CSCE 120: Learning To Code

Organizing Code I
Hacktivity 9.1

Introduction

Prior to engaging in this hacktivity, you should have completed all of the pre-class activities as outlined in this module. At the start of class, you will be randomly assigned a partner to work with on the entirety of this hacktivity as a *peer programming* activity. Your instructor will inform you of your partner for this class.

One of you will be the driver and the other will take on the navigator role. Recall that a driver is in charge of the keyboard and computer while the navigator is in charge of the handout and directing the activity. However, you are *both* responsible for contributing and discussing solutions. If you were a driver/navigator in the prior activity, switch roles for this activity.

1 Knowledge Check

With your partner, discuss and answer each of the following questions. Write your answers down on a separate sheet of paper or type them up in a plain text file.

1. What is a function signature? What are the three parts of a function's signature?
2. Consider the following piece of code:

```
var x = 10;
var y = 20;
var z;
function compute(a, b) {
  return (a + b) / 2;
}
z = compute(x, y);
x = compute(y, z);
```

What are the values of the variables `x`, `y`, and `z` after this code executes?

3. Consider the following piece of code:

```
var x = 10;
var y;
function add(a) {
  a = a + 5;
}
y = add(x);
```

   What are the values of the variables `x`, `y` after this code executes?

4. Consider the following piece of code:

```
function compute(a, b, c) {
  if(c === undefined) {
    c = 25;
  }
  return (a + b + c) / 3;
}
```

   What value is returned on the following function calls?

   a) `compute(10, 20, 30);`,

   b) `compute(10, 20);`,

   c) `compute(10);`,

5. What syntax would you use to make the `add` and `compute` functions part of an object named `MyFunctions`?

6. What is a callback? How and why would you use one?

7. What is an anonymous function and why would you use one?

2 Warm-up Exercises

Download the code we've provided from GitHub using the URL, https://github.com/cbourke/FunctionExercises. Open the project in Light Table and open the file, `exercises.js`. Complete the following exercises.

1. Write a function to convert from Celsius to Fahrenheit. Then write a function to convert from Fahrenheit to Celsius. Test your function with various values. The formulas for these conversions are as follows.

$$c = \frac{9}{5}(f - 32)$$

$$f = c \cdot \frac{5}{9} + 32$$

2. Rewrite the two temperature functions to be members of an object named `TemperatureUtils`

3. Recall that the natural logarithm, $\ln(x)$ is the logarithm with base $e = 2.71828\ldots$. You can use the natural logarithm to compute the logarithm using any base $b > 0$ using the formula

   $$\frac{\ln(x)}{\ln(b)}$$

   Write the following function that takes two arguments where the second argument is the base. However, write your function so that b is optional: if it is not provided, it should use the default base e:

   ```
   function logarithm(x, b) {
   }
   ```

   Recall that the math library provides a function to compute the natural logarithm: `Math.log()`. Test your functions with the following values:

   - $x = 64, b = 2$ should be 6
   - $x = 64, b = e$ should be 4.15888
   - $x = 64, b = 10$ should be 1.806179

4. Write a function that takes an array and returns the sum of its elements. Be sure to rigorously test your function.

5. Write a function that takes an array and computes the mean of its elements (hint: how can you reuse your sum function?)

6. An investment may pay you dividends, but over time inflation also tends to reduce your returns. To compute your inflation-adjusted returns, you can use the formula:

   $$\left(\frac{1 + \text{investment return}}{1 + \text{inflation rate}} - 1\right) \cdot 100$$

   Write a function that takes two parameters and computes the inflation-adjusted return rate.

7. Write a function that takes two numbers: a, b and compares their values. Design the function to return -1 if $a < b$; 0 if they have the same value and 1 if $a > b$. This is known as a *comparator function*. Name your function `cmpNumeric`

8. Write a similar comparator function that takes two student objects a, b and orders them by last name/first name. It should return:

   - -1 if a's name comes before b's name
   - 0 if they have the same names
   - 1 if b's name comes before a's name

Be sure to handle the situation where they share the same last name, but different first names. Test your function with several examples using the following formatting. Name your function `cmpPerson`

```
var john = {
    firstName: "John",
    lastName: "Student"
};
var jane = {
    firstName: "Jane",
    lastName: "Doe"
};
```

9. Write a function that takes an array of numbers as an argument and returns the maximal element in it.

10. Now suppose that we wanted to write a function to find the maximum value in an array of strings. We'd have to write another function. Now suppose we wanted to find the maximum value among an array of objects representing students: we would have to write dozens of functions for each criteria (the "maximum" with respect to name, GPA, class rank, etc.).

    A better solution would be to write a general purpose `getMax()` that could work with array of *any* type of data (not just numbers). To do this, we'll provide the `getMax()` function with a comparator *callback* function that is responsible for ordering pairs of data by returning something negative, zero, or something positive depending on the relative ordering of two elements.

    Consider the following (incomplete) solution.

```
function getMax(arr, cmp) {
  var max = arr[0];
  for(var i=1; i<arr.length; i++) {
    if(cmp(max, arr[i]) ______) {
      max = arr[i]
    }
  }
  return max;
}
```

    Discuss what each line in this function does. Then determine what code is missing and implement this function. Fully test your function with the provided data and the comparator functions that you previously implemented.