if(spin){ ← 2
726
727                             if(spin){ ← 1
728
729                             if(spin){ ← 2
730
731                             if(spin){ ← 1
732
733                             if(spin){ ← 2
734
735                             if(spin){ ← 1
736
737                             if(spin){ ← 2
738
739                             if(spin){ ← 1
740
741                             if(spin){ ← 2
742
743                             if(spin){ ← 1
744
745                             if(spin){ ← 2
746
747                             if(spin){ ← 1
748
749                             if(spin){ ← 2
750
751                             if(spin){ ← 1
752
753                             if(spin){ ← 2
754
755                             if(spin){ ← 1
756
757                             if(spin){ ← 2
758
759                             if(spin){ ← 1
760
761                             if(spin){ ← 2
762
763                             if(spin){ ← 1
764
765                             if(spin){ ← 2
766
767                             if(spin){ ← 1
768
769                             if(spin){ ← 2
770
771                             if(spin){ ← 1
772
773                             if(spin){ ← 2
774
775                             if(spin){ ← 1
776
777                             if(spin){ ← 2
778
779                             if(spin){ ← 1
780
781                             if(spin){ ← 2
782
783                             if(spin){ ← 1
784
785                             if(spin){ ← 2
786
787                             if(spin){ ← 1
788
789                             if(spin){ ← 2
790
791                             if(spin){ ← 1
792
793                             if(spin){ ← 2
794
795                             if(spin){ ← 1
796
797                             if(spin){ ← 2
798
799                             if(spin){ ← 1
800
801                             if(spin){ ← 2
802
803                             if(spin){ ← 1
804
805                             if(spin){ ← 2
806
807                             if(spin){ ← 1
808
809                             if(spin){ ← 2
810
811                             if(spin){ ← 1
812
813                             if(spin){ ← 2
814
815                             if(spin){ ← 1
816
817                             if(spin){ ← 2
818
819                             if(spin){ ← 1
820
821                             if(spin){ ← 2
822
823                             if(spin){ ← 1
824
825                             if(spin){ ← 2
826
827                             if(spin){ ← 1
828
829                             if(spin){ ← 2
830
831                             if(spin){ ← 1
832
833                             if(spin){ ← 2
834
835                             if(spin){ ← 1
836
837                             if(spin){ ← 2
838
839                             if(spin){ ← 1
840
841                             if(spin){ ← 2
842
843                             if(spin){ ← 1
844
845                             if(spin){ ← 2
846
847                             if(spin){ ← 1
848
849                             if(spin){ ← 2
850
851                             if(spin){ ← 1
852
853                             if(spin){ ← 2
854
855                             if(spin){ ← 1
856
857                             if(spin){ ← 2
858
859                             if(spin){ ← 1
860
861                             if(spin){ ← 2
862
863                             if(spin){ ← 1
864
865                             if(spin){ ← 2
866
867                             if(spin){ ← 1
868
869                             if(spin){ ← 2
870
871                             if(spin){ ← 1
872
873                             if(spin){ ← 2
874
875                             if(spin){ ← 1
876
877                             if(spin){ ← 2
878
879                             if(spin){ ← 1
880
881                             if(spin){ ← 2
882
883                             if(spin){ ← 1
884
885                             if(spin){ ← 2
886
887                             if(spin){ ← 1
888
889                             if(spin){ ← 2
890
891                             if(spin){ ← 1
892
893                             if(spin){ ← 2
894
895                             if(spin){ ← 1
896
897                             if(spin){ ← 2
898
899                             if(spin){ ← 1
900
901                             if(spin){ ← 2
902
903                             if(spin){ ← 1
904
905                             if(spin){ ← 2
906
907                             if(spin){ ← 1
908
909                             if(spin){ ← 2
910
911                             if(spin){ ← 1
912
913                             if(spin){ ← 2
914
915                             if(spin){ ← 1
916
917                             if(spin){ ← 2
918
919                             if(spin){ ← 1
920
921                             if(spin){ ← 2
922
923                             if(spin){ ← 1
924
925                             if(spin){ ← 2
926
927                             if(spin){ ← 1
928
929                             if(spin){ ← 2
930
931                             if(spin){ ← 1
932
933                             if(spin){ ← 2
934
935                             if(spin){ ← 1
936
937                             if(spin){ ← 2
938
939                             if(spin){ ← 1
940
941                             if(spin){ ← 2
942
943                             if(spin){ ← 1
944
945                             if(spin){ ← 2
946
947                             if(spin){ ← 1
948
949                             if(spin){ ← 2
950
951                             if(spin){ ← 1
952
953                             if(spin){ ← 2
954
955                             if(spin){ ← 1
956
957                             if(spin){ ← 2
958
959                             if(spin){ ← 1
960
961                             if(spin){ ← 2
962
963                             if(spin){ ← 1
964
965                             if(spin){ ← 2
966
967                             if(spin){ ← 1
968
969                             if(spin){ ← 2
970
971                             if(spin){ ← 1
972
973                             if(spin){ ← 2
974
975                             if(spin){ ← 1
976
977                             if(spin){ ← 2
978
979                             if(spin){ ← 1
980
981                             if(spin){ ← 2
982
983                             if(spin){ ← 1
984
985                             if(spin){ ← 2
986
987                             if(spin){ ← 1
988
989                             if(spin){ ← 2
990
991                             if(spin){ ← 1
992
993                             if(spin){ ← 2
994
995                             if(spin){ ← 1
996
997                             if(spin){ ← 2
998
999                             if(spin){ ← 1
1000
1001                             if(spin){ ← 2
1002
1003                             if(spin){ ← 1
1004
1005                             if(spin){ ← 2
1006
1007                             if(spin){ ← 1
1008
1009                             if(spin){ ← 2
1010
1011                             if(spin){ ← 1
1012
1013                             if(spin){ ← 2
1014
1015                             if(spin){ ← 1
1016
1017                             if(spin){ ← 2
1018
1019                             if(spin){ ← 1
1020
1021                             if(spin){ ← 2
1022
1023                             if(spin){ ← 1
1024
1025                             if(spin){ ← 2
1026
1027                             if(spin){ ← 1
1028
1029                             if(spin){ ← 2
1030
1031                             if(spin){ ← 1
1032
1033                             if(spin){ ← 2
1034
1035                             if(spin){ ← 1
1036
1037                             if(spin){ ← 2
1038
1039                             if(spin){ ← 1
1040
1041                             if(spin){ ← 2
1042
1043                             if(spin){ ← 1
1044
1045                             if(spin){ ← 2
1046
1047                             if(spin){ ← 1
1048
1049                             if(spin){ ← 2
1050
1051                             if(spin){ ← 1
1052
1053                             if(spin){ ← 2
1054
1055                             if(spin){ ← 1
1056
1057                             if(spin){ ← 2
1058
1059                             if(spin){ ← 1
1060
1061                             if(spin){ ← 2
1062
1063                             if(spin){ ← 1
1064
1065                             if(spin){ ← 2
1066
1067                             if(spin){ ← 1
1068
1069                             if(spin){ ← 2
1070
1071                             if(spin){ ← 1
1072
1073                             if(spin){ ← 2
1074
1075                             if(spin){ ← 1
1076
1077                             if(spin){ ← 2
1078
1079                             if(spin){ ← 1
1080
1081                             if(spin){ ← 2
1082
1083                             if(spin){ ← 1
1084
1085                             if(spin){ ← 2
1086
1087                             if(spin){ ← 1
1088
1089                             if(spin){ ← 2
1090
1091                             if(spin){ ← 1
1092
1093                             if(spin){ ← 2
1094
1095                             if(spin){ ← 1
1096
1097                             if(spin){ ← 2
1098
1099                             if(spin){ ← 1
1100
1101                             if(spin){ ← 2
1102
1103                             if(spin){ ← 1
1104
1105                             if(spin){ ← 2
1106
1107                             if(spin){ ← 1
1108
1109                             if(spin){ ← 2
1110
1111                             if(spin){ ← 1
1112
1113                             if(spin){ ← 2
1114
1115                             if(spin){ ← 1
1116
1117                             if(spin){ ← 2
1118
1119                             if(spin){ ← 1
1120
1121                             if(spin){ ← 2
1122
1123                             if(spin){ ← 1
1124
1125                             if(spin){ ← 2
1126
1127                             if(spin){ ← 1
1128
1129                             if(spin){ ← 2
1130
1131                             if(spin){ ← 1
1132
1133                             if(spin){ ← 2
1134
1135                             if(spin){ ← 1
1136
1137                             if(spin){ ← 2
1138
1139                             if(spin){ ← 1
1140
1141                             if(spin){ ← 2
1142
1143                             if(spin){ ← 1
1144
1145                             if(spin){ ← 2
1146
1147                             if(spin){ ← 1
1148
1149                             if(spin){ ← 2
1150
1151                             if(spin){ ← 1
1152
1153                             if(spin){ ← 2
1154
1155                             if(spin){ ← 1
1156
1157                             if(spin){ ← 2
1158
1159                             if(spin){ ← 1
1160
1161                             if(spin){ ← 2
1162
1163                             if(spin){ ← 1
1164
1165                             if(spin){ ← 2
1166
1167                             if(spin){ ← 1
1168
1169                             if(spin){ ← 2
1170
1171                             if(spin){ ← 1
1172
1173                             if(spin){ ← 2
1174
1175                             if(spin){ ← 1
1176
1177                             if(spin){ ← 2
1178
1179                             if(spin){ ← 1
1180
1181                             if(spin){ ← 2
1182
1183                             if(spin){ ← 1
1184
1185                             if(spin){ ← 2
1186
1187                             if(spin){ ← 1
1188
1189                             if(spin){ ← 2
1190
1191                             if(spin){ ← 1
1192
1193                             if(spin){ ← 2
1194
1195                             if(spin){ ← 1
1196
1197                             if(spin){ ← 2
1198
1199                             if(spin){ ← 1
1200
1201                             if(spin){ ← 2
1202
1203                             if(spin){ ← 1
1204
1205                             if(spin){ ← 2
1206
1207                             if(spin){ ← 1
1208
1209                             if(spin){ ← 2
1210
1211                             if(spin){ ← 1
1212
1213                             if(spin){ ← 2
1214
1215                             if(spin){ ← 1
1216
1217                             if(spin){ ← 2
1218
1219                             if(spin){ ← 1
1220
1221                             if(spin){ ← 2
1222
1223                             if(spin){ ← 1
1224
1225                             if(spin){ ← 2
1226
1227                             if(spin){ ← 1
1228
1229                             if(spin){ ← 2
1230
1231                             if(spin){ ← 1
1232
1233                             if(spin){ ← 2
1234
1235                             if(spin){ ← 1
1236
1237                             if(spin){ ← 2
1238
1239                             if(spin){ ← 1
1240
1241                             if(spin){ ← 2
1242
1243                             if(spin){ ← 1
1244
1245                             if(spin){ ← 2
1246
1247                             if(spin){ ← 1
1248
1249                             if(spin){ ← 2
1250
1251                             if(spin){ ← 1
1252
1253                             if(spin){ ← 2
1254
1255                             if(spin){ ← 1
1256
1257                             if(spin){ ← 2
1258
1259                             if(spin){ ← 1
1260
1261                             if(spin){ ← 2
1262
1263                             if(spin){ ← 1
1264
1265                             if(spin){ ← 2
1266
1267                             if(spin){ ← 1
1268
1269                             if(spin){ ← 2
1270
1271                             if(spin){ ← 1
1272
1273                             if(spin){ ← 2
1274
1275                             if(spin){ ← 1
1276
1277                             if(spin){ ← 2
1278
1279                             if(spin){ ← 1
1280
1281                             if(spin){ ← 2
1282
1283                             if(spin){ ← 1
1284
1285                             if(spin){ ← 2
1286
1287                             if(spin){ ← 1
1288
1289                             if(spin){ ← 2
1290
1291                             if(spin){ ← 1
1
```

Şimdi sırada **spin** değişkeninin **tanımlanma (declare)** kısmı var:

3. Egg (yumurta) scriptinde bu değer **true**, çünkü oyun başladığında bir objenin ilk önce dönmeye başlaması lazım.
 4. Capsule (kapsül) scriptinde ise bu değer **false**.

Şimdi **otherObject** değişkenine sonraki **GameObject**'i atamaya gelirsek:

5. Egg scriptinde sonraki obje olarak **Capsule** objesi atanıyor, böylece yumurtadan sonra kapsül objesi dönmeye başlayacak.

TwirlCapsule scriptinde herhangi bir atama ifadesi olmadığına dikkat edin.

private (özel) değişken public (genel) değişkene karşı

```
1 private var spin : boolean = true;
2
3 private var otherObject : GameObject;
4
5 otherObject = GameObject.Find("Capsule");
6
7 function Update () {
8
9     if(spin){
```

```
1 private var spin : boolean = false;
2
3 var otherObject : GameObject;
4
5 function Update () {
6
7     if(spin){
```

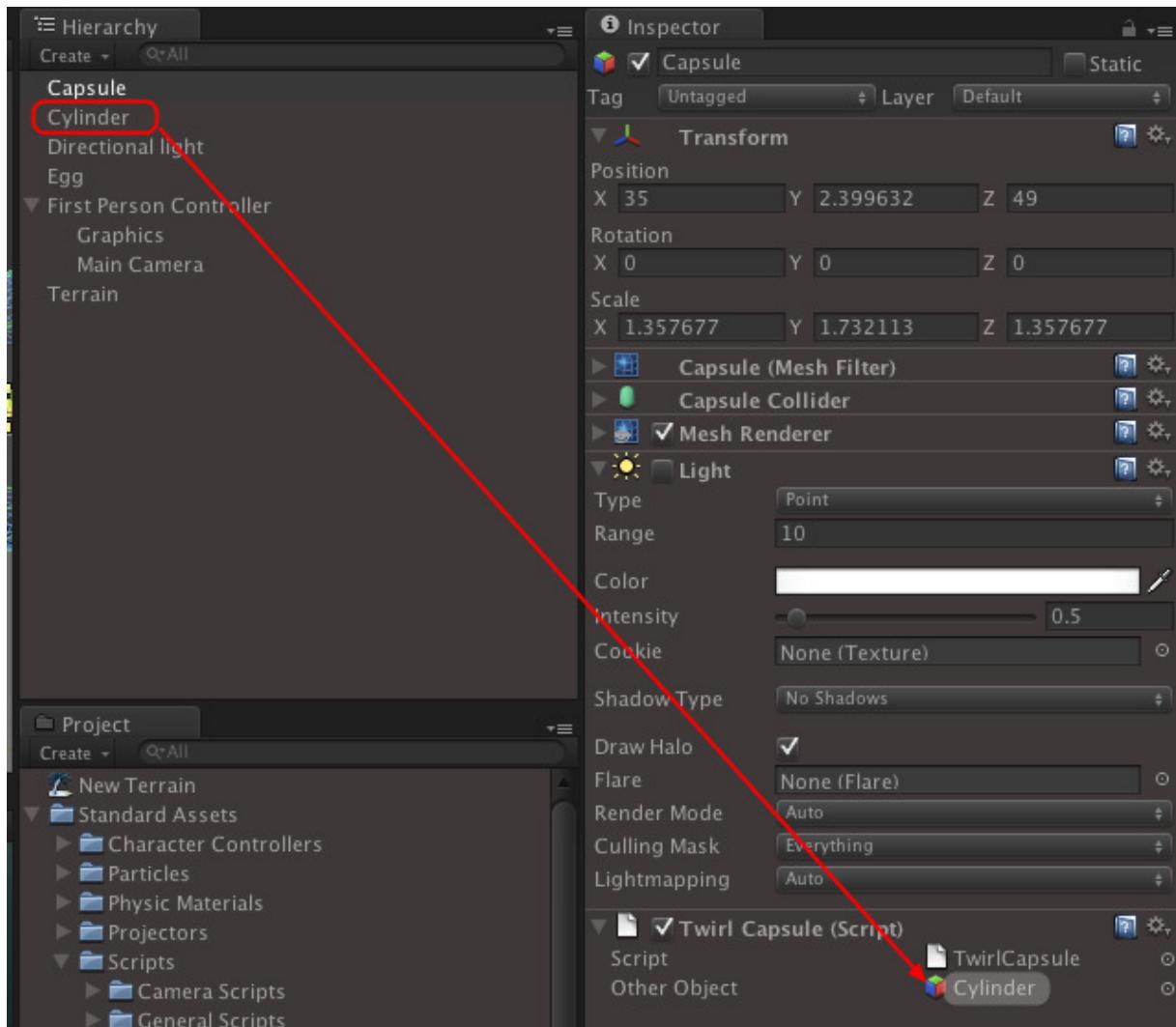
Niçin TwirlCapsule scriptinde otherObject değişkenine herhangi bir GameObject ataması yapılmıyor?

Çünkü bu atama işlemini Inspector'dan elle yapacağız. Bu özellik sayesinde daha esnek çalışabilir ve daha hızlı olabiliriz. Ayrıca bir değişiklik yapacağımız çoğu zaman da bu orijinal kodu değiştirmemize de gerek kalmaz.

Bu işlemi **Inspector**'dan yapabilmemiz için **değişkenin (variable)** de **Inspector** panelinde gözükmesi lazım.

1. TwirlEgg scriptinde **otherObject** değişkeni **private**, bunun anlamı bu değişken **Inspector** panelinde gözükmez.
 2. Burada ise değişken **private** değil, yani **public**. Bu yüzden bu **otherObject** değişkeni **Inspector**'da gözükmür.

Atama (assign) yapmak için sürüklemek yeterli



Capsule objesi dönmesini tamamladıktan sonra **Cylinder** (silindir) objesi dönmeye başlayacaktır.

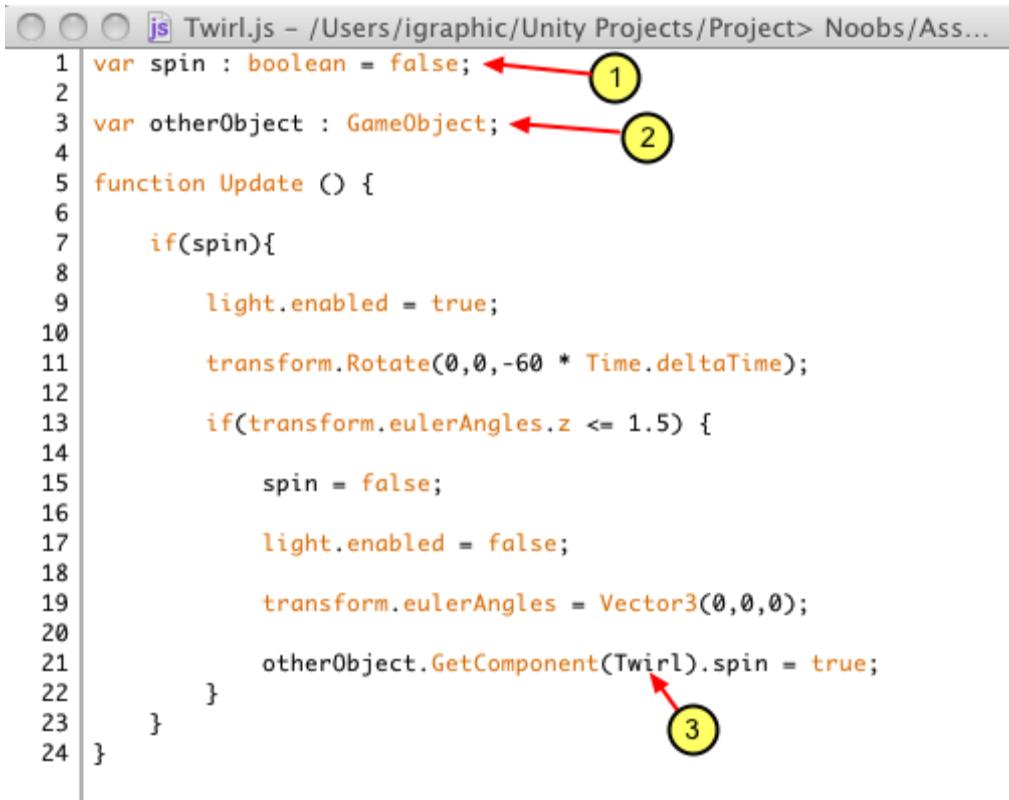
Cylinder objesi **otherObject** değişkenine çok basit bir şekilde atandı. Tek yaptığımız **Cylinder** objesini **Hierarchy** (Hiyerarşi) panelinden tutup sürükleyerek **Inspector** panelindeki **otherObject** değişkeninin değerinin olduğu yere bırakmaktı. Unity'nin **otherObject** değişkeninin ismini 2 kelimeye ayırdığını dikkat edin: "Other Object".

Tamamdır! **Cylinder GameObject**'ının scriptte herhangi bir şekilde yer almadığını dert etmenize bile gerek yok. Ayrıca atama (**assign**) işlemi script üzerinden yapılmış olsaydı bile başka bir objeyi **Inspector** panelinde değişkenin üzerine sürüklemek; scriptteki objeyi geçersiz kılıp (**override**) sürüklendiğimiz yeni objenin esas alınmasını sağlar.

Inspector'u ve tek bir scripti kullanarak aynı işlemi yapmak

Bu bölümde 3 scripti de sildim ve yerlerine tek bir script oluşturarak 3 objeye de o scripti atadım: *Twirl.js*

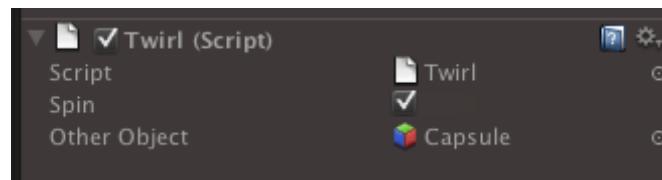
Ayrıca **spin** değişkenini de **public** (**genel**) yaptım, böylece değeri **Inspector**'da gözükebilecek. Ayrıca böylece ilk hangi objenin döneceğine de karar verebiliriz.



```
1 var spin : boolean = false; ← 1
2
3 var otherObject : GameObject; ← 2
4
5 function Update () {
6
7     if(spin){
8
9         light.enabled = true;
10
11        transform.Rotate(0,0,-60 * Time.deltaTime);
12
13        if(transform.eulerAngles.z <= 1.5) {
14
15            spin = false;
16
17            light.enabled = false;
18
19            transform.eulerAngles = Vector3(0,0,0);
20
21            otherObject.GetComponent(Twirl).spin = true; ← 3
22        }
23    }
24 }
```

1. Artık **spin** değişkeni **public** ve böylece **Inspector** panelinde de gözükecek. Ayrıca değeri de şimdi **false**.
2. **otherObject** değişkeni de artık **public**, böylece bu değişken de **Inspector**'da gözükecek.
3. Artık 3 obje için de farklı bir script kullanmak yerine hepsi için **Twirl** scriptini kullandım.

Inspector özellikleri



Bu, Egg (yumurta) **GameObject**'ının **Twirl component**'inden (parça) bir görüntü. **Spin** değişkeninin işaretli olduğuna dikkat edin. Bu ona scriptte ayarlanmış olan **false** değerinin üzerine geçer (**override**) ve onu etkisiz kılar. Bunun anlamı oyun başladığında yine ilk olarak yumurta objesi dönecektir.

Yumurtanın dönmesi tamamlanınca da **otherObject**'e atadığımız **Capsule GameObject**'ı dönecektir.

**Tebrikler, artık scriptlemenin
en önemli temel bilgilerini
biliyorsunuz**

Sırada ne var?

Artık değişkenler (variable), fonksiyonlar (function) ve temel iletişim (communication) prensipleri ile ilgili önemli temellere sahipsiniz, böylece çoğu scriptleme korkunuzun gittiğini düşünüyorum

Daha çok öğrenme

Gördüğünüz her türlü scripte artık çok zorlanmadan bakabilirsiniz. Bazı scriptler gerçekten uzun olabilir ancak onlar da temellerinde sadece **değişkenlerden, fonksiyonlardan** ve **Dot Syntax**'tan oluşuyor. Artık en uzun scripti bile bu basit bilgilerden yola çıkarak yorumlamaya başlayabilirsiniz.

Bu temel kılavuz çoğu diğer özelliklere degenmedi, döngüler (**loop**) ya da **array**'lar gibi (Türkçe karşılığını bilmiyorum.). Ancak zaten bu kılavuzun amacı da o değildi. Buna rağmen gördüğünüz her script eğitim dosyasına (tutorial) bakmanızı tavsiye ederim, artık başka tutoriallere bakarken de pek zorlanmazsınız zaten.

Ben de daha fazlasını yapmayı planlıyorum

Ben de bu yolda daha fazla tutorial oluşturarak scriptlemenin daha derinlerine inmeyi düşünüyorum. Derslerin çoğu video olarak verilecektir, ancak kimisi de bunun gibi bir kılavuz olarak. Ancak dikkat edin, yeni eğitim dosyalarımın bir kısmı ücretsiz olurken bir kısmı da paralı olacaktır.

(Çevirmen Eklemesi) : Artık biz de beleş olan eğitici kılavuzlarını çeviririz. Bir eğitim dosyasının daha sonuna geldik. Daha fazla ders için her zaman sitemi ziyaret edebilirsiniz:

<http://yasirkula.wordpress.com/>

Bu tip eğitici projelerin devamı için çalıntı yapmamaya dikkat edin. Eğer çalıntı yapılrsa da o; yapanın vicdanına kalmış. Başka derslerde görüşmek üzere!