This is a review of what we covered in this tutorial on loops.

When we're writing programs, we often find that we want to repeat a bit of code over and over, or repeat it but change something about it each time. To save ourselves from writing all that code, we can use a ***loop***. JavaScript has two kinds of loops, a ***while loop*** and a ***for loop***.

A ***while loop*** is a way to repeat code until some condition is false. For example, this while loop will display the value of y at (30, y) as long as y is less than 400. The loop adds 20 to y each time it runs, so that y starts off at 40 but then increments to 60, 80, 100, 120, etc.

```
var y = 40;
while (y < 400) {
  text(y, 30, y);
  y += 20;
}
```

It's important that the condition inside the parenthesis becomes false at some point - otherwise we'll have what's known as an ***infinite loop***! That's what would happen if we removed `y += 20`, because y would be 40 forever, and always be less than 400, and the program would never know when to stop.

```
var y = 40;
while (y < 400) {
  text(y, 30, y);
}
```

The ***for loop*** is similar to a while loop, but with a more specialized syntax. Programmers invented the for loop when they realized they were always doing the same three things- creating loop counter variables (like y above), incrementing them by some amount, and checking that they're less than a value. A for loop syntax has special places for each of those three things. Here's the same loop as above, as a for loop:

```
for (var y = 40; y < 400; y += 20) {
  text(y, 30, y);
}
```

Loops can also be ***nested***. It's actually very common to nest for loops, especially in 2-d drawings, because it makes it easy to draw grid-like shapes. When we nest a loop inside a loop, we're telling the program to "do this thing X many times, and for each time you do

that, also do this other thing Y many times." Think about drawing a grid- we'd want to tell the program to "create a column 10 times, and for each column, also create 15 cells inside of it." Here's how you can use nested for loops to achieve that:

```
for (var col = 0; col < 10; col++) {
  for (var row = 0; row < 15; row++) {
    rect(col*20, row*20, 20, 20);
  }
}
```

When should you use a for loop versus a while loop? That's up to you. Many programmers prefer for loops because it's harder to accidentally create an infinite loop (because it's harder to forget to increment your counter variable), but sometimes a while loop might make more sense. Try them both!