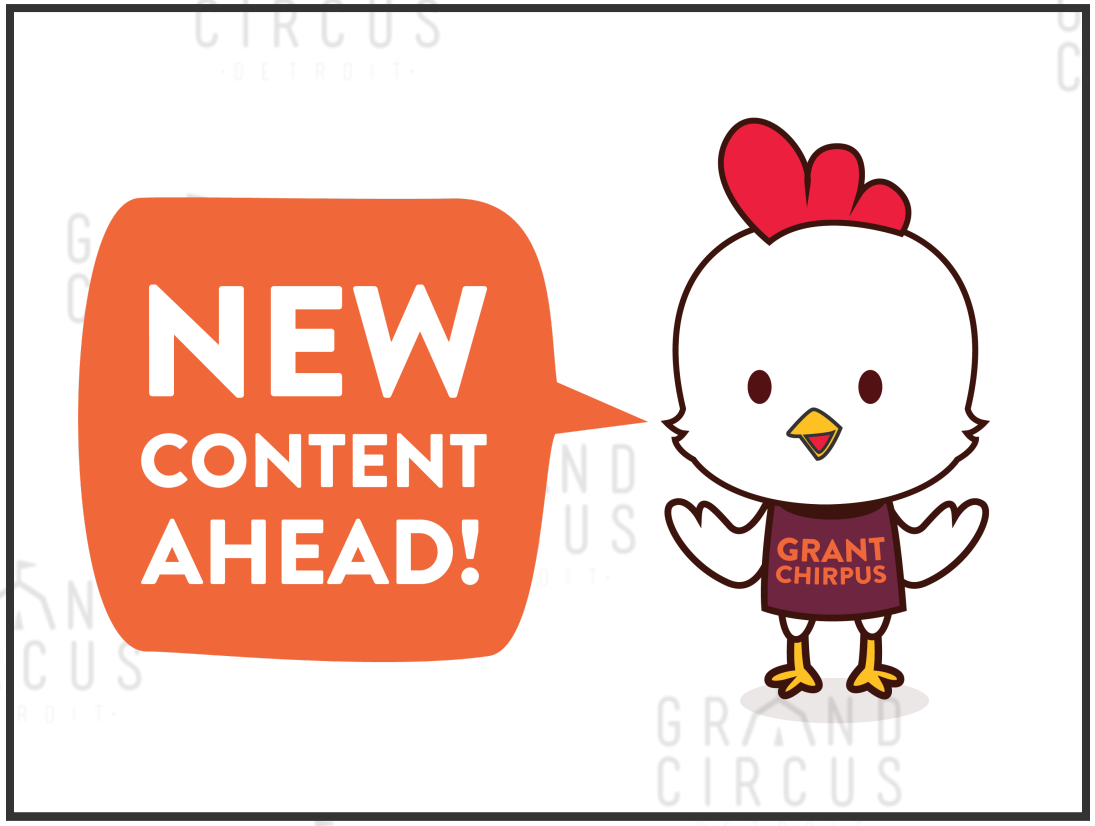
NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

GOALS FOR THIS UNIT

1. Review
2. Loops
3. Intro to Functions

REVIEW

LOOPS

LOOPS

Loops allow us to iterate (execute multiple times) over sections of code based on a condition we set. From this point on, when I say "iteration", I mean executing the code inside of a loop one time.

There are a few different types of loops. The loop you use will depend on the task you're trying to accomplish.

```
for(var i=0; i < n; i++) {
  // repeating things
}

while(true) {
  // repeating things
}

do {
  // repeating things
} while(true);
```

LOOPS

In practice, I usually use for loops or a more specialized iterator like `forEach()` (more on that later).

FOR LOOP

For loops have three optional expressions, enclosed in parentheses and separated by semicolons, followed by a statement or a set of statements executed in the loop.

```
for ([initialization]; [condition]; [final-expression]) {
   // repeating things
}
```

It is common to use a counter to determine the number of times a program should run through a for loop. We use `i` as this counter's variable name by convention.

INITIALIZATION

The first expression assigns a value to a variable, usually a counter.

```
for (i = 0; [condition]; [final-expression]) {
   // repeating things
}
```

CONDITION

The second expression is evaluated before the loop is run. If it evaluates to true, we run the statement(s) inside. If it is false, we break out of the loop and continue executing the program.

```
for ([initialization]; i < n; [final-expression]) {
   // repeating things
}
```

FINAL EXPRESSION

The third and final expression in parentheses is run at the end of each iteration of the loop. It is primarily used to update the counter.

```
for ([initialization]; [condition]; i++) {
   // repeating things
}
```

QUICK EXERCISE

In jsbin.com, build a `for` loop that counts from 1 to 10. Print each number to the console. (10 mins.)

WHILE LOOP

A while loop iterates as long as a given condition evaluates to true. The condition is evaluated before each iteration of the loop.

```
while (condition) {
  // repeating things
}
```

The block inside the while loop must provide a way to exit the loop. Otherwise, we get caught in an infinite loop that will continue to run over and over again.

```
var a = 0;
var j = 0;

while (a < 30) {
  a++;
  j += a;
}
```

QUICK EXERCISE

In jsbin.com, build a while loop that produces the following output. (10 min)

For fun, remove whatever counter you used to stop the loop to force an infinite loop.

```
You are 10 years old!
You are 11 years old!
You are 12 years old!
You are 13 years old!
You are 14 years old!
You are 15 years old!
You are 16 years old!
You are 17 years old!
You are legally an adult!
```

DO...WHILE LOOP

A do...while loop is very similar to a while loop, but code contained within the loop is always run at least once because the condition is evaluated at the end of each iteration, rather than the beginning.

```
do
 // repeating things
while (condition);
```

WHAT IS IT GOOD FOR?

Suppose we want to ask the user a question over and over until we get an acceptable answer. We'd end up having to repeat code using a while loop. Using do...while instead allows us to cut down on the code we write.

```
var faveFruit = prompt("What's your favorite fruit?");

while (faveFruit !== "apples") {
  faveFruit = prompt("What's your favorite fruit?");
}
```

```
var faveFruit;

do
  faveFruit = prompt("What's your favorite fruit?");
while (faveFruit !== "apples");
```

FOR EACH LOOP

A loop that iterates over every element in an array.

```
myArray.forEach(function(element) {
  // repeating things
});
```

name below is a variable that holds each array element one by one. You can pick any name you want for this variable.

```
var array = [ "Hasan", "Bert", "Jose" ];
array.forEach(function(name) {
    console.log(name);
});
```

QUICK EXERCISE

In jsbin.com, build a forEach loop that adds together all the numbers in this array. (10 min)

Hint: As a first step, just get it to print all the numbers. Then change it to add them.

```
var numbers = [ 1, 1, 2, 3, 5, 8 ];
// The answer should be 20.
```