

Introduction To Methods

What Are Methods?

- Methods (also called Functions) execute blocks of code that makes our game do things.
- To achieve this we must:
 1. DECLARE and define our method
 2. CALL our method when we want it to execute

```
[SerializeField] float runSpeed = 10f;  
[SerializeField] float jumpSpeed = 8f;  
[SerializeField] float climbSpeed = 5f;
```

```
const string CLIMB_POOL = "ClimbPool";  
const string JUMP_TRIGGER = "JumpTrigger";  
const string LADDER_TAG = "LadderTag";
```

```
bool atLadder;  
Vector3 screenPos = new Vector3(0, 0, 0);  
Renderer spriteRenderer;  
Animator animator;
```

```
void Update()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```

Real World Example

DECLARE Method

CleanYourRoom

- Pick up clothes
- Throw out garbage
- Kill the mutated science project that threatens to destroy life as we know it

CALL Method



Hey, go and
CleanYourRoom

“Go and do the steps we’ve defined and agreed to”

Syntax Used For Declaring Method

```
void CleanYourRoom()
```

```
{
```

```
}
```

Things To Do;

Return value

void = return nothing

Function name

WHAT to do

Parameter

() = nothing required

More Flavour Can Be Added

Two more common things when calling methods:

1. We can ask for some information to be **RETURNED**
2. We can specify some **PARAMETERS** are needed when calling the method

Using our real world example again...

```
[SerializeField] float runSpeed = 10f;  
[SerializeField] float jumpSpeed = 8f;  
[SerializeField] float climbSpeed = 5f;
```

```
public string CLIMB_POOL = "ClimbPool";  
public string JUMP_TRIGGER = "JumpTrigger";
```

```
Vector3 screenPos = new Vector3()  
SpriteRenderer spriteRenderer;  
Animator animator;
```

```
void Update ()
```

```
MoveHorizontally();  
ClimbLadders();  
ProcessJump();  
SaveTheWorld();
```


More Flavour Can Be Added

DECLARE Method

CleanYourRoom(*deadline*)

- Do it before (*deadline*)
- Pick up clothes
- Throw out garbage
- Kill the mutated science project that threatens to destroy life as we know it
- Is the room dirty?

CALL Method



Hey, go and
CleanYourRoom
(*you've got 1 hour*).
isRoomDirty?

*Parameter / Argument

*Return data

Syntax Used For Declaring Method

```
void CleanYourRoom()
```

```
{
```

```
Things To Do;
```

```
}
```

Return value

void = return nothing

Function name

WHAT to do

Parameter

() = nothing required

Syntax Used For Declaring Method

```
bool CleanYourRoom(int time)
{
    Things To Do;
    Deadline = time;
    return isRoomDirty;
}
```

Return value
(Type = bool)

Return keyword
(Type = bool)

Parameter
Needs type
and name

Syntax Used For Calling Method

CleanYourRoom();

Remember
semi colon

Executes statements defined in the code block...

```
{  
  ...  
}
```

```
[SerializeField] float runSpeed = 10.0f;  
[SerializeField] float jumpSpeed = 8.0f;  
[SerializeField] float climbSpeed = 10.0f;  
  
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string CLIMB_TAG = "Climb";  
  
bool atLadder;  
Vector3 screenPos = new Vector3();  
SpriteRenderer spriteRenderer;  
Animator animator;  
  
void Update()  
{  
  // Update logic  
  MoveHorizontally();  
  ClimbLadders();  
  ProcessJump();  
  SaveTheWorld();  
}
```

Defining & Calling

DECLARE Method

```
void CleanYourRoom()  
{  
    Pick up clothes;  
    Clean garbage;  
    Kill mutated experiment;  
}
```

CALL Method

```
CleanYourRoom();
```

```
[SerializeField] float runSpeed = 10f;  
[SerializeField] float jumpSpeed = 8f;  
[SerializeField] float climbSpeed = 10f;
```

```
const string CLIMB_BOOL = "Climb";  
const string JUMP_TRIGGER = "Jump";  
const string LADDER_TAG = "Ladder";
```

```
Vector3 screenPos = new Vector3()  
SpriteRenderer spriteRenderer;  
Animator animator;  
  
void Update()  
{  
    MoveHorizontally();  
    ClimbLadders();  
    ProcessJump();  
    SaveTheWorld();  
}
```


But What About Start() & Update()?!

- We've been defining **Start()** & **Update()** but not calling them
- Where are they called?
- Unity's internal logic is taking care of calling them for us at the right time
- These are referred to as "callbacks"

```
[SerializeField] float runSpeed = 10f;
[SerializeField] float jumpSpeed = 8f;
[SerializeField] float climbSpeed = 10f;
```

```
bool string CLIMB_POOL = false;
bool string JUMP_TRIGGER = false;
bool string LADDER_TAG = false;
```

```
bool atLadder;
Vector3 screenPos = new Vector3(0, 0, 0);
Animator animator;
```

```
void Update()
{
    MoveHorizontally();
    ClimbLadders();
    ProcessJump();
    SaveTheWorld();
}
```