Cloning

This sheet explains how to create lots of copies of a sprite, without copy-pasting code. For example, making a bullet every time you tap, or making a new enemy every second.

Functions

We can create more than one copy of a sprite by copying its code. But we don't want to copy/paste the code every time we make a sprite!

We can write a **function** to make this easier. (This is just like a custom block in Scratch.) A function lets you give a name to a piece of code. When we want to run that code, we can refer to it by name, rather than copy/pasting the code again.

A function looks something like this. You can call it whatever you like, it doesn't have to be `makeCow`.

```
function makeCow() {
    ...
}
```

We then need to write the inside of our function, so our program knows how to make this sort of sprite. Let's start writing our function:

```
function makeCow() {
    var cow = new Sprite
    cow.costume = '🐮'
    ...
}
```

Let's make one copy of our sprite, by "calling" our function. Add this at the **end of your program**, after the last `}` of our function.

```
makeCow()
```

The brackets are important – they tell JavaScript that this is a function we want to run.

👉 Save, and check that the new sprite appears.

Copying a sprite

Now we can easily create 100 copies of our sprite. This needs to go at the bottom of your program **below the function**, after the final `}` for the function.

```
for (let num of uw.range(100)) {
    makeCow()
}
```

👉 Make sure there are 100 sprites created. You'll need to randomize their positions (or something) or they'll all be overlapping each other in the middle!

Delaying the copies

Or, if we wanted to make a new copy every 3 seconds, we could use `setInterval`. This allows us to give a function that we want to execute each time and the amount of time to wait in between.

```
setInterval(function() {
    makeCow()
}, 3000)
```

Note that the amount of time is in milliseconds (3000ms = 3seconds).

If you wanted to only wait *once*, rather than multiple times, you can use `setTimeout` in exactly the same way.

For more information on setTimeout and setInterval, including how to stop a `setInterval` from executing from running using `clearInterval`, have a look here.

On tap

We can use our new function to create a sprite when you tap the screen.

```
world.onTap(e => {
    makeCow()
})
```

Moving clones

It's important to note that the `cow` variable can only be used inside our `makeCow` function. **Variables in a function can only be used in that function.**

This means we can make variables specific to a clone by putting them inside the function. For example, we might decide to give our sprites gravity.

We'll start them off with a random amount of upward force. Put this **at the top of your function**.

```
function makeCow() {
    var speedY = uw.randomInt(0, 10)
    ...
```

Now let's make them fall due to gravity. Add this `forever` block **at the bottom of your function**, just before the final `}`.

```
    cow.forever(() => {
        cow.posY += speedY
        speedY -= 0.5
    })
```

Your final function should look something like this:

```
function makeCow() {
    var speedY = uw.randomInt(0, 10)

    var cow = new Sprite
    cow.costume = '🐄'

    cow.forever(() => {
        cow.posY += speedY
        speedY -= 0.5
    })
}
```

👉 Save, and check the sprites fall down independently.

Each sprite made by our function gets its own `speedY`.

Destroying clones

If we keep creating sprites, but never destroy them, then the number of sprites in our world will grow and eventually our game will get really slow! We can fix that by destroying sprites once they're no longer needed.

For example, we can destroy sprites when they go off-screen. Add this **inside your sprite's `forever` block**.

```
    if (!cow.isOnScreen()) {
        cow.destroy()
    }
```

Destroying a sprite removes it from the screen permanently. This also stops any `forever` loops attached to it.