for

A way to repeat a block of code when the number of iterations is known.

`for` loops are helpful for repeating statements a certain number of times. For example, a `for` loop makes it easy to express the idea "draw five lines" like so:

```
for (let x = 10; x < 100; x += 20) {
  line(x, 25, x, 75);
}
```

The loop's header begins with the keyword `for`. Loops generally count up or count down as they repeat, or iterate. The statements in parentheses `let x = 10; x < 100; x += 20` tell the loop how it should repeat:

- `let x = 10` tells the loop to start counting at `10` and keep track of iterations using the variable `x`.
- `x < 100` tells the loop to count up to, but not including, `100`.
- `x += 20` tells the loop to count up by `20` at the end of each iteration.

The code between the curly braces `{}` is the loop's body. Statements in the loop body are repeated during each iteration of the loop.

It's common to create infinite loops accidentally. When this happens, sketches may become unresponsive and the web browser may display a warning. For example, the following loop never stops iterating because it doesn't count up:

```
for (let x = 10; x < 100; x = 20) {
  line(x, 25, x, 75);
}
```

The statement `x = 20` keeps the variable `x` stuck at `20`, which is always less than `100`.

`for` loops can also count down:

```
for (let d = 100; d > 0; d -= 10) {
  circle(50, 50, d);
}
```

`for` loops can also contain other loops. The following nested loop draws a grid of points:

```
// Loop from left to right.
for (let x = 10; x < 100; x += 10) {

  // Loop from top to bottom.
  for (let y = 10; y < 100; y += 10) {
    point(x, y);
  }

}
```

`for` loops are also helpful for iterating through the elements of an array. For example, it's common to iterate through an array that contains information about where or what to draw:

```
// Create an array of x-coordinates.
let xCoordinates = [20, 40, 60];

for (let i = 0; i < xCoordinates.length; i += 1) {
  // Update the element.
  xCoordinates[i] += random(-1, 1);

  // Draw a circle.
  circle(xCoordinates[i], 50, 20);
}
```

If the array's values aren't modified, the `for...of` statement can simplify the code. They're similar to `for` loops in Python and `for-each` loops in C++ and Java. The following loops have the same effect:

```
// Draw circles with a for loop.
let xCoordinates = [20, 40, 60];

for (let i = 0; i < xCoordinates.length; i += 1) {
  circle(xCoordinates[i], 50, 20);
}
```

```
// Draw circles with a for...of statement.
let xCoordinates = [20, 40, 60];

for (let x of xCoordinates) {
  circle(x, 50, 20);
}
```

In the code snippets above, the variables `i` and `x` have different roles.

In the first snippet, `i` counts from `0` up to `2`, which is one less than `xCoordinates.length`. `i` is used to access the element in `xCoordinates` at index `i`.

In the second code snippet, `x` isn't keeping track of the loop's progress or an index. During each iteration, `x` contains the next element of `xCoordinates`. `x` starts from the beginning of `xCoordinates` (`20`) and updates its value to `40` and then `60` during the next iterations.

Examples

```
function setup() {
  createCanvas(100, 100);

  describe('Five black vertical lines on a gray background.');
}

function draw() {
  background(200);

  // Draw vertical lines using a loop.
  for (let x = 10; x < 100; x += 20) {
    line(x, 25, x, 75);
  }
}
```

```
function setup() {
  createCanvas(100, 100);

  describe('Five white concentric circles drawn on a gray background.');
}

function draw() {
  background(200);

  // Draw concentric circles using a loop.
  for (let d = 100; d > 0; d -= 20) {
    circle(50, 50, d);
  }
}
```

```
function setup() {
  createCanvas(100, 100);

  describe('A grid of black dots on a gray background.');
}

function draw() {
  background(200);

  // Set the spacing for points on the grid.
  let space = 10;

  // Increase the stroke weight.
  strokeWeight(3);

  // Loop from the left to the right.
  for (let x = space; x < 100; x += space) {
    // Loop from the top to the bottom.
    for (let y = space; y < 100; y += space) {
      point(x, y);
    }
  }
}
```

```
// Declare the variable xCoordinates and assign it an array of
numbers.
let xCoordinates = [20, 40, 60, 80];

function setup() {
  createCanvas(100, 100);

  describe('Four white circles drawn in a horizontal line on a gray background.');
}

function draw() {
  background(200);

  // Access the element at index i and use it to draw a circle.
  for (let i = 0; i < xCoordinates.length; i += 1) {
    circle(xCoordinates[i], 50, 20);
  }
}
```

```
// Declare the variable xCoordinates and assign it an array of
numbers.
let xCoordinates = [20, 40, 60, 80];

function setup() {
  createCanvas(100, 100);

  describe('Four white circles drawn in a horizontal line on a gray background.');
}

function draw() {
  background(200);

  // Access each element of the array and use it to draw a
circle.
  for (let x of xCoordinates) {
    circle(x, 50, 20);
  }
}
```

This page is generated from the comments in src/core/reference.js. Please feel free to edit it and submit a pull request!

Related References

class
A template for creating objects of a particular type.

console
Prints a message to the web browser's console.

for
A way to repeat a block of code when the number of iterations is known.

function
A named group of statements.