07.02 Lab Exercises - SOLUTION

converting map() & filter() to arrow function versions

This is a repeat of Lab 07.01, except this time, use arrow functions. Where possible, use the most concise syntax by omitting **{}**, the **return** keyword and argument **()**.

1. Given an array of vegetables, use **map()** to make new array of fresh veggies: ['fresh beet', 'fresh carrot', etc]

```
    const veggies = ['beet', 'carrot', 'celery', 'cucumber', 'broccoli',
'cauliflower', 'lettuce'];

    const freshVegs = veggies.map(e 'fresh ' + e);

    console.log(freshVegs); // ['fresh beet', 'fresh carrot', 'fresh
celery', 'fresh cucumber', 'fresh broccoli', 'fresh cauliflower', 'fresh
lettuce'];
```

2. Use **filter()**, make a new array containing only the veggies that start with the letter 'c':

```
    const Cvegs = veggies.filter(v => v[0] == 'c');

    console.log(Cvegs); // ['carrot', 'celery', 'cucumber', 'cauliflower']
```

3. Using filter-into-map chaining, get just the veggies that start with 'c', but with the word 'crunchy' before each veggie:

```
    const crunchyVegs = veggies.filter(v => v[0] == 'c').map(v => 'fresh '
+ v);
    console.log(crunchyVegs); // ['crunchy carrot', 'crunchy celery',
'crunchy cucumber', 'crunchy cauliflower']
```

pluralize words
Using **map**, pluralize each fruit in the provided fruits array:

```
    const fruits = ["apple", "banana", "blueberry", "cherry", "grape",
"kiwi", "lemon", "mango", "orange", "papaya", "peach", "strawberry"]
```

Pluralization rules are as follows:

- if word ends in "y", drop the "y" and add "ies" ("cherry" -> "cherries")
- if word ends in "h" or "o", add "es" ("peach" --> "peaches", "mango" --> "mangoes")

- otherwise, add "s" ("apple" --> "apples")

```
    const fruitPl = fruits.map(fru {
        let lastChar = fru[fru.length-1];
        let allButLastChar = fru.slice(0, fru.length-1);
        if(lastChar == "y") {
            return allButLastChar + "ies";
        } else if(lastChar == "h" || lastChar == "o") {
            return fru + "es";
        } else {
            return fru + "s";
        }
    });

    console.log(fruitPl);
```

4. Given two arrays, **furniture** and **woods**, use map to generate a new **woodFurniture** array, having all ten pieces of furniture, each with a random wood types:

```
    const furniture = ["Desk", "Chair", "Bed", "Table", "Sofa", "Card
Table", "Tea Table", "Chest", "Dresser", "Sideboard"];

    const woods = ["Oak", "Walnut", "Mahogany", "Maple"];
```

5. Use map; inside the function, generate a random number in the range of the woodTypes array and use that value to select a random wood type:

```
    const woodFurn = furniture.map(e => {
        let r = Math.floor(Math.random() * woods.length);
        return woods[r] + " " + e;
    });

    console.log(woodFurn);
```

assigning apartment numbers using map to make a 2D array from a 1D array

An apartment building has four apartments on each of six floors. The units are provided:

```
    const letters = ['A', 'B', 'C', 'D'];
```

Using map, generate all 24 apartment units and save them to a nested array, consisting of 6 items, each an array of 4 items.

Desired output: // [['1A', '1B', '1C', '1D'], ['2A', '2B', '2C', '2D'], ['3A', '3B', '3C', '3D'], ['4A', '4B', '4C', '4D']]

6. Call the map method on the array, saving whatever is returned in the end to a new array, aptNums:

```
    const aptUnits = unitLetters.letters.map();
```

7. each iteration of map requires a loop to make the four-item array, which is all the apts for one floor.

```
    const apts = letters.map(e => {
        let floor = [];
        for(i = 1; i <= 4; i++) {
            floor.push(i + e);
        }
        return floor;
    });

    console.log(apts); // [ ['1A', '1B', '1C', '1D'], ['2A', '2B', '2C',
'2D'], ['3A', '3B', '3C', '3D'], ['4A', '4B', '4C', '4D'] ]
```