CSCE 120: Learning To Code

Making Decisions
Hacktivity 4.1

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, writing your answers down on a separate sheet of paper.

1. Consider the following snippet of code

```
var a = 10;
var b = 20;
var c = 30;
var d = "zebra";
```

   For each of the following expressions, indicate if it evaluates to true or false.

   a) `(a <= b)`

   b) `(a < b)`

c) `(a === b)`

d) `(a !== c)`

e) `(a*b > c)`

f) `(a+b === c)`

g) `(a+b < c)`

h) `(b < 20)`

i) `("alpha" === d)`

j) `(d > "apple")`

k) `(d <= "zebra")`

l) `(d !== "zebra")`

2. Consider the following snippet of code

```
var a = 10;
var b = 20;
var c = true;
```

For each of the following expressions, indicate if it evaluates to true or false.

a) `(a < 10 || b === 30)`

b) `(a <= 10 || b === 30)`

c) `(a <= 10 && b !== 30)`

d) `(a === 10 && !c)`

e) `(a <= 10 && (b > 30 || c))`

f) `(a <= 10 && b > 30 || c)`

g) `(a > 10 || b > 30 || b < 50)`

h) `(a === 10 && b <= 30 && c)`

3. For each of the following code snippets, predict what the code snippet will print to the console. Verify your work by executing the code.

a)

```
var a = 10;
if(a !== 10) {
  console.log("foo");
} else {
  console.log("bar");
}
```

b)

```
var a = 10;
if(a < 10) {
  console.log("foo");
} else if(a > 10) {
  console.log("bar");
} else {
  console.log("baz");
}
```

c)

```
var a = 10;
var b = 20;
if(a < 10) {
  if(b < 20) {
    console.log("foo");
  } else {
    console.log("bar");
  }
} else {
  console.log("baz");
}
```

d)

```
var a = 10;
var b = 20;
if(a-b < 0) {
  console.log("foo");
} else if (a-b >= 0) {
  console.log("bar");
} else if (a-b > 0) {
  console.log("baz");
}
```

2 Getting Some Practice

2.1 Leap Years

About every 4 years we have a *leap year* where an extra day (February 29th) is included in the calendar. The rules are actually more complex since the number of days the Earth rotates around the sun is about 365.2422 and not exactly 365.25. To correct for this, a year is a leap year if it is:

- Divisible by four (so 2012, 2016, etc.), but
- Not divisible by 100 unless it is also divisible by 400

2012, 2016, 2020 are leap years. 1600 and 2000 were leap years, but 1700, 1800, and 1900 were *not* leap years.

1. Declare a variable to hold a value of a year
2. Write code to determine if the value stored in your variable is a leap year; print an appropriate message to the console to indicate if it is or is not a leap year
3. Test your code with the years mentioned above.

2.2 Ranges

Suppose that we want to check for a *range* of values of a variable. For example, suppose we want to execute some code if the value of a variable x satisfies $0 \leq x \leq 10$. Consider the following code that attempts to do this:

```
var x;
if(0 <= x <= 10) {
  console.log("yes");
} else {
  console.log("no");
}
```

1. Type this code out and test it for values of `x = -5;`, `x = 5;`, and `x = 15;`. Indicate what values are printed and whether or not they are "correct".
2. Discuss with your partner possible reasons why the above code does not work.
3. Rewrite the code using a proper logic operator and retest your values

2.3 Ordering Names

In this exercise, you'll use comparison and conditional statements to *order* two objects representing people by name. Take care, if they have the same last name, then your code should examine their first name.

1. Define two people objects as follows:

```
var a = {
  "firstName": "Matt",
  "lastName": "Smith"
};
var b = {
  "firstName": "David",
  "lastName": "Tennant"
};
```

2. Write code to order the two person objects and print out a message to the console such as `"Matt Smith comes before David Tennant"`

3. Test your code by changing the name values with the following combinations: "Starlin Castro", "Kris Bryant"; "John Smith", "Adam Smith"; "Art Jones", "Bart Jonesy"

2.4 Experiments

2.4.1 Defensive Coding

Recall that certain operations are invalid or lead to invalid results.

- Recall that division by zero is invalid. Write code to compute the division of two variables, `c = a / b;` but that *first* checks to see if you are dividing by zero. If you are, print an error message to the console. If you are not, compute the value and print it. Test your code with at least 3 values for `a` and `b`.
- Write similar code for math operations, `Math.sqrt(a)` and `Math.log(a)` and prevent invalid operations with bad inputs.

Making checks like this before invalid operations is known as *defensive coding*.

2.4.2 Useless Code

Let's test what happens when we create certain logical statements

- Create a variable `var x`.
- Write an if statement using the condition `(x || !x)` that prints out one message if it is true and a different message if it is false.
- Set `x` to `true` and print the results. Reset `x` to `false` and print the results, what do you observe?
- Replace the condition with `(x && !x)`; what do you observe now?
- Note: a logical statement that is *always* true is called a *tautology*, while a logical statement that is always false is a *contradiction*. Identify which of the above statements is a tautology and which is a contradiction. Discuss with your partner: would it ever make sense to use these kind of statements in a program?

2.4.3 De Morgan's Laws

Let's test another type of statement.

1. Write the following piece of code:

```
var x = true;
var y = true;
var a = !(x || y);
var b = (!x && !y);
console.log("a is " + a + ", and b is " + b);
```

2. Evaluate your code for all possible combinations (there are four) of values for `x` and `y`, what do you observe?

3. Write the following piece of code:

```
var x = true;
var y = true;
var a = !(x && y);
var b = (!x || !y);
console.log("a is " + a + ", and b is " + b);
```

   and repeat your experiment with all possible combinations of `x` and `y`, what do you observe?

These two equivalences are known as *De Morgan's Laws*.

3 Finding the Maximum

In this exercise, we'll create a full program.

1. Create a new program that reads in 3 numbers from the user using the `prompt()` function. Be sure to convert the input to numbers using `parseFloat()`.

2. Create another variable, `max` and write code to assign it a value that represents the maximum value among the three that were read in.

3. Write code to display a popup message (using `alert()` to inform the user of what the maximum value is).

4. Discuss with your partner potential *corner-cases* in which your code may fail. Test your program with these values and make sure it works.

5. Multiple solutions are possible. At the end of class, we will look at a few examples and critique them. In software development, this is known as a *code review*: a developer presents her code the team and they look for bugs, flaws, potential problems, etc. and offers suggestions on changes and improvements.