C1 - Learn JavaScript

C-COD-120

JavaScript Day01

Learn JavaScript

1.2

JavaScript Day01

repository name:	javascript_d01
repository rights:	ramassage-tek
language:	JavaScript

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).

Foreword

For the next half day, you will have a series of 7 exercises to get familiar with the JavaScript language. The exercises are independent, therefore, you can solve them in the order that best suits you.

We will use **node** to launch each of your JavaScript files.

EXERCICE 01 (2 PTS)

File to hand in: ex_01/drawTriangle.js

Write a function **drawTriangle**. This function will take a height as parameter and draw a triangle on the standard output.

That height will be the triangle height.
You must perform error handling in your code.

Prototype: drawTriangle(height);

Terminal

```
~/C-COD-120> cat drawTriangle.js
//
// Here
// Is
// Your
// Own
// Algo
//

// Code to execute
drawTriangle(6);
```

Terminal

```
~/C-COD-120> node drawTriangle.js
$
$$
$$$
$$$$
$$$$$
$$$$$$
```

EXERCICE 02 (2 PTS)

File to hand in: ex_02/arrayIsEqual.js

You will write a function named **arrayIsEqual** returning true or false. This function takes two arrays as parameters.

Your function will return true if both arrays are equal and false otherwise.

Prototype: arrayIsEqual(arr1, arr2);

Terminal

```
~/C-COD-120> cat arrayIsEqual.js
// Here a little test
var a = [1, 2];
var b = [3, 4];
console.log(arrayIsEqual(a,b));
```

Terminal

```
~/C-COD-120> node arrayIsEqual.js
false
```

EXERCICE 03 (2 PTS)

File to hand in: ex_03/countGs.js

Write a function countGs that takes a string as its only argument and returns a number that indicates how many uppercase 'G' characters are in the string.

Prototype: countGs(str);

Terminal

```
~/C-COD-120> cat countGs.js
console.log(countGs("abcgGggGeaGfdsGG"));
```

Terminal

```
~/C-COD-120> node countGs.js
5
```

EXERCICE 04 (3 PTS)

File to hand in: ex_04/fizzBuzz.js

Write a function **fizzBuzz** that print all the numbers from 1 to 20.

Three requirements:

- For numbers divisible by **3**, print "**Fizz**" instead of the number.
- For numbers divisible by **5 (and not 3)**, print "**Buzz**" instead of the number.
- For numbers that are divisible by both **3 AND 5** print "**FizzBuzz**".

Every print, be it a number or the appropriate replacements, should be comma separated.

Prototype: fizzBuzz();

```
Terminal
∼/C-COD-120> node fizzBuzz.js | cat -e
1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz, 16, 17,
Fizz, 19, Buzz$
```

Exercice 05 (4 pts)

File to hand in: ex_05/range.js

You will write a **range** function that takes three arguments, **start**, **end**, **step** and returns an array containing all the numbers from start up to (and **including**) end.

Your range function third argument that indicates the **step** value used to build up the array is optional.

Indeed, if a third argument is not provided, the array elements go up by increments of one, corresponding to the classic step-ping behavior.

Make sure it also works with negative step values so that range

Prototype: range(start, end, step);

```
Terminal
~/C-COD-120> cat range.js
// your code is here

// few tests
console.log(range(1, 10, 2));
console.log(range(19, 22));
console.log(range(5, 2, -1));
```

```
Terminal
~/C-COD-120> node range.js
[1, 3, 5, 7, 9]
[19, 20, 21, 22]
[5, 4, 3, 2]
```

EXERCICE 06 (4 PTS)

File to hand in: ex_06/objectsDeeplyEqual.js

Write a function **objectsDeeplyEqual**, that takes two values and returns true only if they are the same value or are objects with the same properties whose values are also equal when compared with a recursive call to **objectsDeeplyEqual**.

The word is out, you will be using **recursion**

Tip 1: your function must compare two things by identity or by examining their properties.
Tip 2: null is also an "object".
Tip 3: if your function passes the tests below your gecko will be happy.

Your function is not supposed to be too complex, keep in mind that this is only the first day.

Prototype: objectsDeeplyEqual(cmp1, cmp2);

Terminal

```
~/C-COD-120> cat objectsDeeplyEqual.js
// Your implementation

//some tests
var obj = {here: {is: "an"}, object: 2};

console.log(objectsDeeplyEqual(obj, obj));
console.log(objectsDeeplyEqual(obj, {here: 1, object: 2}));
console.log(objectsDeeplyEqual(obj, {here: {is: "an"}, object: 2}));
```

Terminal

```
~/C-COD-120> node objectsDeeplyEqual.js
true
false
true
```

Exercice 07 (3 pts)

File to hand in: ex_07/arrayFilter.js

Write a function **arrayFilter** taking two arguments: an **array** and a **test**.

The argument named **test** is a function. You dont have to care about this function implementation.

This function must be called for each element contained in the array given as parameter, the function will return a boolean, this return value determines whether an element is included in the returned array or not (if the test succeed it should be included in the returning array).

The arrayFilter function returns a new array containing filtered values.

Prototype: arrayFilter(array, test);

Terminal

```
~/C-COD-120> cat arrayFilter.js
// Your implementation

// Use this to test
var toFilter = [1, 2, 3, 4, 5, 6, 7, 8, 9];

// the anonymous function is the test your filtering function will use to make
decision

var passed = arrayFilter(toFilter, function (value) {
return value % 3 === 0;
});
console.log(passed);
```

Terminal

```
~/C-COD-120> node arrayFilter.js
[3,6,9]
```