Forever

So far, our program doesn't do anything after it starts. Nothing will change, unless we change our program and re-run it!

Animation means to give an appearance of movement. We do this by **changing values over time**.

We need a way to react to time passing. We do this by using `forever` to run code on every *frame*. An app draws a new frame on the screen roughly 60 times a second (or 60 **FPS**, frames per second).

Spinning

Let's make our sprite spin. We do this by increasing the angle of the sprite a little bit on every frame, so it slowly rotates clockwise.

```
player.forever(() => {
    player.angle += 3
})
```

Make sure you get the brackets exactly right!

👉 Save, and check your sprite starts spinning.

In JavaScript, `number += 1` is shorthand for `number = number + 1`. You can read it as **"change by"**.

Everything we've written before this point will only run once, at the start of our game. Anything **inside the forever block**, between the curly braces, will run **every frame** (unless our player is destroyed).

The number `3` is how far we rotate on each frame, so it controls **how fast** the animation happens.

👉 Experiment with changing the number, to get the sprite to rotate faster and slower.

Spinning backwards

👉 How can you rotate anticlockwise?

Hint: it's the opposite of rotating clockwise…

Moving

Time to get our sprite to move. Like before, we do this by changing its attributes a little bit every frame.

Let's move it left. Add this **inside the forever**:

```
    player.posX -= 2
```

👉 Check the sprite starts moving left.

Ending "forever"

Despite the name, a `forever` loop doesn't have to run forever! Sometimes it's useful to only repeat a bit of code until something happens.

If we write `return false`, then this will stop the `forever` loop. We normally do this inside an `if` condition (otherwise the `forever` block would only run once!).

For example, we can move our sprite to the left until it hits the left edge of the screen.

```
player.forever(() => {
    player.posX -= 2
    if (player.left < 0) {
      return false
    }
})
```

👉 Check the sprite moves to the left edge and then stops.

Growing and shrinking

We can animate any attribute of a sprite: `scale`, `opacity`, and so on.

For example, we could make our sprite grow by gradually increasing the scale by 5%. We would do that by multiplying it by the fraction `1.05` (again this would have to be inside the forever):

```
    player.scale *= 1.05
```

➡️ Next, let's see how to react to taps…