Opgave 1

Kommenter (//) alt koden i dit projekt med egne ord, som så:

```
// Called 1st frame
void Start ()
{
    // Gets the rigidbody from this gameObject
    rb = GetComponent<Rigidbody>();
}
```

Opgave 2

Sæt dette script på et tomt gameobject og indsæt selv paranteser(){}<> og citationstegn(”):

```
using UnityEngine;

public class NoParentheses : MonoBehaviour

    bool textIsOn = false;
    float timer = 1;
    string test =  YOU DID IT ;

    TextMesh textMesh;

       void Start

        textMesh = gameObject.AddComponent TextMesh ;
        textMesh.text = test;

        transform.position = Vector3.zero;

    void Update

        timer -= Time.deltaTime;
        if timer <= 0

            textIsOn = !textIsOn;

            if (textIsOn)

                textMesh.text = test;

            else

                textMesh.text = ;


            textMesh.color = new Color Random.Range 0, 2 , Random.Range 0, 2 ,
Random.Range 0, 2 ;
            timer = 1;
```

Kommenter derefter koden.

NNY 2018

Parentheses, how do they work?!

Scope {}

Curly brackets "{}" bliver brugt til at definere scopet på class, statements og funktioner. Man kan tænke på det, som at de outliner en box, hvori der kan ligge noget kode. Se billede.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CheatSheet : MonoBehaviour
{
    void Start()
    {
        if (true)
        {

        }
    }

    void EnFunktion()
    {

    }
}
```

Det kode man skriver indenfor et scope, er lokalt til det scope. Hvis man f.eks. i koden til højre skrev `int test = 0;` indenfor `Start()`-funktionen, så ville `test` ikke kunne ses i `EnFunktion()`, men den ville kunne ses i if-statementet, da det ligger indenfor `Start()`-funktionen's scope.

Som tommelfingerregel, så skal enhver linje kode afsluttes med `;`, medmindre linjen har et scope (altså ingen `;` efter `if(true)` eller `Start()`).

Type parameter <>

Pile-parenteserne bruges til funktioner, hvis de skal bruge en type. Det er sjældent man møder det i starten, så man kan bare huske at de bruges til `GetComponent<Type>()`, og ellers glemme det for nu.

Funktion/statement parenteser ()

Alle funktioner skal have "()" efter deres navn, se billedet ovenfor. På samme måde skal man skrive de 2 parenteser hvis man kalder funktionen; `EnFunktion();`.

Nogle funktioner skal bruge en værdi, som f.eks: `AddForce()`, der skal bruge en Vector3-værdi. Denne værdi skrives indenfor parenteserne; `AddForce(Vector3.up)`. Dette kan sagtens blive kludret hvis man f.eks. laver en ny Vector3; `new Vector3(0,0,0)`. Her skal man bare huske at alle start-parentes '(', skal have en tilsvarende slut-parentes ')': `AddForce(new Vector3(0,0,0))`.

Statements har også altid parenteser og reglerne er de samme; `if(true)`.

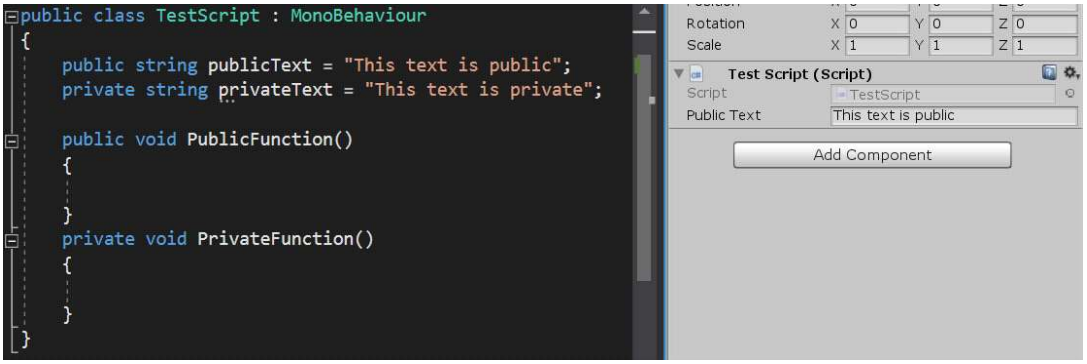
NNY 2018

Access Modifiers

public & private

Foran enhver variabel eller funktion kan man skrive en access-modifier, som fortæller hvor synlig den er. Der findes mange access modifiers, men de vigtigste er `public` og `private`.

I Unity bliver `public` variabler vidst i Inspectoren, mens `private` variabler er gemte. Funktioner vises aldrig i inspectoren.



Access-modifiers bestemmer også hvor synlige variabler/funktioner er for andre scripts, så man kan godt få fat i `public` variabler og funktioner fra et andet script, men ikke `private`.