Writing a Script

```
PlatformerPlayer.cs
Miscellaneous Files                    PlatformerPlayer
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class PlatformerPlayer : MonoBehaviour {
6
7       // Use this for initialization
8       void Start () {
9
10      }
11
12      // Update is called once per frame
13      void Update () {
14
15      }
16  }
17
18
```

Using what we have learned we can start making Unity Scripts. A script is a standalone piece of code you can write and then attach onto objects. A script is intended to only change the object it is attached to. A script is a blueprint, when you attach it to an object it becomes a component. What's the difference? A component is real code, while a script is only a text file that has to be made real (i.e. attached) to do something.

Code

A script has 2 big methods where you will write your code: `Start()` and `Update()`.

`Start()`
Any code in the body of Start is called when the script is first created

`Update()`

Any code in the body of Update is called every update cycle of the game

Game Cycle

Every game is an infinite loop that runs like so

1) Check for inputs

2) Update all objects/scripts

3) Draw objects on screen

4) Go to step 1

This loop generally can happen 100-400 times a second. That means your scripts update() is called 100-400 times a second! Interesting Tidbit: The more intense code you write the more burden you put on the game and you may have less updates per second. When you get less than 30 updates a second you start noticing screen freezing or stuttering in a game. Because it is not updating fast enough!

When you write your own script you will want to do a few things

1) Write your own script specific Variables

2) Write your own script specific Methods

3) Get access to other scripts on the same object your script is on

4) Get access to scripts on other objects

5) Write code that does what you want to do using data from other scripts

API (Application Programming Interface)

So every script has its own set of data (variables) and actions (Methods).

If I tell you about my script and tell you what data you can get from it and what actions you can do with it, what I am telling you is my script's API.

For example if you wanted to use our Save Manager, you can look up its API and then use its Save Method from your script whenever you wanted to.

API's are very powerful building blocks to making your own scripts. If you ever want you could even create your own API for others to use!

Creating your own Script

To create your script in Unity you can choose 'create new script' from the Project interface. Once created you can attach the script to an object.

Creating your own Variables and Methods in your scripts

Variables

Put these variables anywhere you want within the curly braces of your script.

Methods

You can make your own Methods you want by writing its signature and body togetherer anywhere within the curly braces of your script

Script security access

One cool new thing with scripts is you can either allow or deny other scripts to use your variables or methods. If you want it to be publicly accessible put the `public` keyword before your variable or Method. API's are made up of public variables and methods.

```
public Vector3 startPosition;
```

```
public void Explode(){...}
```