# Spawning Entities

## Basics

### Overview

Spawning entities consists of 3 steps:

- Making the variable storing the future entity
- Modifying the attributes of the entity
- Spawning the entity

### Making a variable to store the entity

#### Example

> **level** is just a placeholder, in your code it needs to be defined, for many events you can use `event.level` in place of `level` and it will work

> You can create a entity from a **block** instead of **level**, and this is often preferred to learn that, scroll to that section afterward

```
let myEntity = level.createEntity("cow")
```

#### Breaking down the example

- **let**
  - Indicate that we are making a new variable and get the game ready to store it.
  - Not required in 1.16.
- **myEntity**
  - This is the name of the variable.
  - Can be anything you chose that is a-Z,0-9 without spaces (you know like any other variable).
- **=**
  - sets **myEntity** to what is about to follow.
- **level**
  - This is any level object that you choose.

    - This can be obtained numerous ways and will depend on what you are trying to do.
    - In many events you can use `event.level` to get the level.
    - Note: this is a LevelJS object, not a minecraftLevel object.
        - `minecraftLevel.asKJS()` returns a LevelJS.
- **.**
    - The dot operator either
        - Gets a property of the object.
        - Calls a method of the object.
        - Calls a beaned method of the object.
    - In this case it is used to call the method `createEntity`. You can tell because the following parenthesis mean its a method.
- **createEntity(...)**
    - As mentioned above is the method called by the dot operator
- **"cow"**
    - this is the name of the entity
    - "minecraft:cow" or "create:potato_projectile" are also valid
        - in fact when you put a *resource location* without a prefix, then `minecraft:` will be assumed.

# Modifying the properties

## Example

```
myEntity.x = 0
myEntity.y = 69
myEntity.z = 0
myEntity.motionY = 0.1
myEntity.noGravity = true
```

## Breaking Down the Example

- **myEntity**
    - Gets the variable that was made earlier.
- **.**
    - The dot operator mentioned earlier.
- **motionY = 0.1**
    - Instead of being a method, like last time, this is a beaned method.
    - This means that there exists a method `setMotion` and under the hood it runs `setMotionY(0.1)` that is automatically called with this code.
    - In this case it sets the `motionY` property of the entity.
    - > You many not change arbitrary bits of NBT this way! Only bits that there is a method for. In the example, all of the lines are just running beaned methods. However, you can do it with a different method, listed in a different section

> below.

## Spawning the entity

### Example

```
myEnity.spawn()
```

With understanding from the previous sections you should be able to figure out what this does.

It get **myEntity**, then calls the method **.spawn()**.

This `spawn()` method creates the entity in the world.

> Note: myEntity is still a variable! So you may not use `let myEntity` again within the scope! However this variable is still linked to the entity so calling `myEntity.motionY = 0.1` will still set the vertical motion of the entity. (This can be a useful thing, but bad if you are unaware)

# Creating the entity from a block

You can also call `createEntity` from a block! This is handy if you want to spawn the entity in the position of a block.

```
let myEntity = block.createEntity("cow")
```

> Again, **block** is just a place holder, you will need to change it to something else like maybe `event.block` for your code to work!

> This does **not** spawn the entity in the center of the block, it just sets the entity's coordinates to that of the block, thus being misaligned

This code offsets the entity to be in the center of the block.

```
let myEntity = block.createEntity("cow")
myEntity.x+=0.5
myEntity.y+=0.5
myEntity.z+=0.5
```

# Setting NBT

You **can** set the NBT to whatever you want! It's recommend using `mergeFullNBT` to do this.

```
myEntity.withNBT({VillagerData:{}})
```

> `myEntity.fullNBT.VillagerData = {}` will not work, because **.fullNBT** is a beaned method, not a property! The only thing that the beaned method lets do is to be able to use `let nbt = myEntity.fullNBT` to set a variable to NBT to be read or use `myEntity.fullNBT = {}` to set all of it at once.

> Note it is **fullNBT** not **nbt**, because kubejs uses `nbt` for a different purpose. A bit confusing, but it is what it is.

# Item Entities

There are two ways to create item entities in KubeJS.

## popItem

If you want to easily create the item from a certain block then you can use the `popItem` method.

### Example

```
block.popItem('minecraft:diamond')
```

The item can be an Item.of() instead if you wish

## createEntity("item")

Creating an item entity with a little more control be done identically to any other entity, except you get a couple more methods.

### Example

```
let itemEntity = block.createEntity("item")
itemEntity.y+=0.8
itemEntity.x+=0.5
itemEntity.z+=0.5
itemEntity.item = Item.of("encahanted_book").enchant("thorns",2)
```

```
itemEntity.item.count = 1
itemEntity.pickupDelay = 600
itemEntity.noGravity = true
itemEntity.motionY = 0.08
itemEntity.spawn()
```

In this example

- the **.item** beaned method is used to set the item of the item stack **(Required)**
- the **.pickupDelay** beaned method is used to set the pickup delay (Optional)

# Examples

Spawns an endermite when braking dirt with a 5% chance

```
onEvent("block.break", event => {
  if (event.block.id != "minecraft:dirt" || Math.random()  > 0.05) return
    //only if its dirt and only has 5% chance
    let myEndermite = event.block.createEntity("endermite")
    myEndermite.x += 0.5
    myEndermite.y += 0.5
    myEndermite.z += 0.5
    myEndermite.spawn()
})
```

Turns gravel to sand and drops clay when right clicked with flint

```
onEvent('block.right_click', event => {
  if (event.block.id == 'minecraft:gravel' && event.item.id == 'minecraft:flint') {
    event.block.set('sand')
    event.item.count--
    event.block.popItem('clay')
  }
})
```

Overrides enchanting table behavior when clicking on it with an item in you hand. Instead will make the item float up a while, then fall back down.

```
onEvent('block.right_click', event => {
    if (event.block.id !='minecraft:enchanting_table') return
    if (event.item.count == 0) return
```

```
  event.cancel()
    let item = event.item.copy()
    //if did not use .copy() the item would still be referencing the one in the hand, so setting the count to 1 would set the count in the hand to 1
    item.count = 1
  event.item.count--
  
    let itemEntity = event.block.createEntity('item')
    itemEntity.y+=0.8 // on the top of the encahnting table, not in it
    itemEntity.x+=0.5
    itemEntity.z+=0.5
    itemEntity.item = item
    itemEntity.item.count = 1
    itemEntity.pickupDelay = 100
    itemEntity.noGravity = true
    itemEntity.motionY = 0.08
    itemEntity.spawn()
  
  function callback (i) {
    //changes the scope of itemEntity (otherwise if used 2 times in a row within 5 seconds, problems would occur)
    event.server.scheduleInTicks(100, callback => { // this code runs 5 seconds later
    i.noGravity = false
    })
    }
  callback(itemEntity)
})
```

---

Revision #4
Created 2022-10-02 06:33:41 UTC by Q6
Updated 2022-12-30 23:44:52 UTC by Q6