CSCE 120: Learning To Code

Processing Data II
Hacktivity 10.1

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. Name at least 2 functions that enable you to *add* elements to an array. Name at least 2 functions that enable you to *remove* elements from an array.
2. Consider the following array: `var arr = [10, 20, 30, 40];` Answer the following questions:
   a) What does the following line of code do?
   `arr.splice(0, 0, 5, 15);`
   b) What does the following line of code do?
   `arr.splice(0, 1, 5, 15);`

c) What does the following line of code do?
`arr.splice(4, 0, 5, 15);`

d) What does the following line of code do?
`arr.splice(0, 4);`

3. What does the following snippet of code do? What are the contents of `nums` after it executes?

```javascript
var num = [1, 4, 9];
var nums = num.map(function(val, index, arr) {
  return val + 1;
});
```

4. Explain the `reduce()` array function: How do you use it, what does it do?

2 Array Exercises

Download the code we've provided from GitHub using the URL, https://github.com/cbourke/AlbumApp. Open the project in Light Table.

Open the `exercises/arrayExercises.js` file. We have provided an array declaration for you to use. Complete the following exercises.

1. Write code to insert -10 at the beginning of the array
2. Write code to insert -20 to the end of the array
3. Write code to insert 1, 2, and 3 between 15 and 20
4. Write code to insert 4, 5, and 6 at the end of the array
5. Write code to remove the first element
6. Write code to remove the last 4 elements
7. Write code to *replace* the first three elements in the array with 4, 6, 8
8. Write code to remove every element from the array

3 String Exercises

Open the `stringExercises.js` file and complete the following exercises.

1. The `replace` string function requires the use of regular expressions. As an alternative, write the following replace function:

```
/**
 * This function returns a new string which is str
 * but with all instances of the character a replaced
 * with the character b
 */
function replaceAll(str, a, b) {
}
```

Note: `a` and `b` are variables, not actual characters. If you were to call this function as `replaceAll("Hello World!", "o", "y")` it should return `"Helly Wyrld!"`

2. Generalize the previous replace function so that both `a` and `b` can be sequences instead of single characters. For example, calling your function with the arguments `replaceAll("Very berry cherry", "rry", "rries")` should result in the string `"Very berries cherries"`. Hint: iterate over each character in the string and test for instances of `a` using `substr`.

3. Write the following function: `removeChar(str, a)` which returns a new string which is `str` but with all instances of the character (or sequence of characters) `a` removed. Hint: how can you use some of your previously developed functions?

4. Write the following function:

```
function stringToObject(keys, values) {
}
```

that takes an array of strings, `keys` that represent object keys and a string, `values` that contains a comma-delimited list of values that represent object values. The function should construct a new object mapping each key to each value. For example, if you call your function with the following inputs,

`stringToObject(["lastName", "firstName", "id"], "Smith,Joe,1234")`

it should construct and return an object that looks like

```
{
  "lastName": "Smith",
  "firstName": "Joe",
  "id": "1234"
}
```

Hint: you can use the syntax `obj["key"] = value` to set the key-value pair of an object!