Luxembourg Tech School

LEVEL GO
2025-2026

6 – Loops in p5.js

What will we cover?

2 important coding concepts
that you need to understand to be able to code cool things:

REUSABILITY:

Functions

REPETITION:

Loops

←
today

Recap Quiz

We have covered a lot so far!

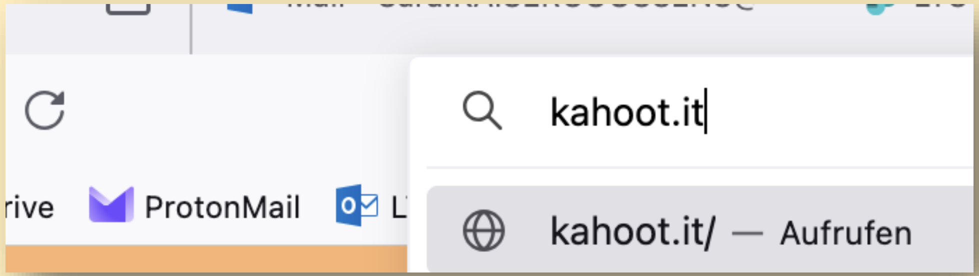
To refresh everyone’s memory, we have a p5.js recap Kahoot!

TASK

1

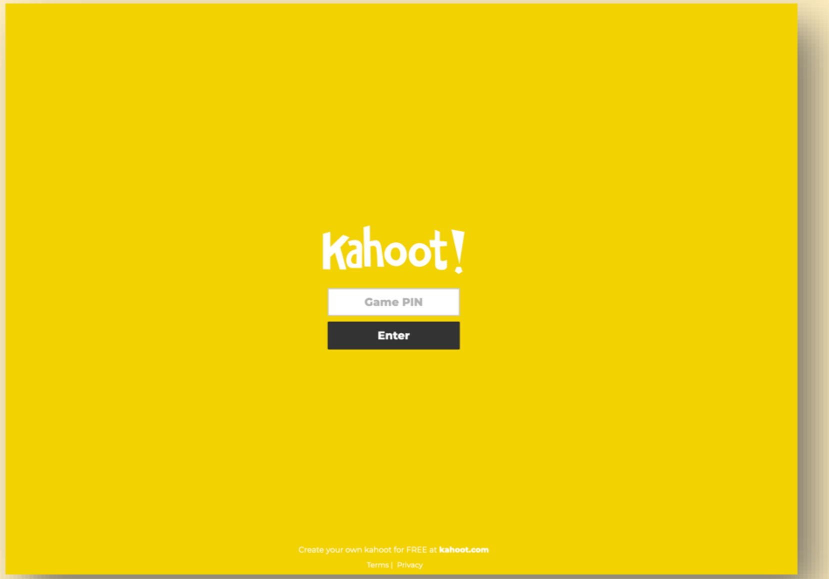
Open a new browser window and go to the website:

kahoot.it



2

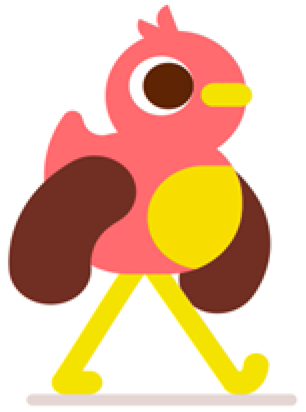
Type the **Game PIN** that will be displayed and give yourself a **(nick)name**.



Loop

Walking with code: 2 possibilities

Do each step separately = code each step individually



1. Take a step
2. Take a step
3. Take a step
4. Take a step
5. Take a step
6. Take a step
7. Take a step
8. Take a step
9. Take a step
10. Take a step

Use a loop to run a code repeatedly and take steps without having to code each single one

1. If steps are less than 10
2. Then take a step

Computers are really good at **repeating** things

And they don't get bored.

They like performing the same task over and over again **until specific criteria are met**.

Image: Charlie Chaplin in Modern Times (1936)

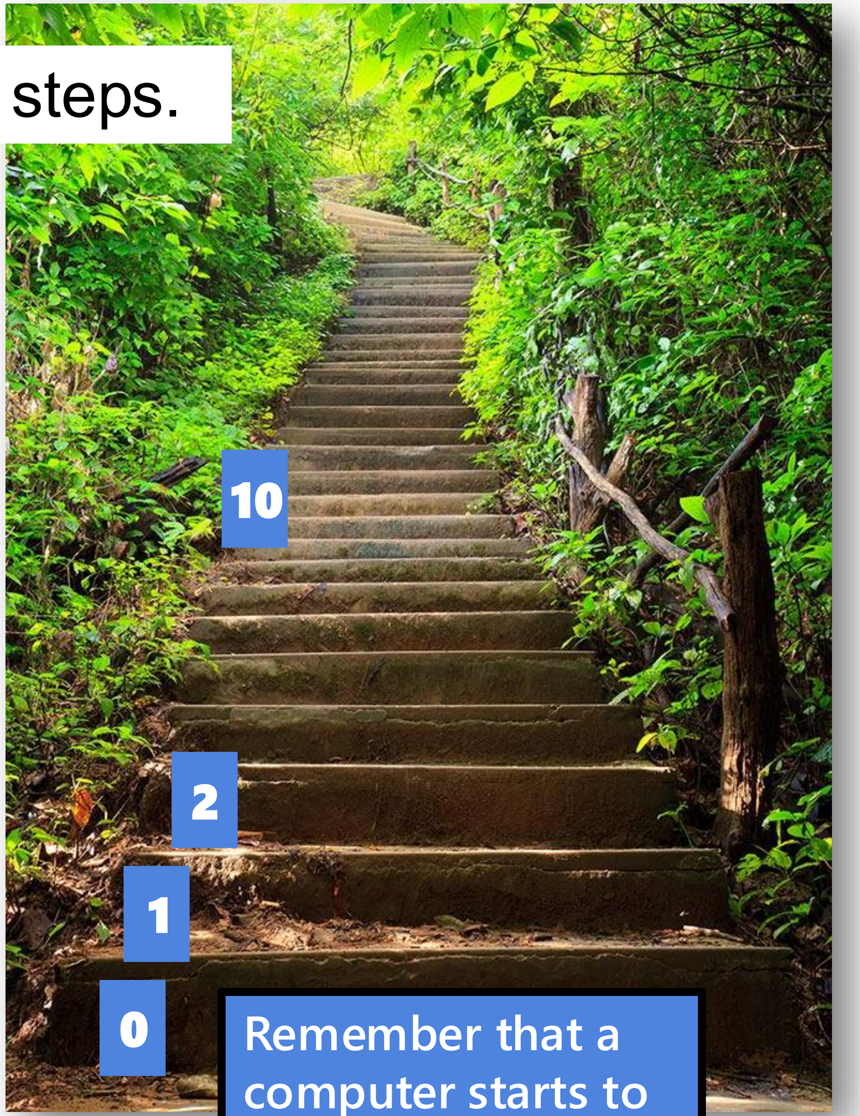
How loops work

Before we dive into **for loops**, let's think about climbing steps.

1. Which step should I **start** from?
2. Until which step (**stop** point) should I continue?
3. How many **steps** should I climb every time?

Do the climbing

repeat



Remember that a computer starts to count at 0.

How loops work

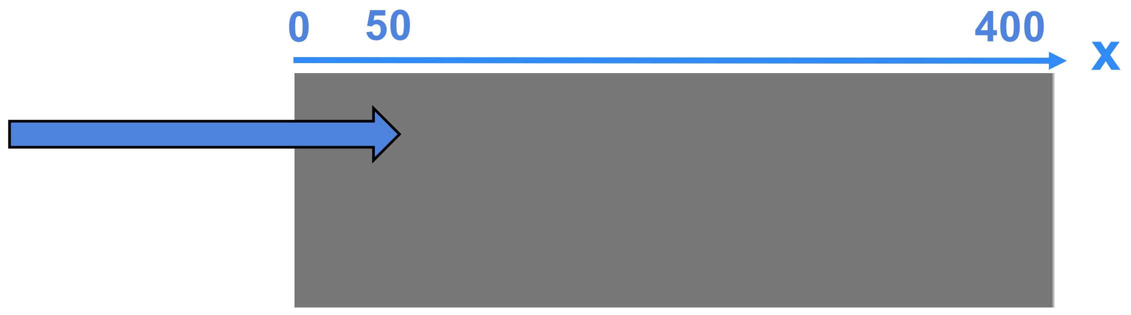
1 Start at 50 px

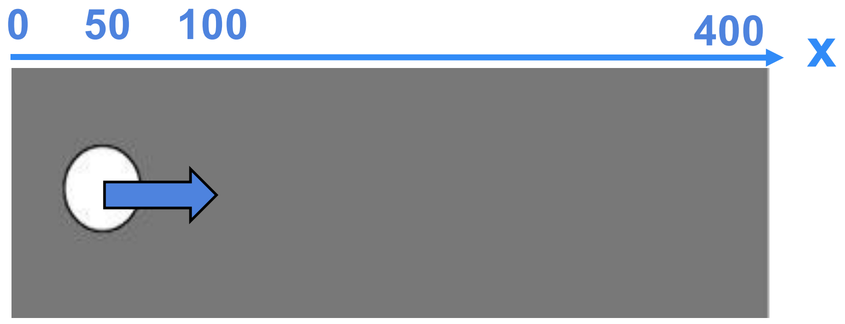
2 Continue until the edge of the canvas (400 px is the stop point)

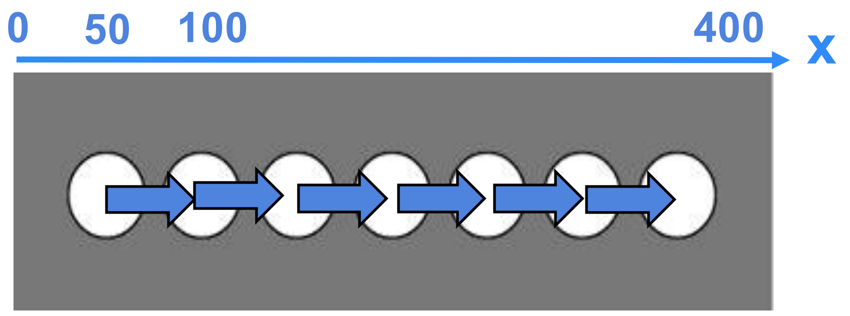
3 Move every time 50 px to the right

Draw an ellipse

repeat







Our for loop explained (1)

1 **START**

Creates a new loop counter to keep track of the pattern, a variable called x with a starting value of 50

2 **CHECK**

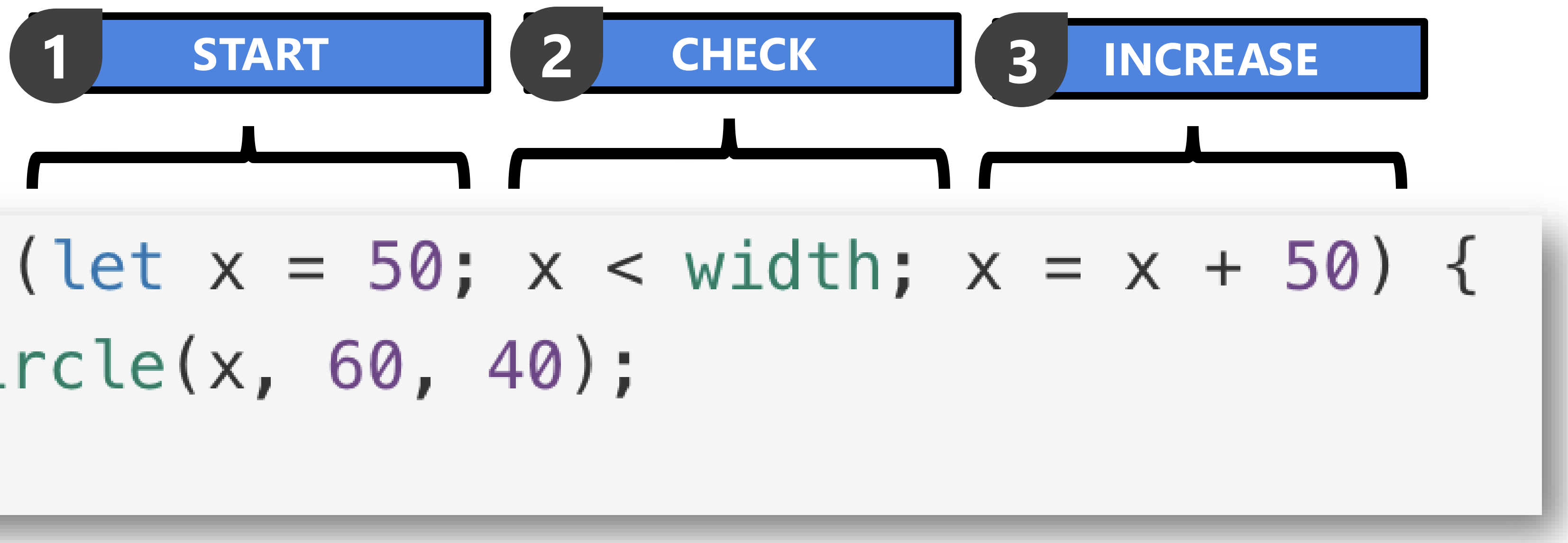
Every round see if the loop should continue. The loop repeats for as long as x is less than our width of the canvas

3 **INCREASE**

Every round the loop counter x is increased by 50

```
for (let x = 50; x < width; x = x + 50) {
  circle(x, 60, 40);
}
```

Our for loop explained (2)

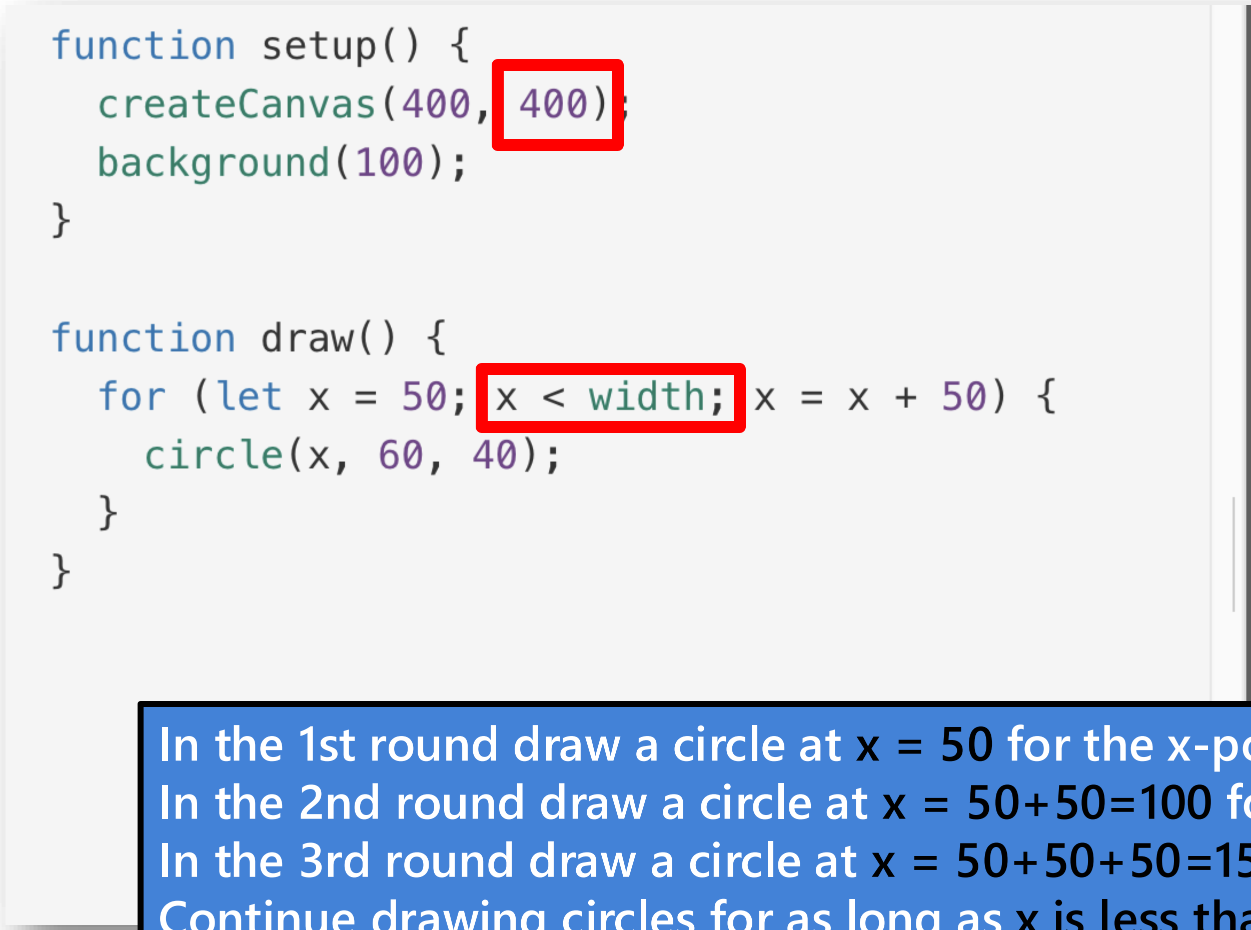


```
for (let x = 50; x < width; x = x + 50) {
  circle(x, 60, 40);
}
```

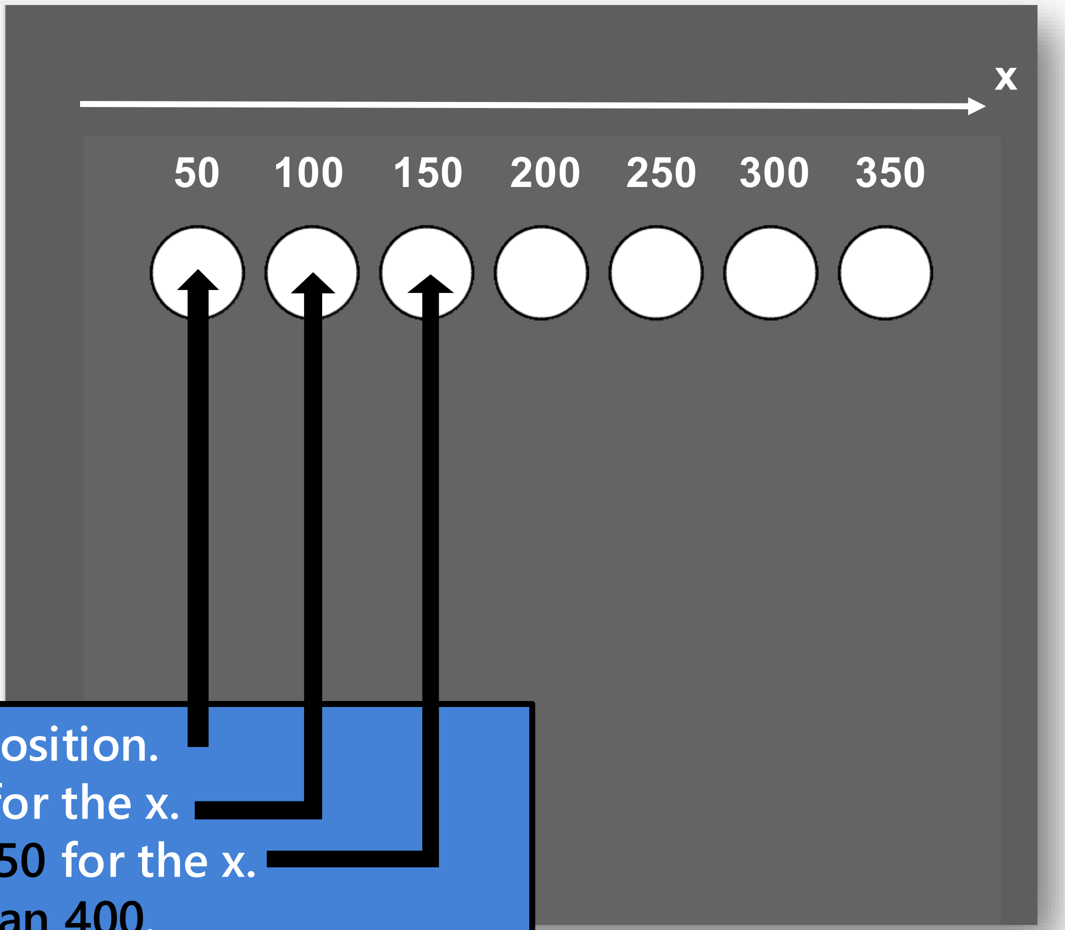
This might seem like a lot to take in, but you can think about it as a few steps:

1. A loop variable **x** is created and initialized to the first number in the pattern.
2. The check is evaluated, and if it's ***false***, the loop exits, and its body is skipped.
   If it's ***true***, then the body is executed.
3. After the body executes, the loop variable **x** is updated.
4. Then the code jumps back to the beginning of the loop and performs the check again.

Our for loop explained: full sketch (3)



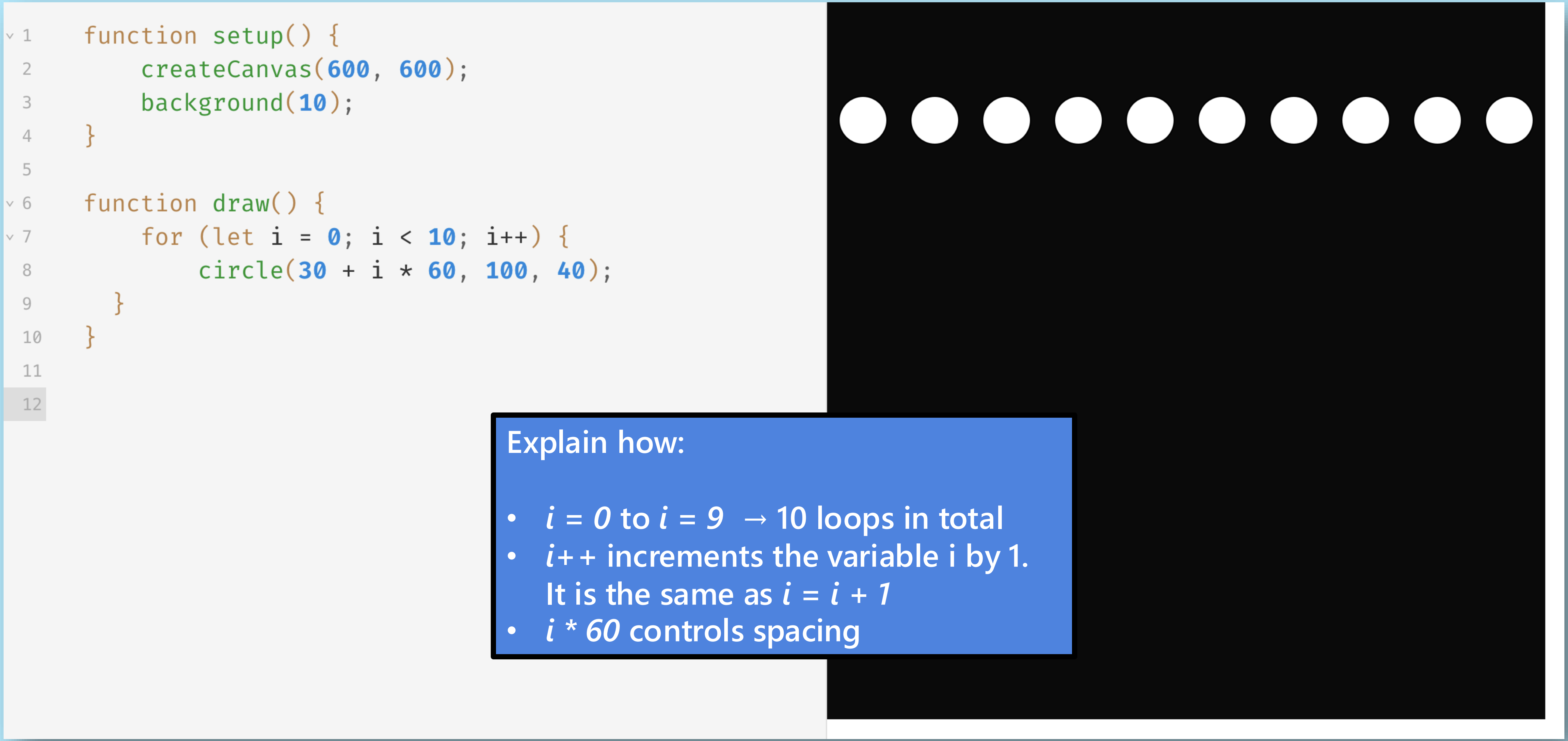
```
function setup() {
  createCanvas(400, 400);
  background(100);
}

function draw() {
  for (let x = 50; x < width; x = x + 50) {
    circle(x, 60, 40);
  }
}
```



In the 1st round draw a circle at x = 50 for the x-position.
In the 2nd round draw a circle at x = 50+50=100 for the x.
In the 3rd round draw a circle at x = 50+50+50=150 for the x.
Continue drawing circles for as long as x is less than 400.
As soon as x reaches 400, stop drawing circles.

Activity

Let's try out a for loop

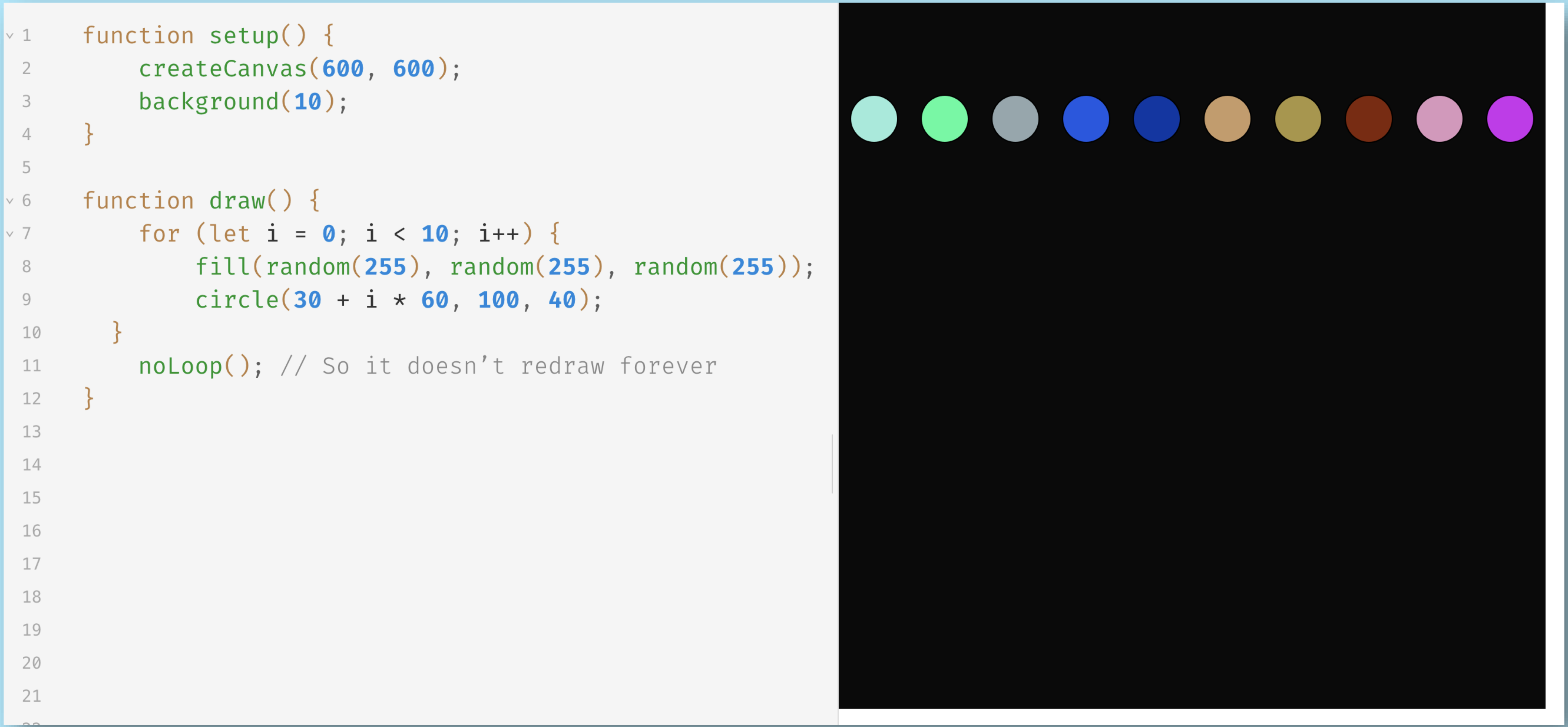


```
function setup() {
    createCanvas(600, 600);
    background(10);
}

function draw() {
    for (let i = 0; i < 10; i++) {
        circle(30 + i * 60, 100, 40);
    }
}

```

Explain how:

- *i = 0* to *i = 9* → 10 loops in total
- *i++* increments the variable i by 1. It is the same as *i = i + 1*
- *i * 60* controls spacing

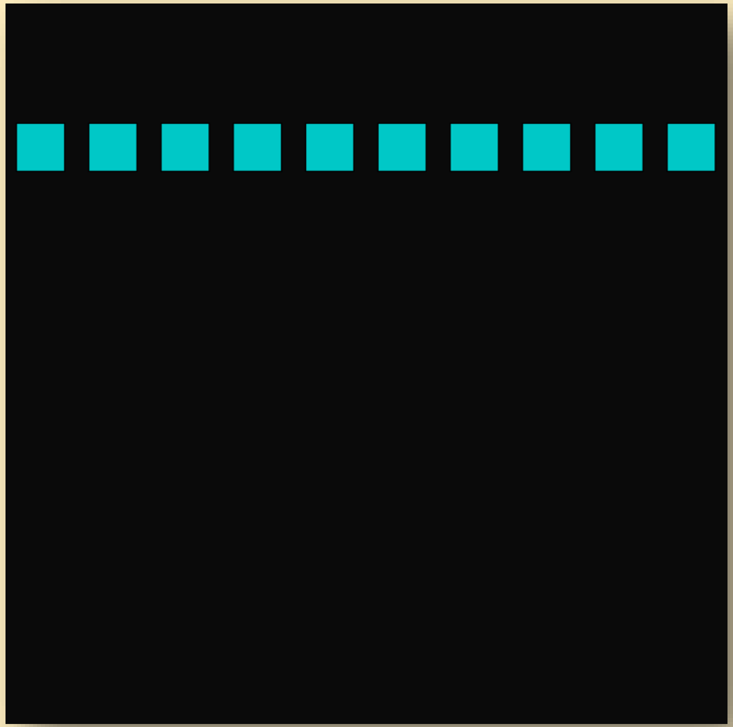
Add some random colours for fun



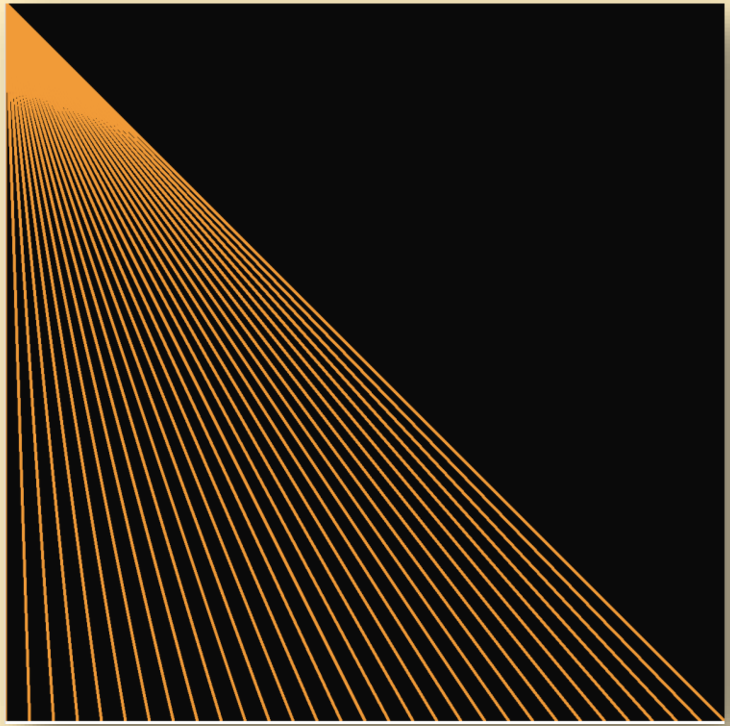
```
function setup() {
    createCanvas(600, 600);
    background(10);
}

function draw() {
    for (let i = 0; i < 10; i++) {
        fill(random(255), random(255), random(255));
        circle(30 + i * 60, 100, 40);
    }
    noLoop(); // So it doesn't redraw forever
}
```

Use a for loop to repeat other shapes

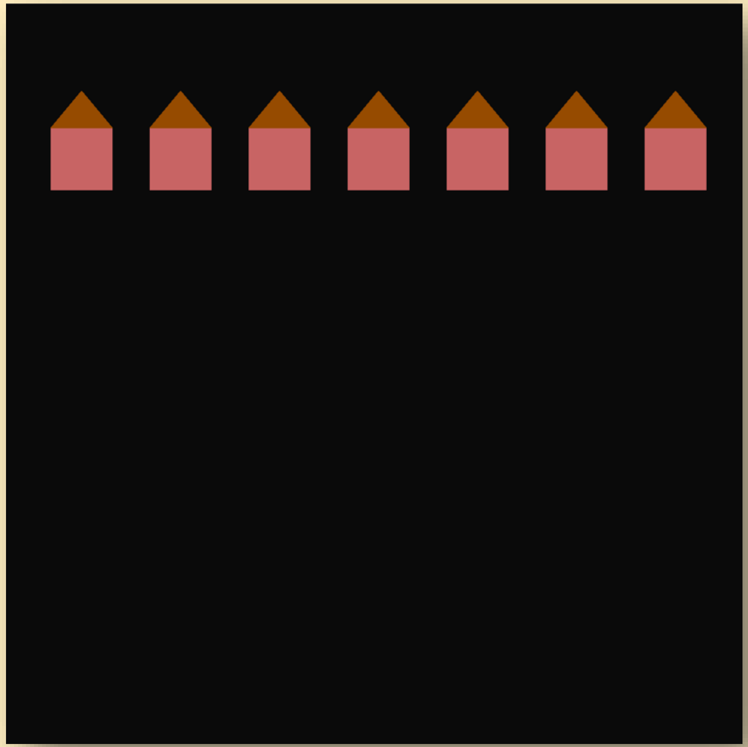
Try to recreate one of the following - *or make your own version*:



`rect(x, y, w, h, detailX?, detailY?)`

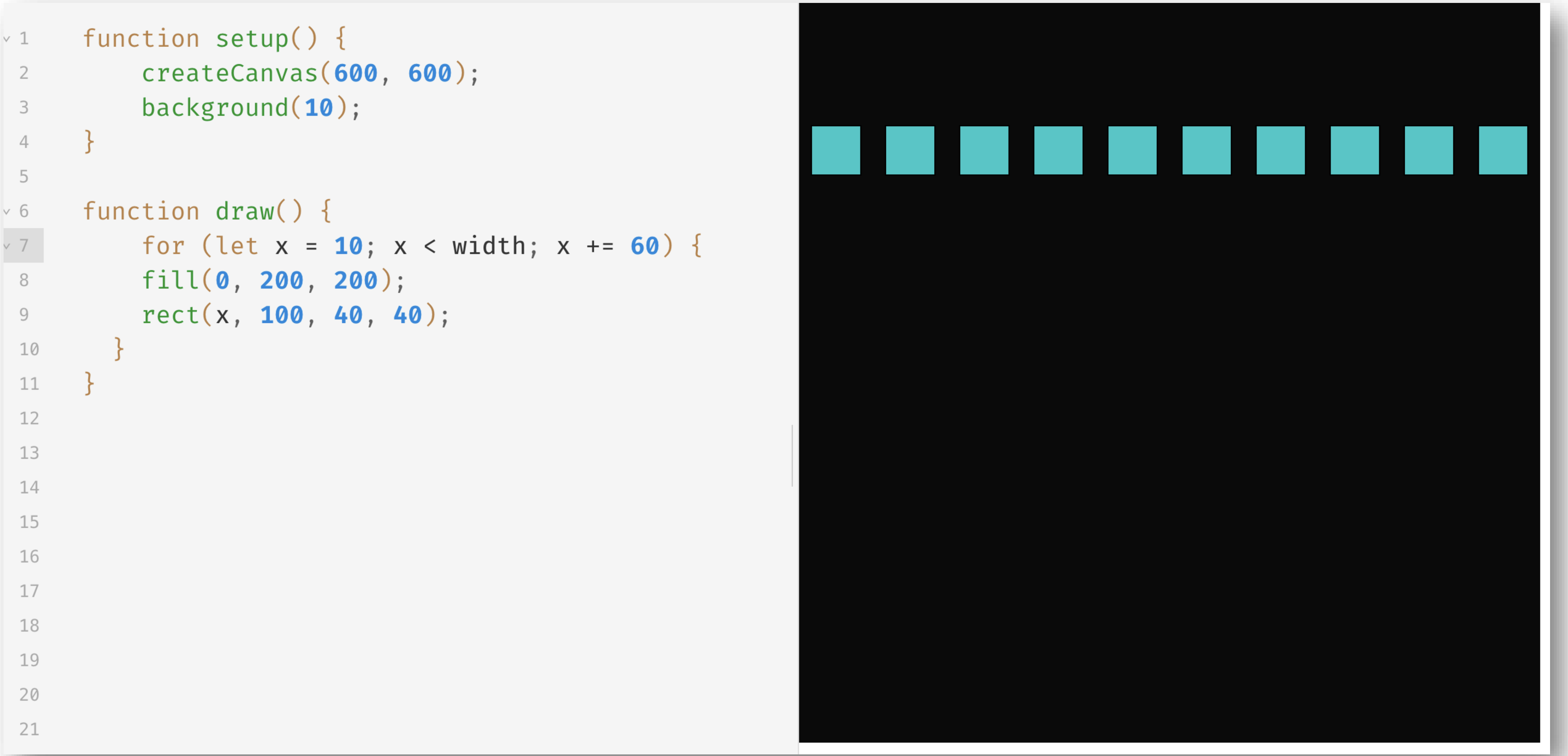


`line(x1, y1, x2, y2)`



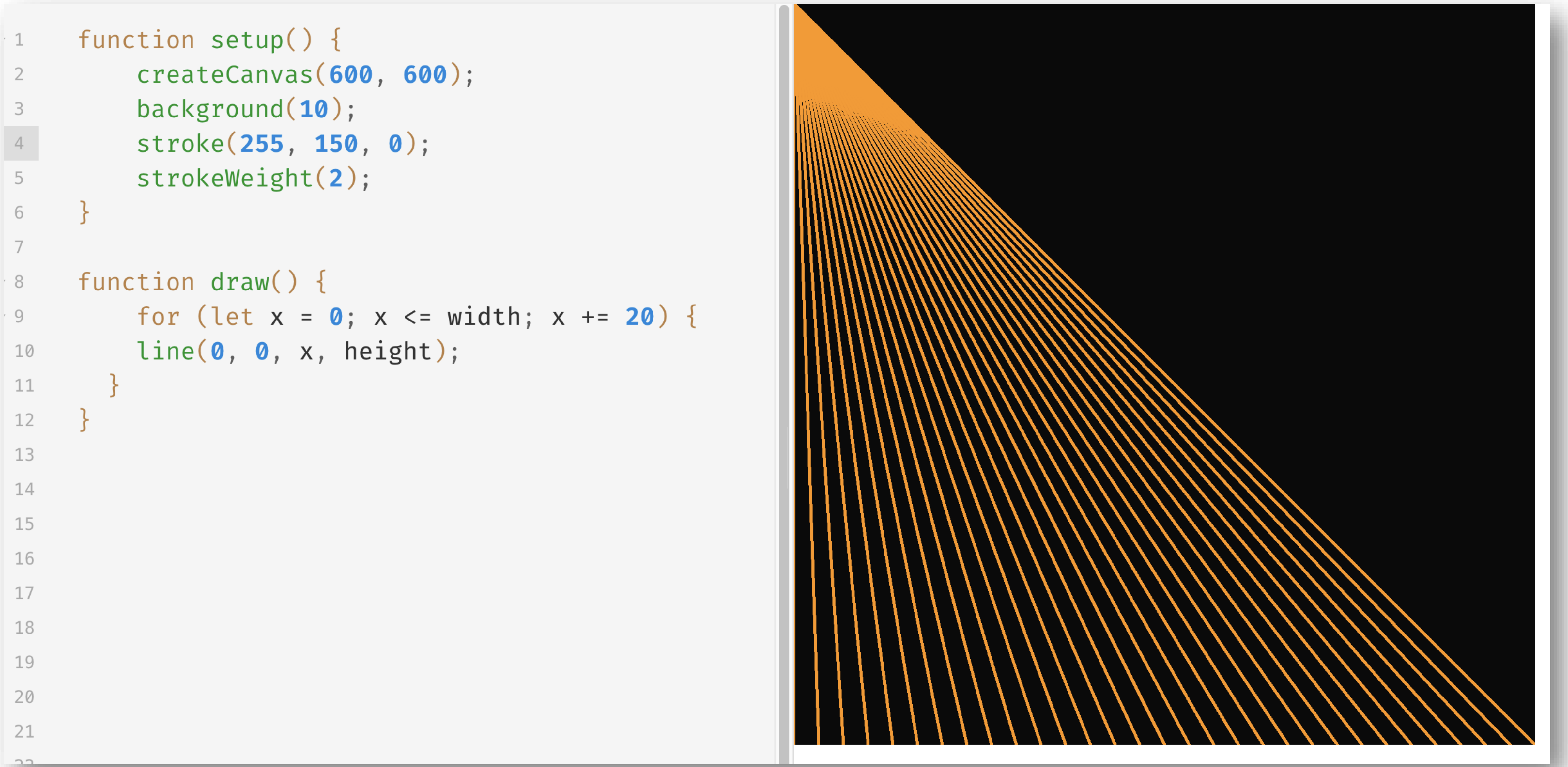
`triangle(x1, y1, x2, y2, x3, y3)`

Example: Loop with *rect()*



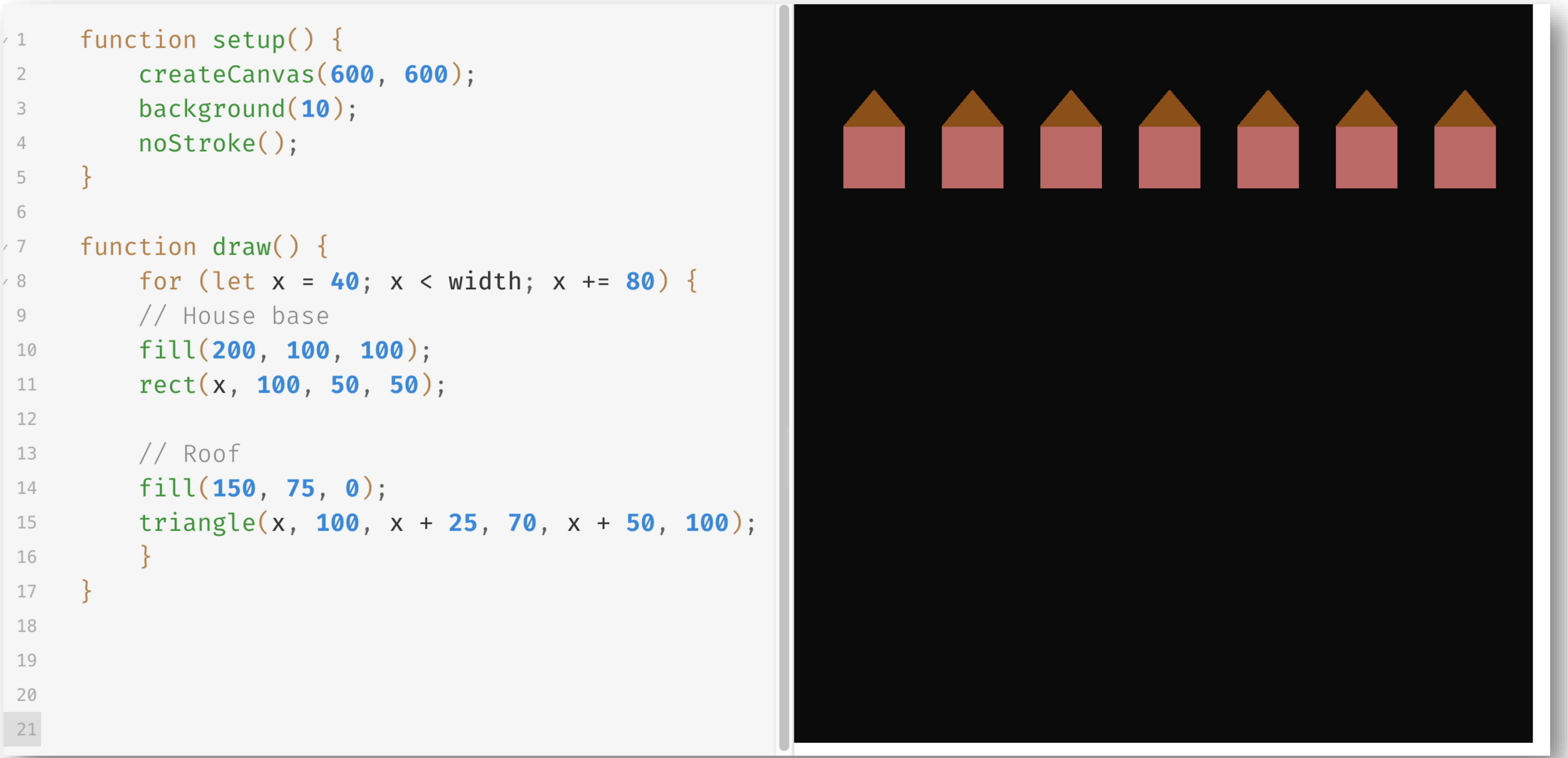
```
function setup() {
    createCanvas(600, 600);
    background(10);
}

function draw() {
    for (let x = 10; x < width; x += 60) {
    fill(0, 200, 200);
    rect(x, 100, 40, 40);
  }
}
```

Example: Loop with *line()*



```
function setup() {
    createCanvas(600, 600);
    background(10);
    stroke(255, 150, 0);
    strokeWeight(2);
}

function draw() {
    for (let x = 0; x <= width; x += 20) {
    line(0, 0, x, height);
  }
}
```

Example: Loop with *rect()* + *triangle()*



```
function setup() {
    createCanvas(600, 600);
    background(10);
    noStroke();
}

function draw() {
    for (let x = 40; x < width; x += 80) {
    // House base
    fill(200, 100, 100);
    rect(x, 100, 50, 50);

    // Roof
    fill(150, 75, 0);
    triangle(x, 100, x + 25, 70, x + 50, 100);
    }
}
```

Nested Loops

Remember: a for loop

1 START

Creates a new loop counter to keep track of the pattern, a variable called x with a starting value of 50

2 CHECK

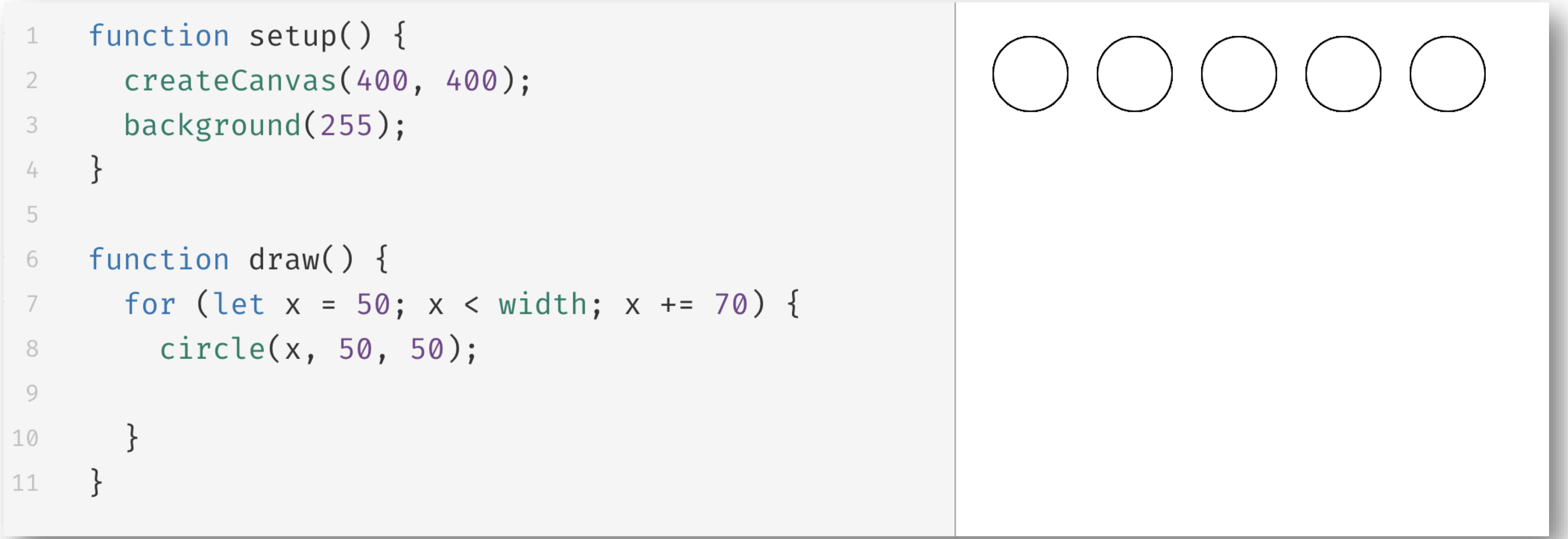
Every round see if the loop should continue. The loop repeats for as long as x is less than our width of the canvas

3 INCREASE

Every round the loop counter x is increased by 50

```
for (let x = 50; x < width; x = x + 50) {
  circle(x, 60, 40);
}
```

Intro to nested loops

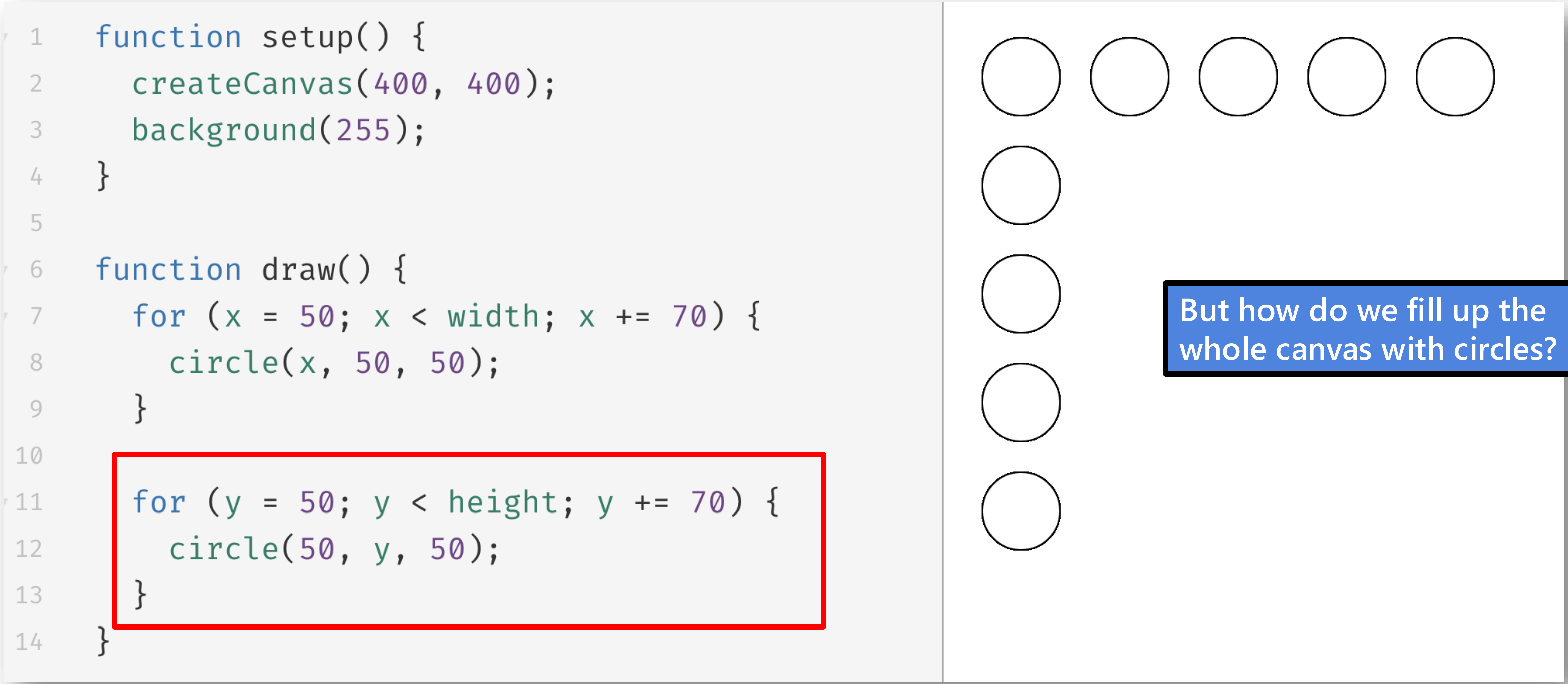


```
function setup() {
  createCanvas(400, 400);
  background(255);
}

function draw() {
  for (let x = 50; x < width; x += 70) {
    circle(x, 50, 50);

  }
}
```

With *x < width;* we draw circles all along the width of our canvas!

Add circles along the y-axis

We can add another for loop to draw circles along the y-axis.

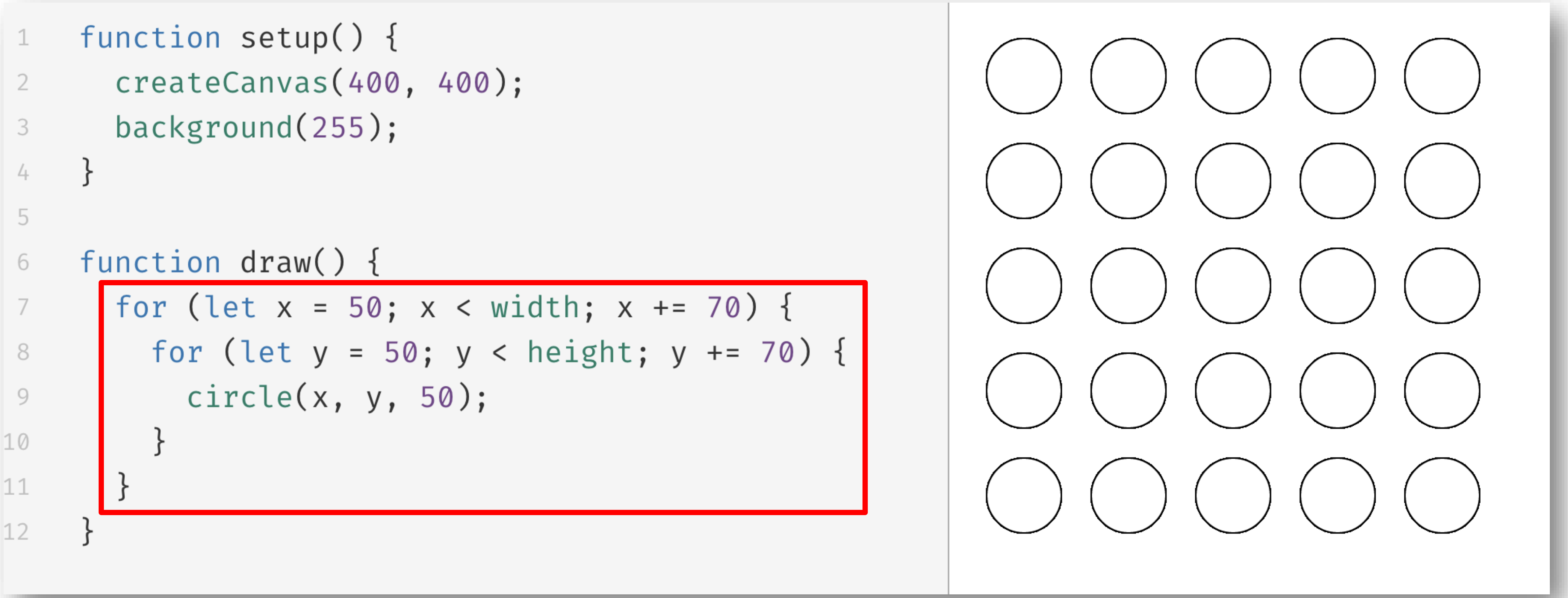


```
function setup() {
  createCanvas(400, 400);
  background(255);
}

function draw() {
  for (x = 50; x < width; x += 70) {
    circle(x, 50, 50);
  }

  for (y = 50; y < height; y += 70) {
    circle(50, y, 50);
  }
}
```

But how do we fill up the whole canvas with circles?

Fill the whole canvas with a nested loop

We have first an outer loop. And then add inside the first loop a second inner loop. This is called **a nested loop.**



```
function setup() {
  createCanvas(400, 400);
  background(255);
}

function draw() {
  for (let x = 50; x < width; x += 70) {
    for (let y = 50; y < height; y += 70) {
      circle(x, y, 50);
    }
  }
}
```

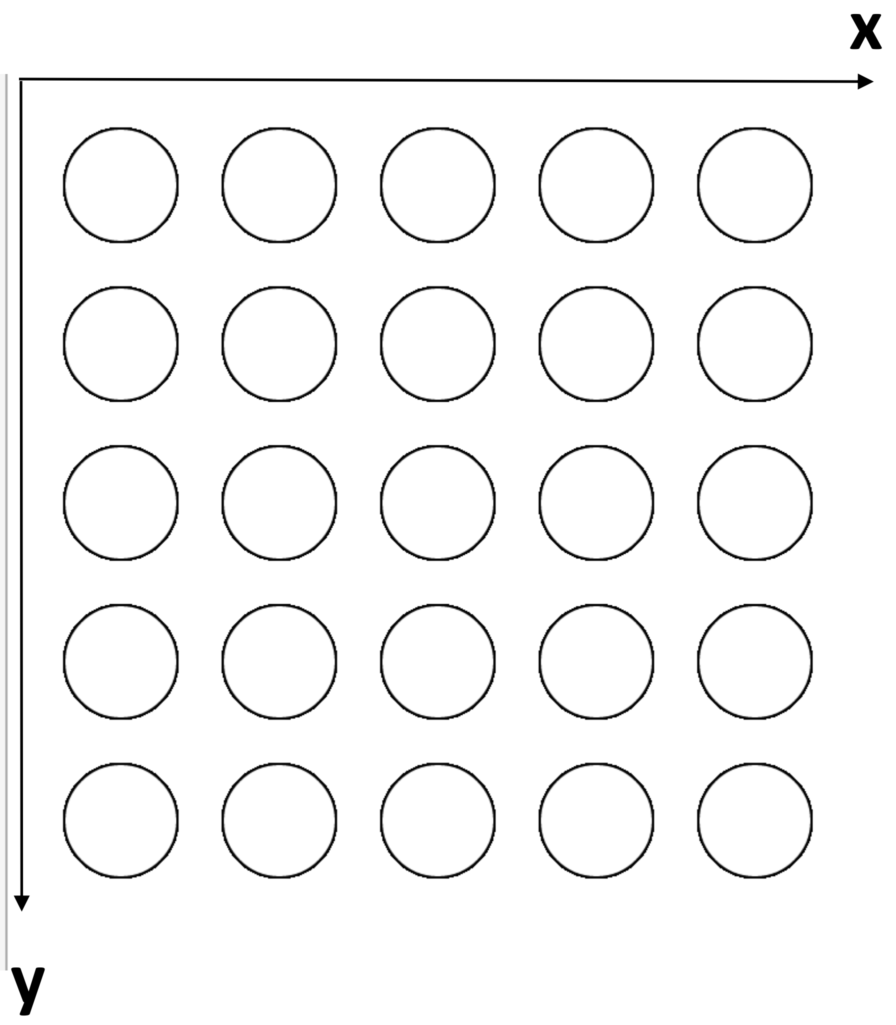
A nested for loop

```
function setup() {
  createCanvas(400, 400);
  background(255);
}

function draw() {
  for (let x = 50; x < width; x += 70) {
    for (let y = 50; y < height; y += 70) {
      circle(x, y, 50);
    }
  }
}
```

You can see the outer loop in green.

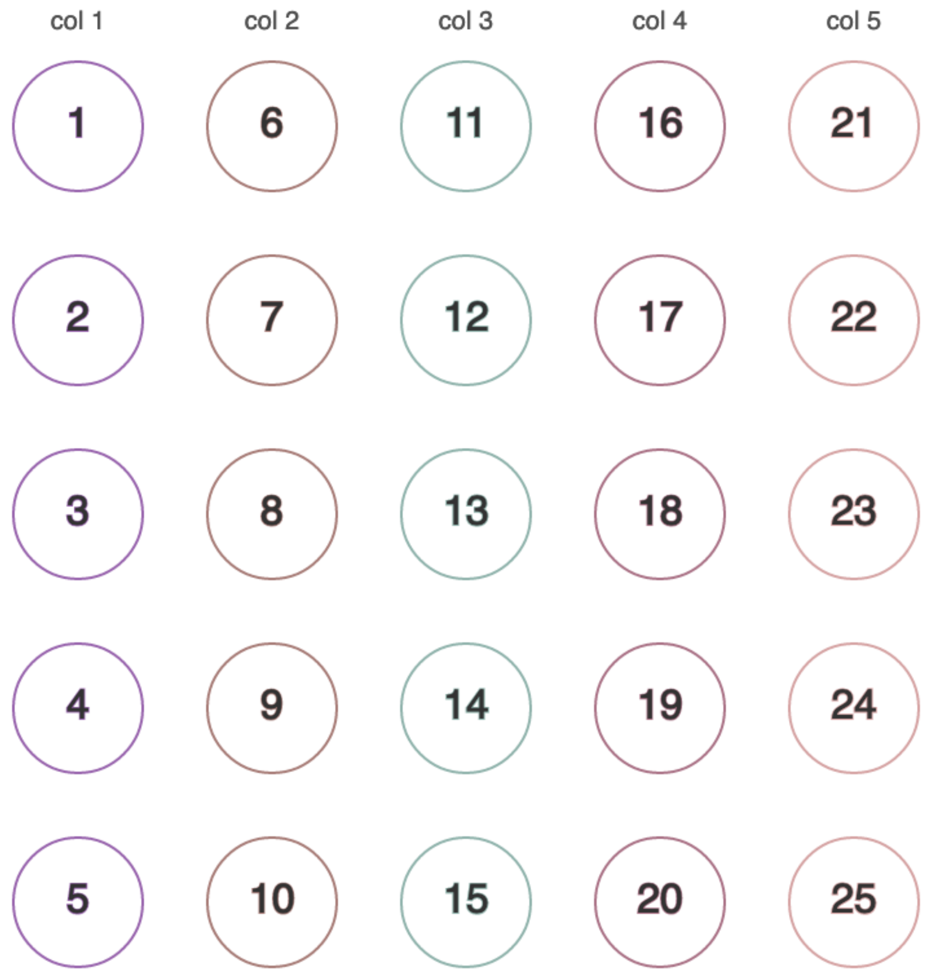
The repeated code is inside the curly brackets.
This is the inner loop.

And the code block (draw circles).

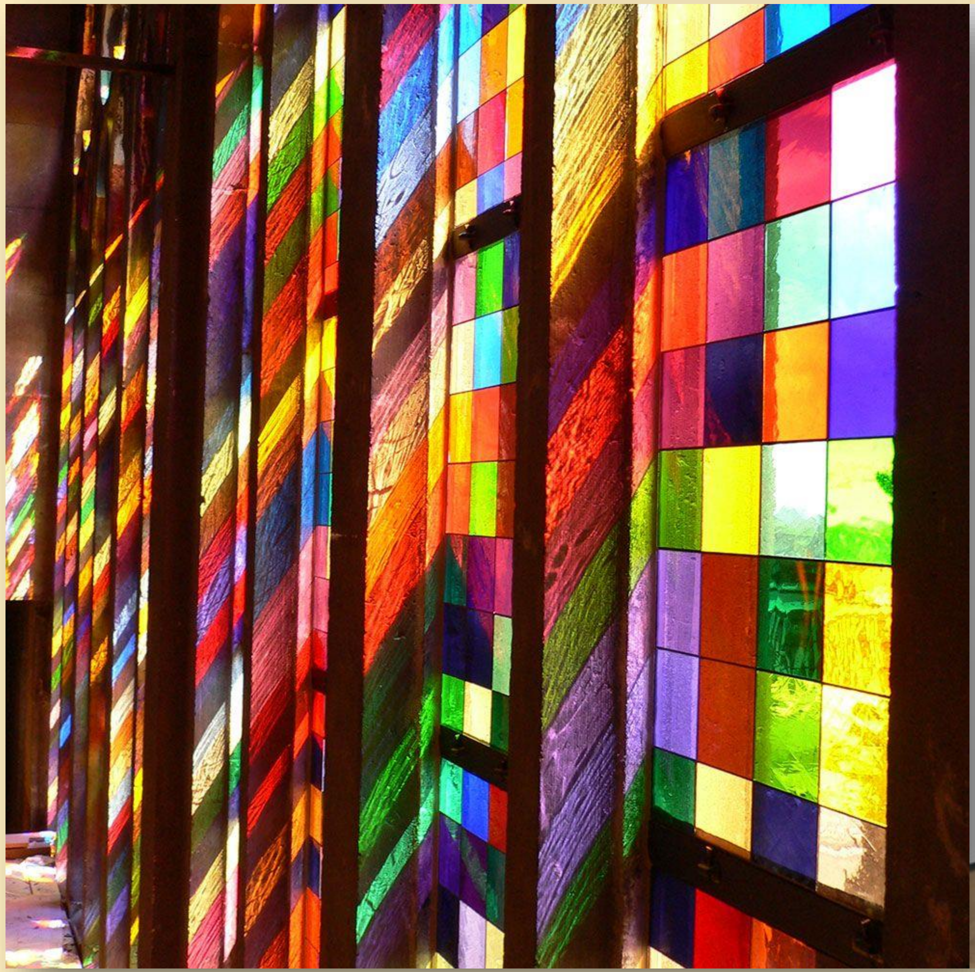


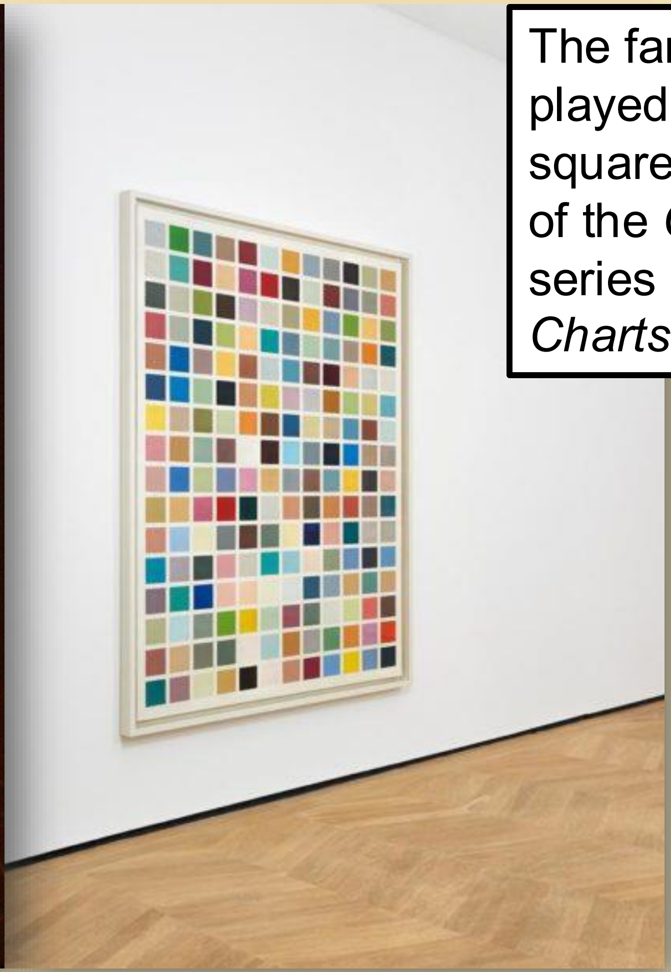
We have 2 loop counters: x and y

Nested loop order



Let’s create a colour grid





The famous artist *Gerhard Richter* played with such randomly colored squares for his creation of the windows of the *Cathedral in Cologne* and for a series of paintings called *Colour Charts*.

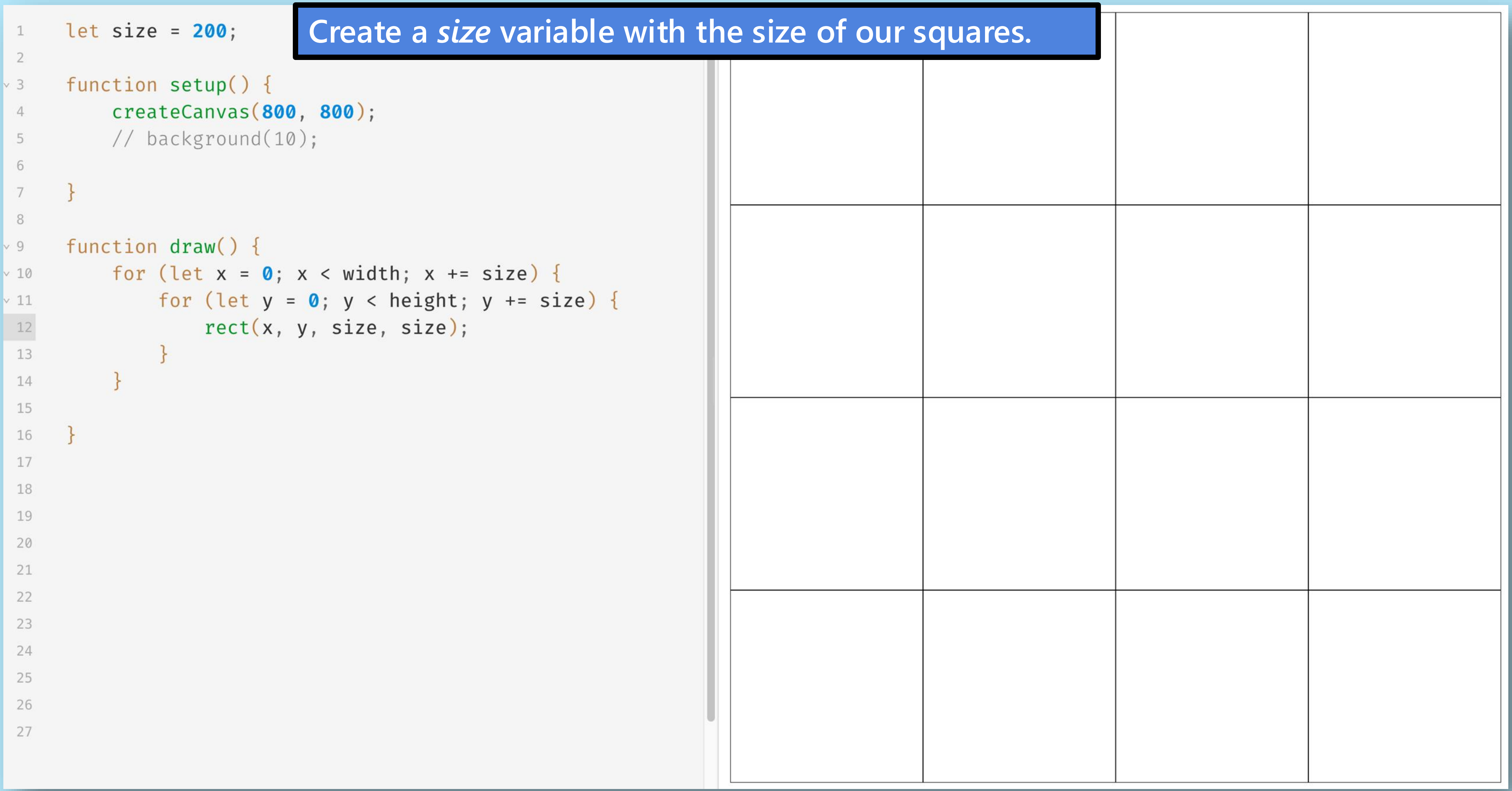
We will do the same with p5.js and even make it animated!

REPETITION:
Loops

Prepare the grid

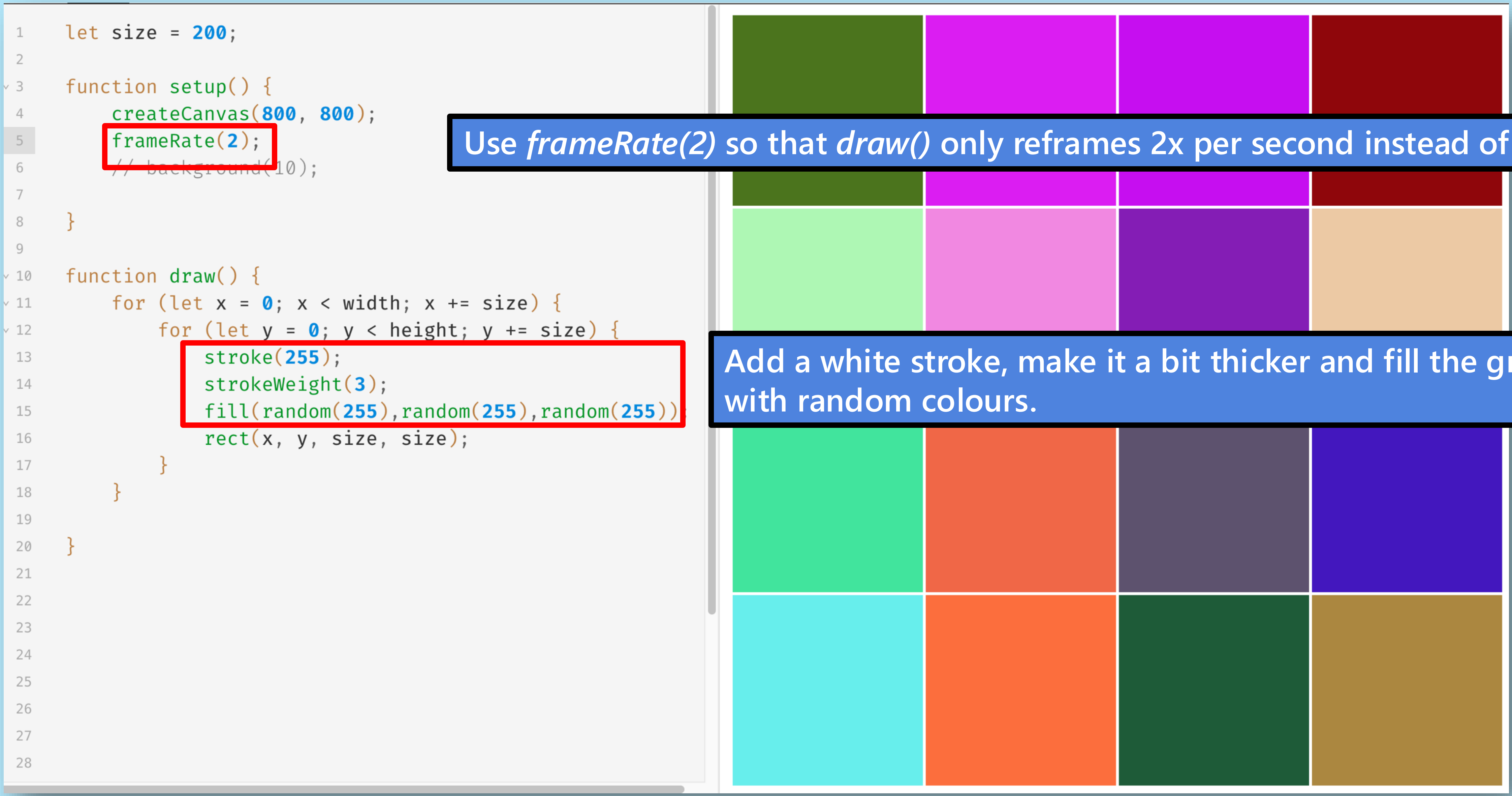
We want to have a grid with 4x4 squares.

Create a *size* variable with the size of our squares.



```
let size = 200;

function setup() {
    createCanvas(800, 800);
    // background(10);

}

function draw() {
    for (let x = 0; x < width; x += size) {
        for (let y = 0; y < height; y += size) {
            rect(x, y, size, size);
        }
    }

}
```

Fill the grid with random colours

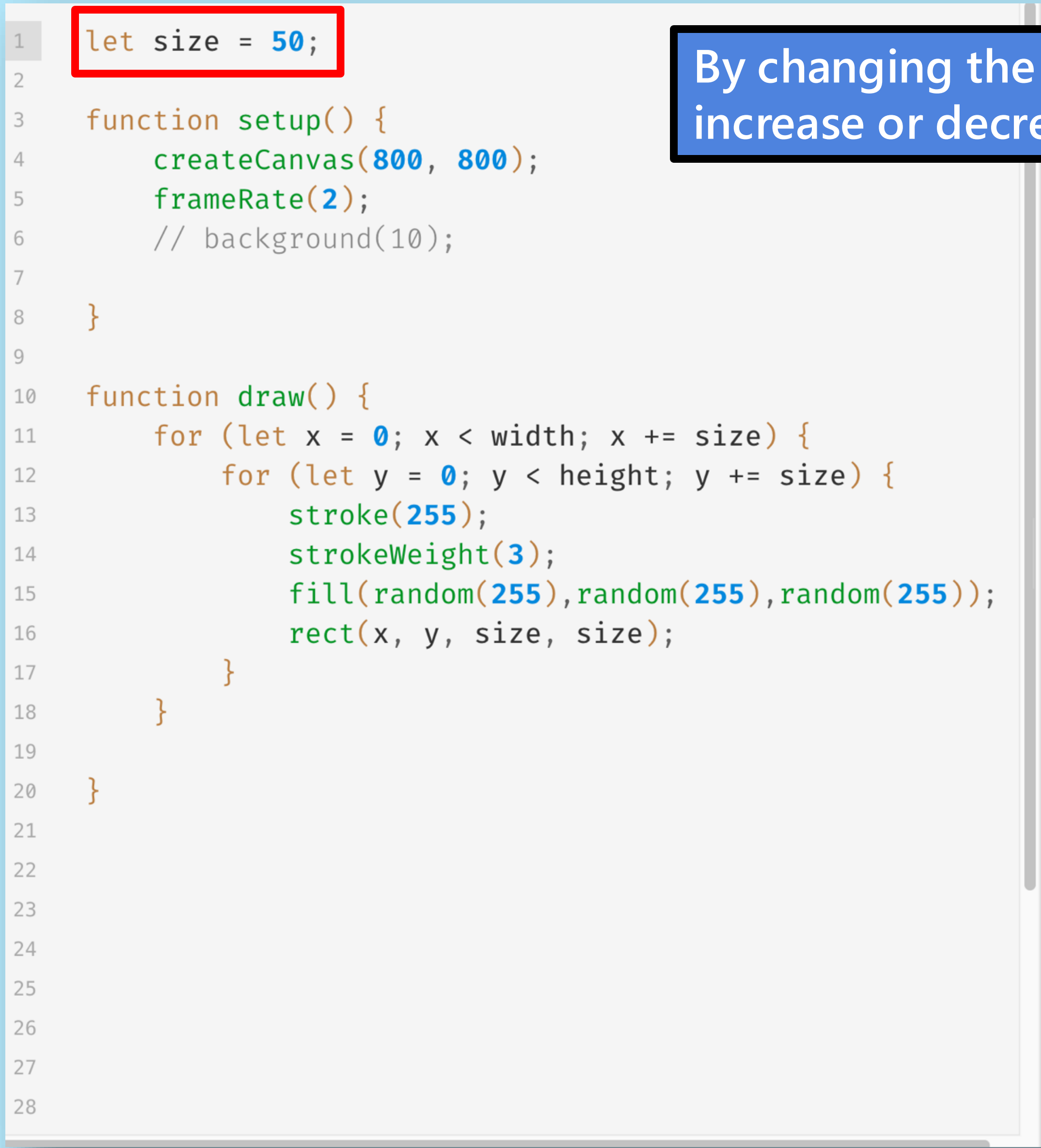


```
let size = 200;

function setup() {
    createCanvas(800, 800);
    frameRate(2);
    // background(10);

}

function draw() {
    for (let x = 0; x < width; x += size) {
        for (let y = 0; y < height; y += size) {
            stroke(255);
            strokeWeight(3);
            fill(random(255),random(255),random(255));
            rect(x, y, size, size);
        }
    }

}
```

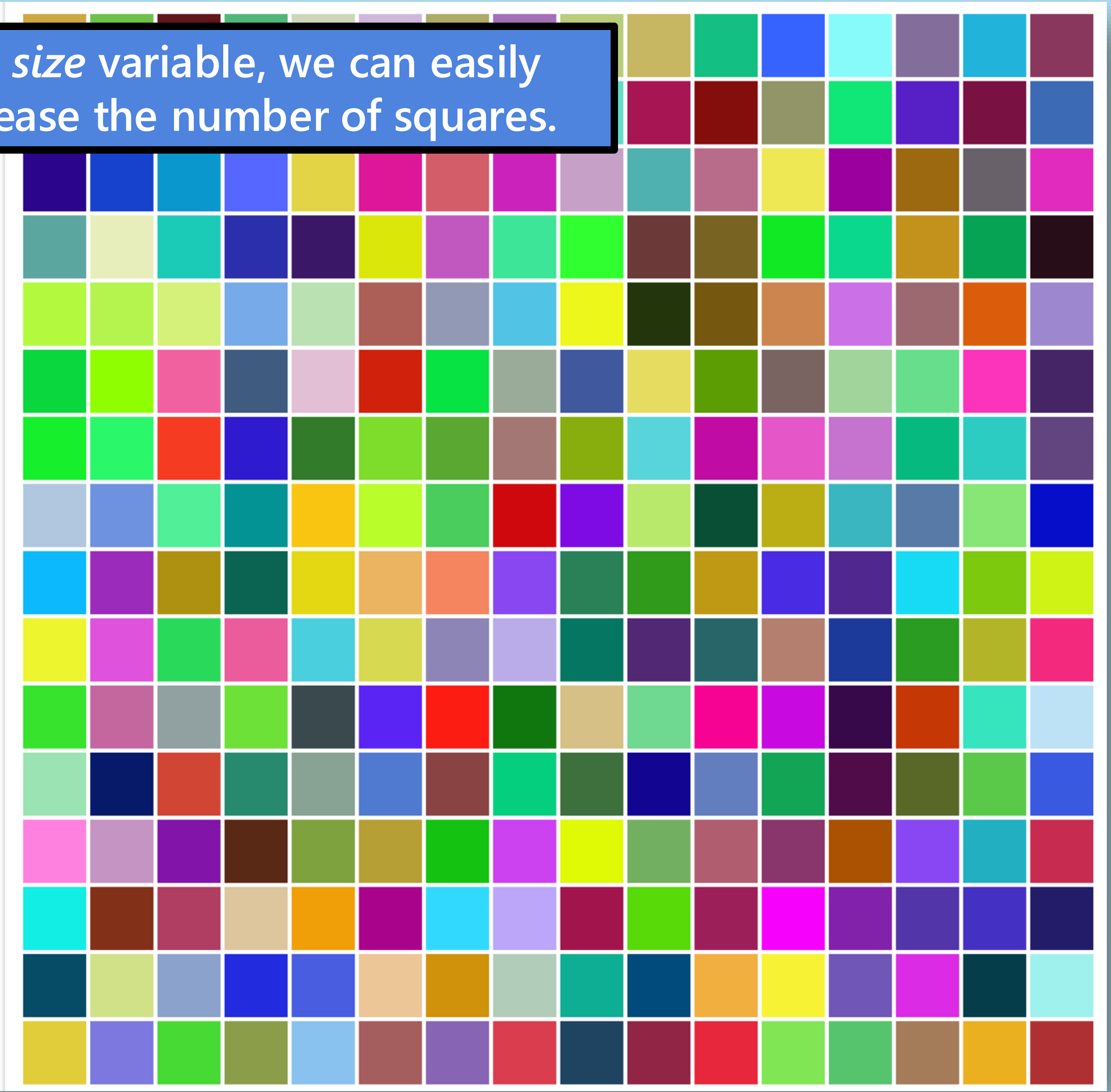
Use *frameRate(2)* so that *draw()* only reframes 2x per second instead of 60x

Add a white stroke, make it a bit thicker and fill the grid with random colours.

Make the squares smaller

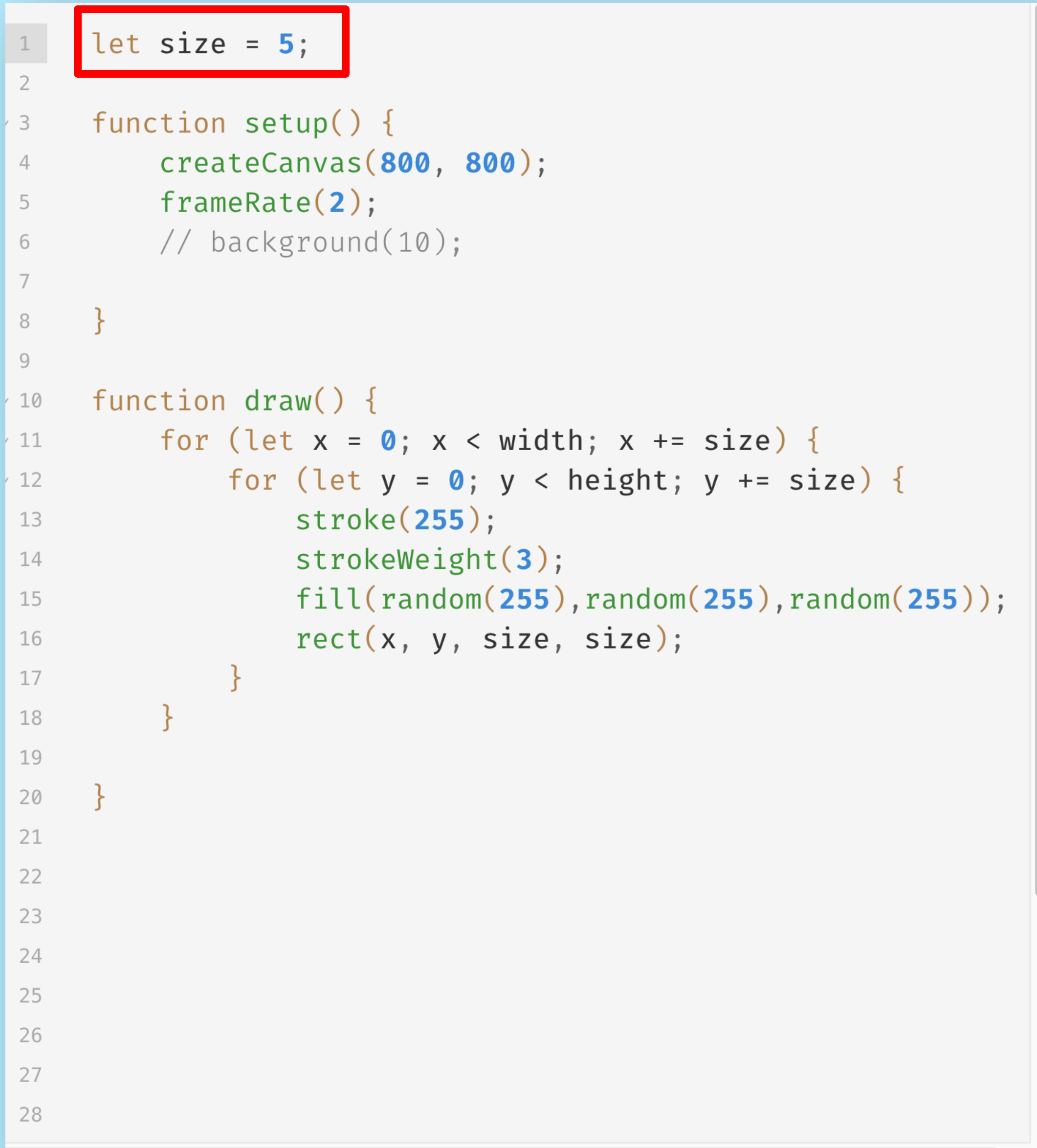


```
let size = 50;

function setup() {
    createCanvas(800, 800);
    frameRate(2);
    // background(10);

}

function draw() {
    for (let x = 0; x < width; x += size) {
        for (let y = 0; y < height; y += size) {
            stroke(255);
            strokeWeight(3);
            fill(random(255),random(255),random(255));
            rect(x, y, size, size);
        }
    }

}
```

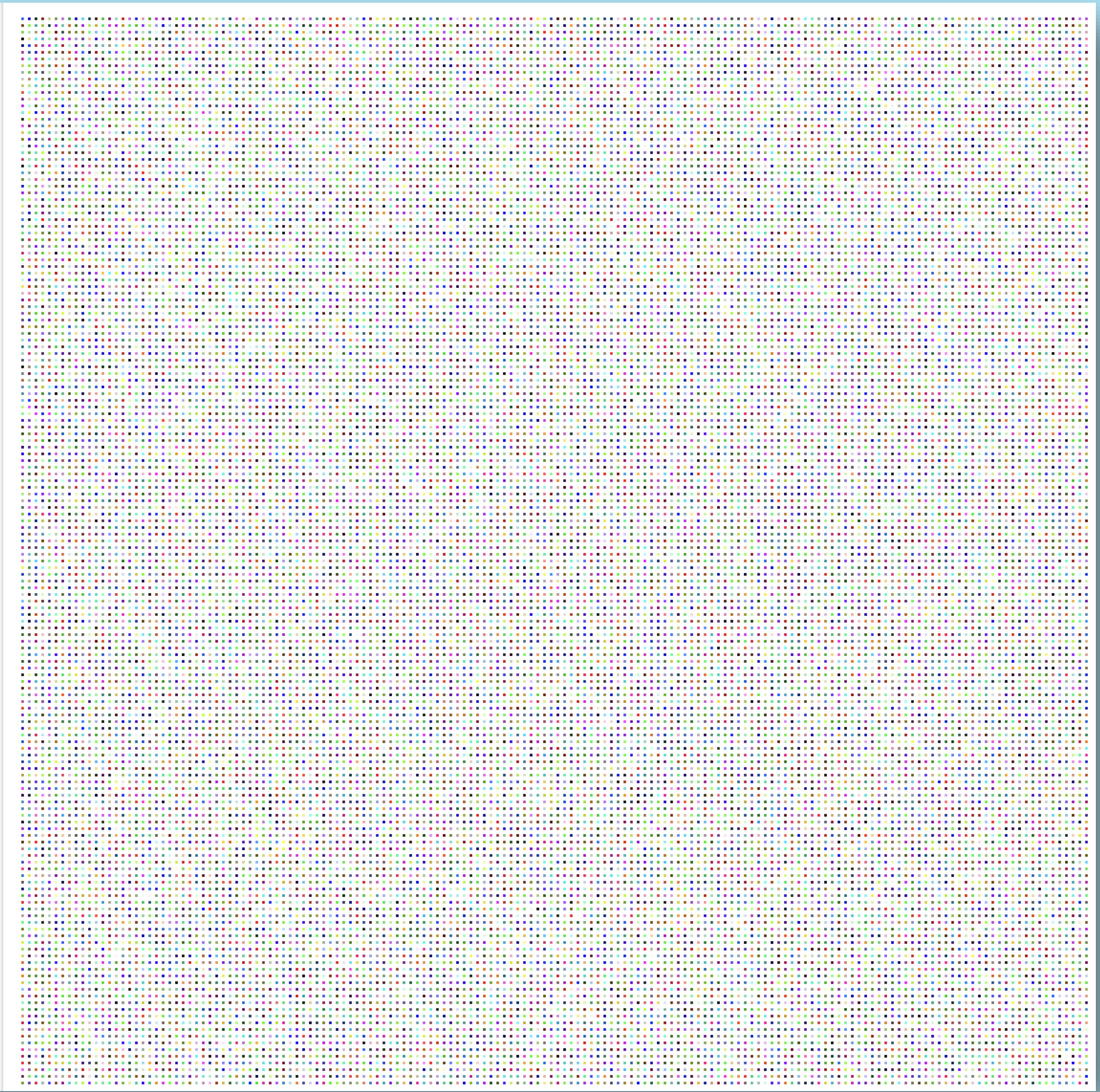


By changing the *size* variable, we can easily increase or decrease the number of squares.

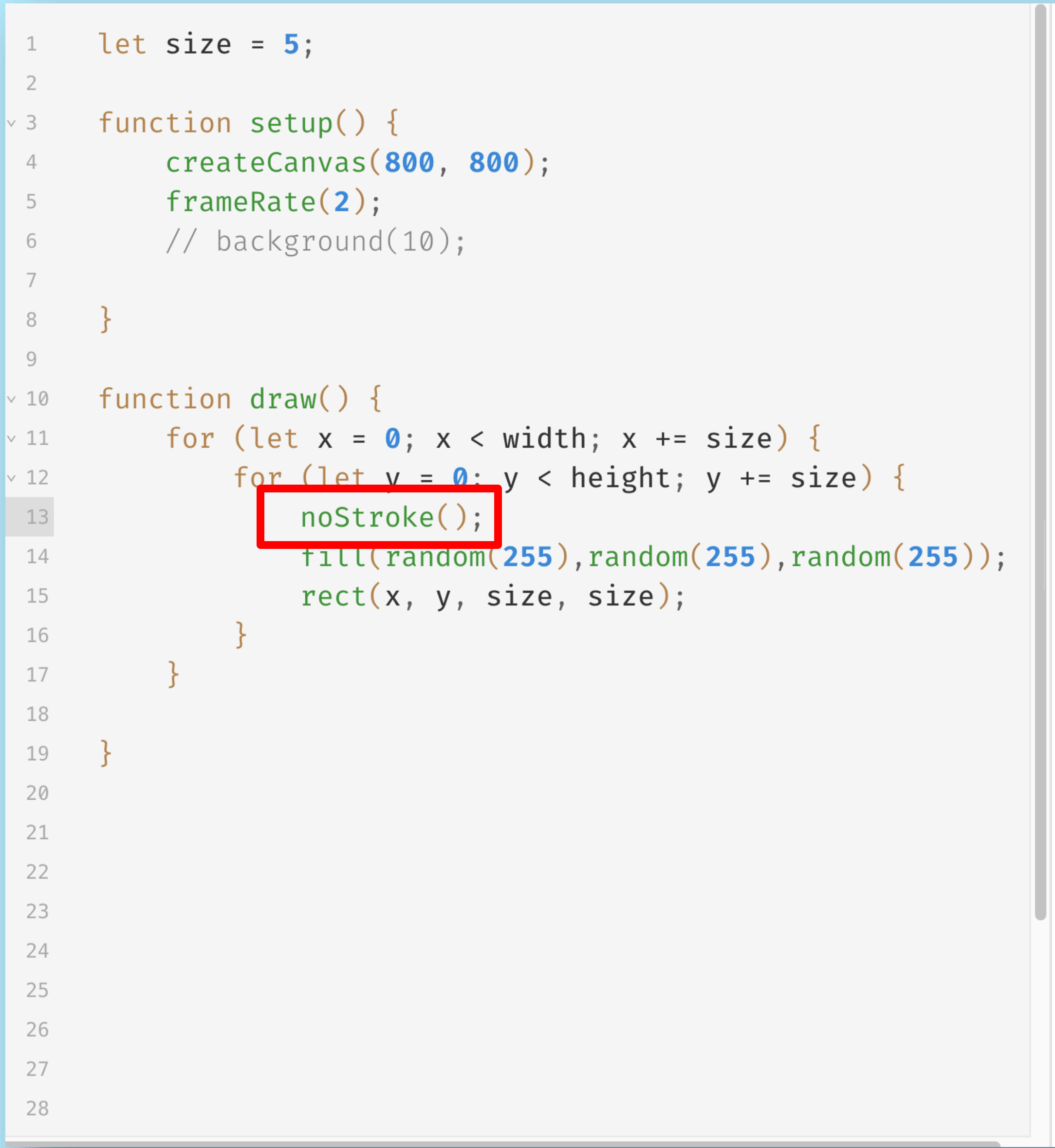
And even smaller



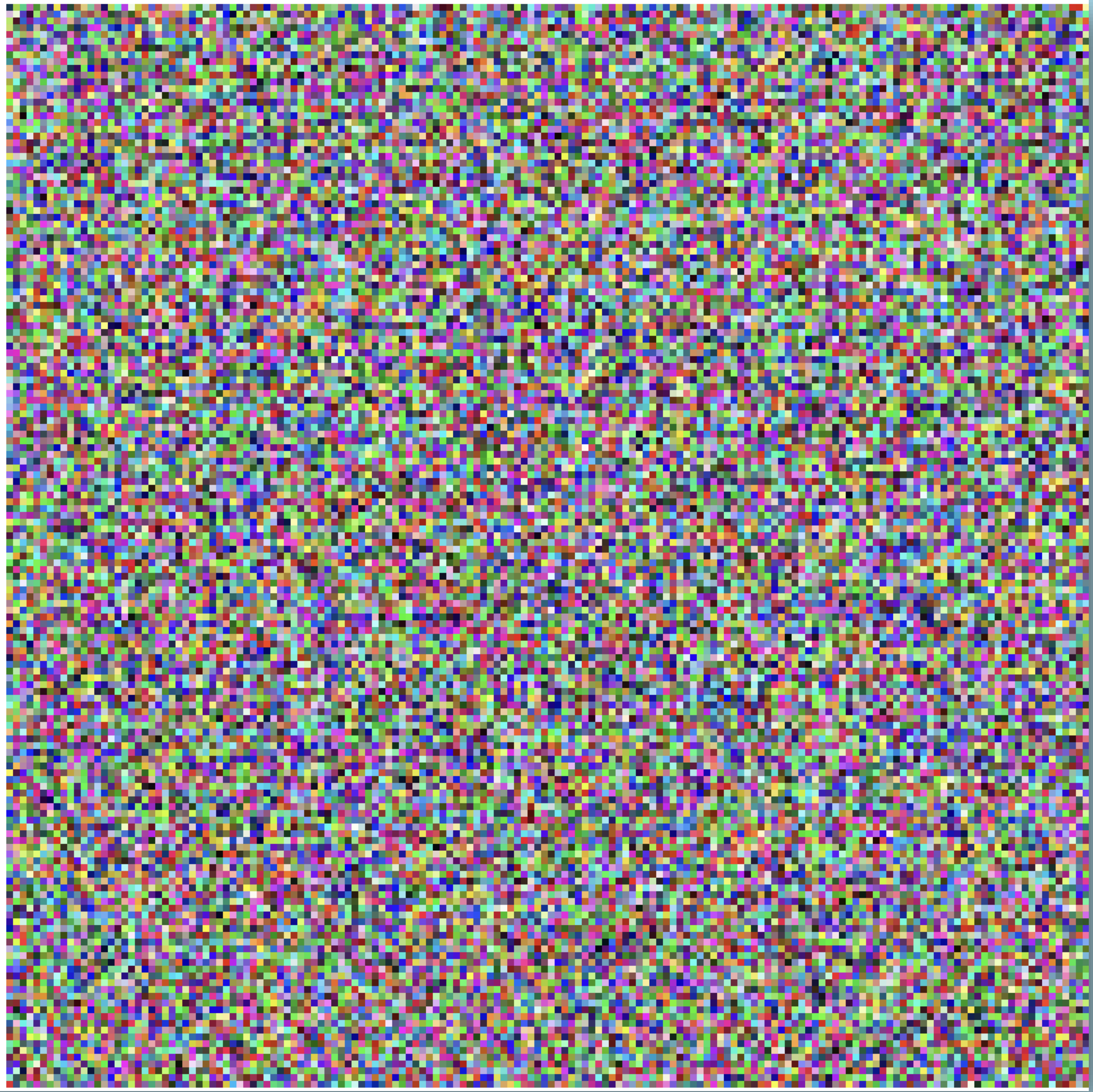
```
let size = 5;

function setup() {
    createCanvas(800, 800);
    frameRate(2);
    // background(10);

}

function draw() {
    for (let x = 0; x < width; x += size) {
        for (let y = 0; y < height; y += size) {
            stroke(255);
            strokeWeight(3);
            fill(random(255),random(255),random(255));
            rect(x, y, size, size);
        }
    }

}
```



Create variations



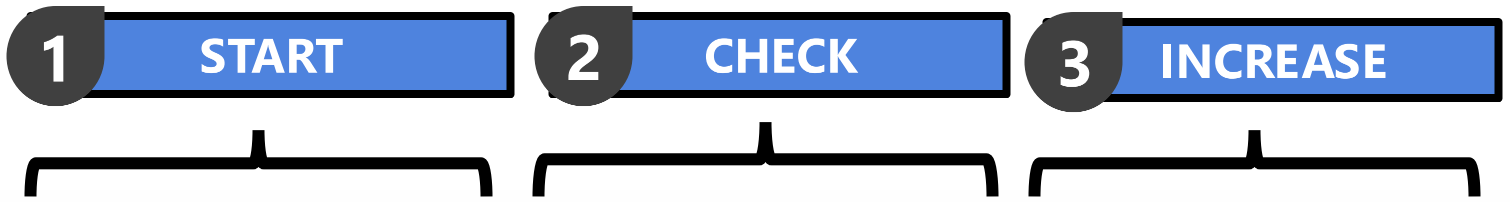
```
let size = 5;

function setup() {
    createCanvas(800, 800);
    frameRate(2);
    // background(10);

}

function draw() {
    for (let x = 0; x < width; x += size) {
        for (let y = 0; y < height; y += size) {
            noStroke();
            fill(random(255),random(255),random(255));
            rect(x, y, size, size);
        }
    }

}
```



Review

Review For Loop

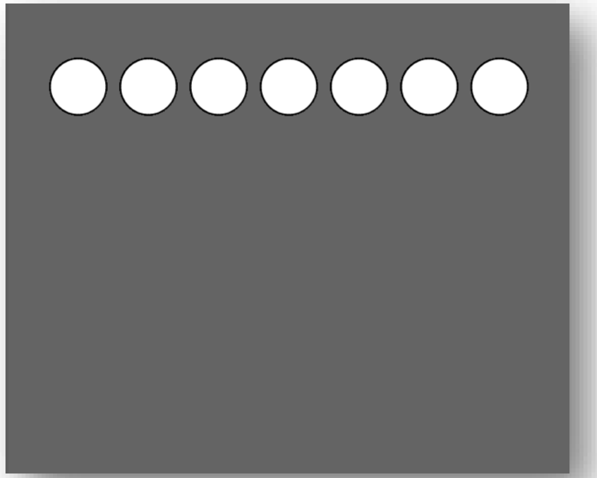
A **for loop** lets you **repeat a pattern** without writing the same line of code over and over again.

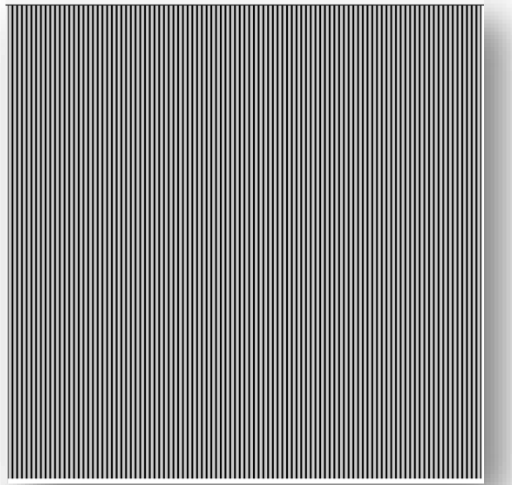


```
for (let x = 50; x < width; x = x + 50) {
  circle(x, 60, 40);
}
```

Review Loop Structure

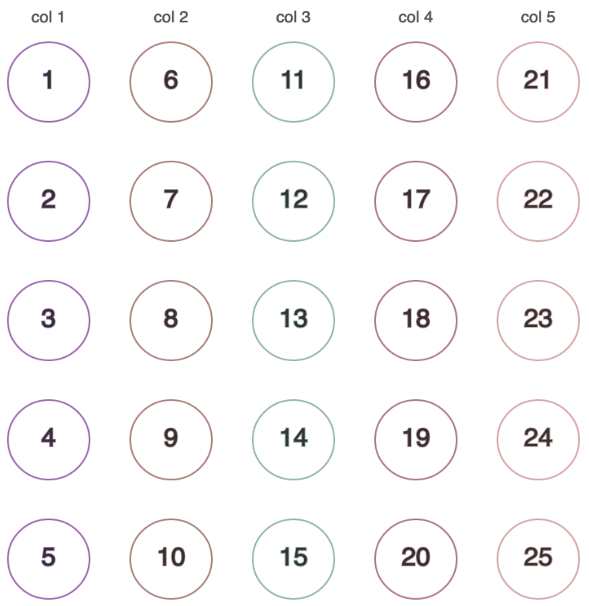
You should use a for loop when you have code that **uses a pattern** that **starts** at a number, **increases** by a number, and **stops** at a number.

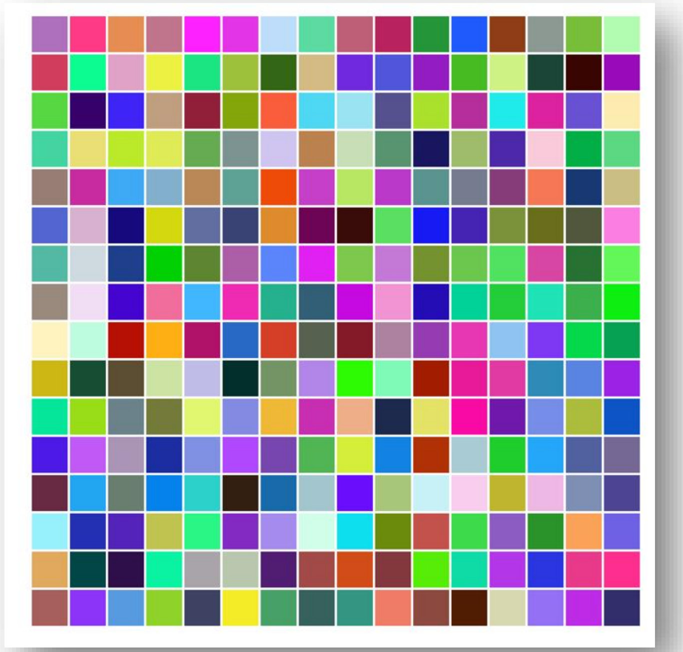




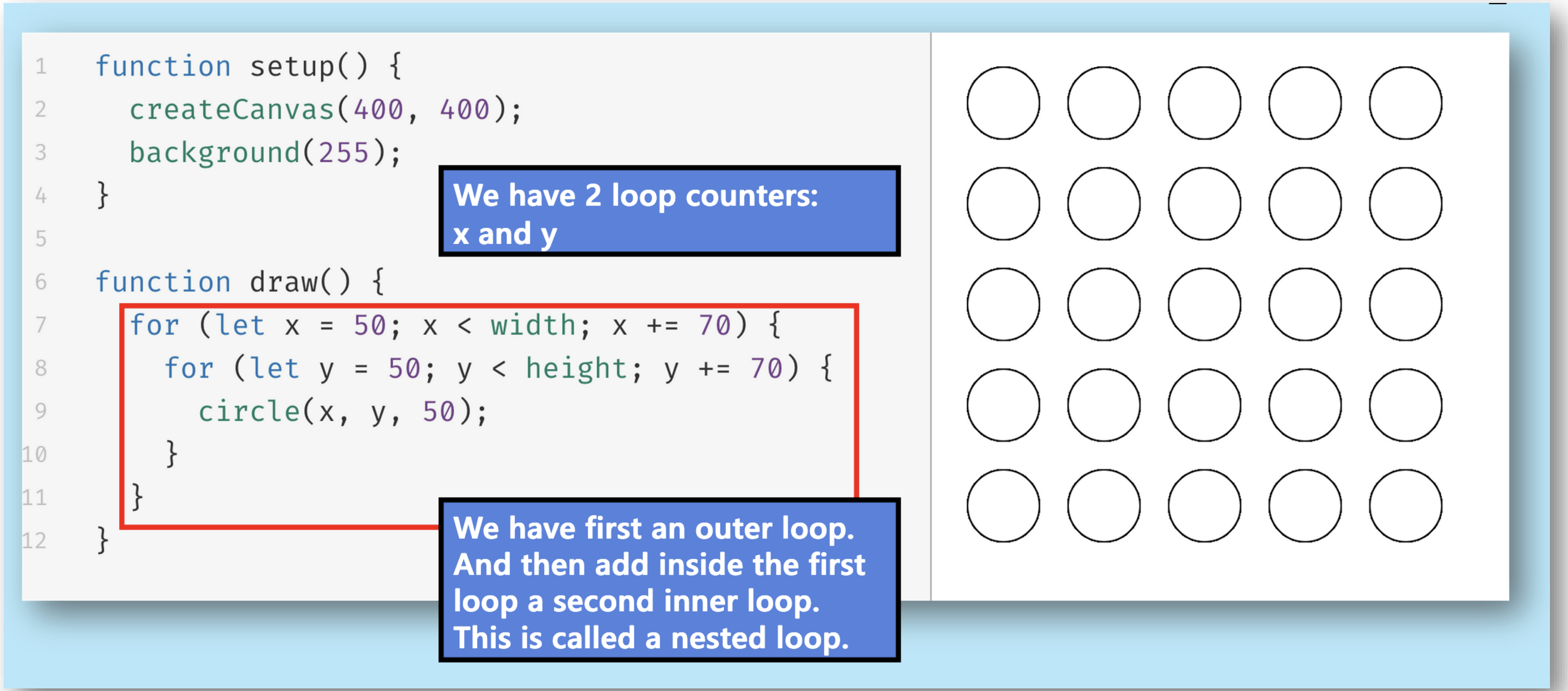
Review Nested Loop

A for loop inside of another for loop is called a **nested for loop**. These are useful when your pattern involves more than one number or if you're working with grids.





Review Nested Loop



```
function setup() {
  createCanvas(400, 400);
  background(255);
}

function draw() {
  for (let x = 50; x < width; x += 70) {
    for (let y = 50; y < height; y += 70) {
      circle(x, y, 50);
    }
  }
}
```

We have 2 loop counters:
x and y

We have first an outer loop.
And then add inside the first loop a second inner loop.
This is called a nested loop.

Luxembourg Tech School

innovative initiatives digital luxembourg.

SCRIPT

cgié centre de gestion informatique de l'éducation

andré losch FONDATION

Luxembourg National Research Fund

LUXEMBOURG SPACE AGENCY

STATE STREET

funded by œuvre nationale

Fondation ABBL pour l'éducation financière

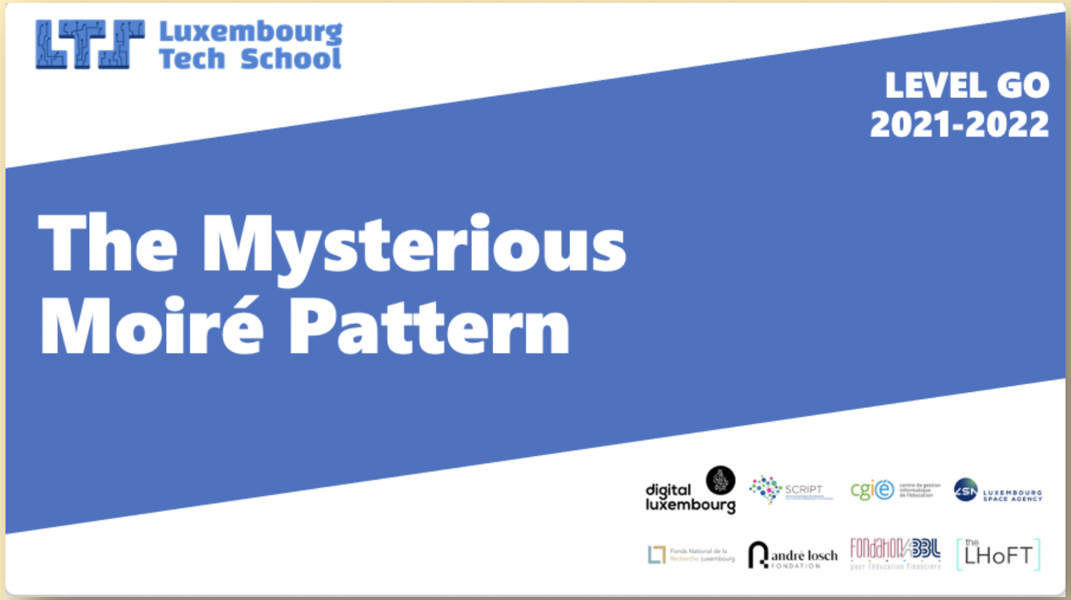
Luxair

the LHoFT

Luxembourg-City Incubator

Extra Activity: Moiré Pattern

You can learn more about **“The mysterious Moiré Pattern“** and how to create one yourself in the slides shared with you on Teams.



Luxembourg Tech School

LEVEL GO
2021-2022

The Mysterious Moiré Pattern