

# Computer Science & Engineering 120

## Learning to Code

### Processing Data I – Loops

Christopher M. Bourke  
cbourke@cse.unl.edu

## Topic Overview

- ▶ Introduction to loops
- ▶ Using a for loop
- ▶ Iterating over arrays
- ▶ While Loops
- ▶ For each loops

## Introduction

- ▶ Need a way to *repeatedly* execute a block of code
- ▶ Apply an operation to each element in an array (sum numbers, print them, insert them into a table, etc.)
- ▶ Repeat an operation until some condition is satisfied
- ▶ Animation: fade in an element by changing its opacity until it is fully opaque

## Loops

- ▶ A *loop* allows us to *repeatedly* execute a block of code until some *condition* is no longer satisfied
- ▶ Once the condition is no longer satisfied, the loop *terminates* its execution

A loop has three main components:

1. An initialization statement – a statement that indicates how the loop *begins*
2. A continuation condition – a logical statement (true or false) that specifies whether or not the loop should continue executing
3. An iteration statement – a statement that makes progress toward the termination of the loop (otherwise, it would continue to execute forever!)

## Example

Printout numbers 1 through 10:

1. Initialize a variable  $i$  to 1
2. While the variable  $i$ 's value is less than or equal to 10...
3. Print  $i$
4. Increment  $i$  by adding 1 to it
5. Go to step 2

## Example

```
1 for(var i=1; i<=10; i++) {  
2   console.log(i);  
3 }
```

- Key Annotations:**
- ▶ Initialization statement: `i=1` – a statement that indicates how the loop *begins*
  - ▶ Usage of the keyword `for`
  - ▶ A continuation condition: `i<=10` – a logical statement (true or false) that specifies whether or not the loop should continue executing
  - ▶ The initialization statement and continuation statement end with semicolons `;`
  - ▶ An iteration statement: `i++` – a statement that makes progress toward the termination of the loop (otherwise, it would continue to execute forever!)
  - ▶ Usage of *punctuation*, parentheses and curly brackets

## Iteration Operators

Note the *iteration operators*:

- ▶ `i++` adds 1 to the variable
- ▶ `i--` subtracts 1 from the variable
- ▶ `i += 5` adds 5 to the variable
- ▶ `i -= 5` subtracts 5 from the variable

## Iterating Over Arrays

- ▶ Recall that array elements are indexed starting at 0
- ▶ Length of an array can be found using `arr.length`
- ▶ Last element is at index `arr.length - 1`
- ▶ A for loop can be used to iterate over array elements

```
1 var myNumbers = [2, 3, 10, 5, 2, 19, 12];
2 var sum = 0;
3 for(var i=0; i<myNumbers.length; i++) {
4     sum = sum + myNumbers[i];
5     //or sum += myNumbers[i];
6 }
7 console.log("Total: " + sum);
```

## Advanced Usage

- ▶ Various array functions can be used to iterate over elements
- ▶ More details: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/Reduce](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce)

```
1 var myNumbers = [2, 3, 10, 5, 2, 19, 12];
2
3 var sum = myNumbers.reduce(
4     function(pVal, cVal, index, arr) {
5         return pVal + cVal;
6     }, 0);
7
8 console.log("Total: " + sum);
```

## While Loop I

- ▶ An alternative is a while loop
- ▶ Same three parts, but located differently
- ▶ Initialization statement appears *before* the loop
- ▶ Keyword `while` is used to specify the continuation condition
- ▶ Iteration statement is *within* the loop (usually at the end)

## While Loop II

```
1 var i=1; //initialization
2 while(i<=10) { //continuation condition
3     console.log(i);
4     i++; //iteration
5 }
```

## While vs For

- ▶ Any while loop can be rewritten as a for loop and vice versa
- ▶ Only difference is semantics/syntax
- ▶ Usually use a while loop when you don't know how many iterations are needed
- ▶ Example: *while* we have not reached the end of an input (we may not know how big the input is prior to processing)
- ▶ Example: *for* each element in the array (we know how many there are)

## jQuery's each() Function I

- ▶ jQuery provides a function, `each()` function that can be applied to a selector result set or an array
- ▶ Replaces boilerplate loop code with a *callback*
- ▶ You provide `each()` with another *function*
- ▶ `each()` then loops for you: passing each element in the array to your function for processing

## jQuery's each() Function II

```
1 var myNumbers = [2, 3, 10, 5, 2, 19, 12];
2 var sum = 0;
3
4 $.each(myNumbers, function(index, value) {
5     sum += v;
6 });
7
8 console.log("Total: " + sum);
```

## jQuery's each() Function III

- ▶ Alternatively, you can apply the `each()` function to a selector result set
- ▶ First argument (array) is omitted

```
1 $("p").each(function(index, element) {
2     console.log(element.innerHTML);
3 });
```

## jQuery's each() Function IV

- ▶ Note: the `element` is a normal DOM element, not a jQuery object
- ▶ `element` has no `text()` function
- ▶ Trick: you can *change* it to a jQuery object by wrapping it in a selector call:  
`$(element)`

```
1 $("p").each(function(index, element) {
2     console.log($(element).text());
3 });
```

## Vanilla forEach() Function

- ▶ As of ES5, JavaScript has a `forEach()` array function
- ▶ Some prefer to use "Vanilla" (plain) JavaScript rather than jQuery

```
1 var myNumbers = [2, 3, 10, 5, 2, 19, 12];
2 var sum = 0;
3
4 myNumbers.forEach(function(value, index, array) {
5     console.log(value + " is at index " + index);
6 });
```

## Part II: Demonstrations & Exercises

## Visualization Demonstration

- ▶ Understanding a loop better by tracking its execution
- ▶ JavaScript execution visualization tool:  
<http://int3.github.io/metajs/>

## Exercise 1

**Exercise:** Given an array of numbers, find the *minimal* element

## Exercise 2

**Exercise:** Given an array, compute the average of its elements. Then, iterate over the elements and insert them into a table, indicating if the value is above, below or equal to the average.

## Exercise 3

**Exercise:** Process the enrollment data from a previous module. First, find all (unique) course records. Then, go over the enrollment records and count the number of students in each course. Produce a table that summarizes the data.