Array Extras

Project code name: array-extras (create folder in dropbox)
Create a single HTML file, and use separate javascript file for each ex

Array forEach() , map() and filter()

- Go back to your exercise-runner, copy 3 of your exercises (38, 44, 46) to the new project and use forEach() map() and filter() instead of for-loops

- Write the function onlyOneWord(strs) – returns only those strings with a single word (no spaces)

```
var input = ['return', 'phrases', 'with one word'];
var expected = ['return', 'phrases']
var actual = onlyOneWord(input)
```

- Write the function reverseAll(strs) – gets an array of strings and returns a new array containing string reversed

```
var input = ['abc', 'xyz'];
var expected = ['cba', 'zyx']
var actual = reverseAll(input)
```

- Write the function capitalizeLongerThan5(strs) – gets an array of strings and returns a new array in which strings that are longer than 5 are capitalized (starts with uppercase)

```
var input = ['abcdefg', 'xyz'];
var expected = ['Abcdefg', 'xyz']
var actual = capitalizeLongerThan5 (input)
```

- Write the function onlyVowels(strs) – gets an array of strings and returns a new array containing only vowels from each string:

```
var input = ['average', 'exceptional', 'amazing'];
var expected = ['aeae', 'eeioa', 'aai']
var actual = onlyVowels(input)
```

- Bonus: Write the function doubleMatrix(mat) – that doubles the numbers in the matrix, maintaining the same structure

```
var input = [[1, 2, 3],
            [4, 5, 6],
            [7, 8, 9]];

var expected = [[2, 4, 6],
                [8, 10, 12],
                [14, 16, 18]];

var actual = doubleMatrix(input)
```

- Make sure you don’t change the original matrix.

array.find and array.findIndex

Managing an array of movies:

```
var gMovies = [
    {imdb: 'tt0373889', name: 'Harry Potter'},
    {imdb: 'tt0000004', name: 'Un bon bock'},
    {imdb: 'tt0000003', name: 'Pauvre Pierrot'},
]
```

Write the following functions:

- getMovieLink(imdb) that returns a link to that movie, the function returns an HTML like:
  <a href="https://www.imdb.com/title/tt0073052/">Harry Potter</a>
- deleteMovie(imdb) that removes a movie from the array
  TIP: use findIndex

array.every and array.some

1. Write a function allPassed(students) that gets an array of students (name, grade) and returns true if they all passed (grade >= 70)

2. Write a function isGameOn(players) that gets an array of players (name,isAlive) and returns true if one of them is still alive

3. Write a function *isMatrix(arr2d)* that gets a 2d array and validate that it is a matrix (= all rows are of the same length)

4. Bonus: Write a function *isWide(arr2d)* that gets a 2d array and check that at least one of its rows has more than 5 column

5. Bonus: Write the function positiveRowsOnly (mat) – return only the rows in the matrix that have all positive integers
   TIP: use filter and every

```
var input = [[1, 10, -100],
             [2, -20, 200],
             [3, 30, 300]];
var expected = [[3, 30, 300]];
var actual = positiveRowsOnly(input)
```

Reduce

Consider the following data structure:

```
var emps = [
   {
     name: 'Joe Schmoe',
     yearsExperience: 5,
     department: 'IT'
   },
   {
     name: 'Sally Sallerson',
     yearsExperience: 15,
     department: 'Engineering'
   },
   {
     name: 'Bill Billson',
     yearsExperience: 5,
     department: 'Engineering'
   },
   {
     name: 'Jane Janet',
     yearsExperience: 15,
     department: 'Management'
   },
   {
     name: 'Bob Hope',
     yearsExperience: 9,
     department: 'IT'
   }
];
```

Reduce them all

Use reducers to calculate the following:

- All experience sum
- Sum each department's collective experience
  (create a map object dept -> experience)
- Group employees by experience (an object in which the key is a number and the value is an array of employee objects)
  (create a map object experience ->{key:[{employee}, {employee}]})
- Count the number of employees in each department

2. Write a function findModes(values) that gets an array and uses Array.reduce to create a map and then prints the numbers that occur most often.

3. Write a function flatten(values) that flattens the array, meaning that if an item in this array is an array, it will push all its values to the result array.
    a. i.e. Input: [‘Hello’, [9, 6] ,18, [4, 7, 8]]
    b. output: [‘Hello’, 9, 6 ,18, 4, 7, 8]
    c. support only one level of nested values
    d. Bonus: use recursion to support any level