SYNERGY
GLOBAL FINANCIAL SOLUTIONS

Бонус задачка 1

Работа с функции, видове функции в JavaScript / анонимни функции / ламбда функции / особености при употреба. Приложение на функциите

Теоретични основи:

https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Functions/Arrow_functions
https://chrisrng.svbtle.com/es6-lambda-expressions
http://exploringjs.com/es6/ch_arrow-functions.html
https://medium.freecodecamp.org/es6-functions-9f61c72b1e86

Полезни ресурси:
http://underscorejs.org
https://github.com/janl/mustache.js/
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

SYNERGY
GLOBAL FINANCIAL SOLUTIONS

Упражнения:

Работа с колекции

#1: Обхождането на колекции е процес по преминаване, през всички налични елементи на един масив или обект. Напишете имплементация на функцията forEach, която приема следните аргументи:

- колекция която трябва да се обходи
- Функция, (callback) която трябва да се изпълни за работа с елемента

Примерен код

```
forEach([1,2,3,4], function(element) {
        console.log(element)
});
```

#2: Когато работим с колекции често ни се налага да елиминираме определени елементи и да обработваме само тези, които отговарят на определено условие. За тази цел е необходимо да създадем нова колекция от вече съществуващата.

Например необходими са ни всички четни числа в даден масив, или всички числа които са по-големи от дадена стойност.

Напишете имплементация на функцията filter, която приема следните аргументи:

- колекция която трябва да се обходи
- Функция (callback) която трябва да се изпълни за работа с елемента

Функцията връща като резултат, нова колекция, копие на първата.

Примерен код

```
filter([1,2,3,4], function(element) {
  return (element % 2)
});
```

Добавете функционалност за работа върху съществуващата колекция. Потребителя подава трети аргумент указващ, дали функцията да връща **копие** или да работи върху **съществуващата подадена като аргумент колекция**

#3: Не всички упражнения имат дълги и сложни условия 😊. Напишете имплементация на функцията map.

Примерен код

```
map([1,2,3,4], function(element) {
  return (element * 2)
```

});

#4: Много е дразнещо когато, трябва да напълним колекцията си с примерни данни за да я ползваме в нашите експерименти, но не можем да измислим повече от 10 числа. Време е да автоматизираме този процес. Напишете имплементация на функцията fill, която пълни колекция със случайно генерирани стойности.

Функцията приема като първи аргумент колекция, в която ще съхраняваме стойностите си. Колекцията **може да съдържа елементи** а **може и да е празна**.

Като втори аргумент функцията приема колекция от символи, които ще бъдат ползвани като своеобразен източник на данни с която ще напълним колекцията.

Изисквания към функцията:

Функцията приема **трети аргумент**, дължина на резултатната колекция. Брой случайно генерирани символи които ще бъдат добавени към колекцията. **Ако символите в колекцията са повече от аргумента, то функцията връща резултатната колекция без промяна**

Примерен код

```
fill([],[‘a’, ‘b’, ‘c’, 1, 10, 8, 7, 10], 5);
```

Примерен резултат

[‘a’, ‘a’, ‘a’, 10, 1]

Примерен код

```
fill([1, 2, 5, 6, 7 , 8, 8, 8],[‘a’, ‘b’, ‘c’, 1, 10, 8, 7, 10], 5);
```

Примерен код

[1, 2, 5, 6, 7, 8, 8, 8]

#5: Напишете имплементация на функцията reverse. Функцията има за цел да вземе колекция с елементи и да промени тяхното позициониране в колекцията.

Примерен код

```
reverse([‘a’, ‘b’, ‘c’, 1, 10, 8, 7, 10]);
```

Примерен резултат

[10, 7, 8, 10, 1, ‘c’, ‘b’, ‘a’]

Функцията трябва да приема колекция от колекции, които да могат да бъдат преобърнати.

reverse([[1,2,3,4], [‘a’, ‘b’, ‘c’, ‘d’]]);

[[‘d’, c’, ‘b’, ‘a’**]** , **[**4,3,2,1**]** **]**

Предаване на упражненията

След приключване на работата си по упражненията, кандидатите качват, кода си в Git репозиторито в което качвате и останалите задачи, като названието на директорията трябва да бъде дефинирано като bonus-1