C1- Code & Go

Javascript_D01

Javascript-D01

Learn Javascript

CODING ACADEMY
BY EPITECH

Administrative Details

- The project is to be done alone.
- Location of files to hand in: **Javascript_D01**

Foreword

Javascript-D01

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

Draw Triangle

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.

Obviously that height will be the triangle height.

```
Terminal
~/Javascript_D01> cat drawTriangle.js
//your code

drawTriangle(6);
```

```
Terminal
~/Javascript_D01> node drawTriangle.js
$
$$
$$$
$$$$
$$$$$
$$$$$$
```

Prototype: drawTriangle(height)

Exercice 02

Array Is Equal ?

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

```
~/Javascript_D01> cat arrayIsEqual.js
// obviously this is done to show a small test
var a = [1, 2];
var b = [3, 4];

if (arrayIsEqual(a,b))
console.log("True");
else
console.log("False");
```

Terminal

```
~/Javascript_D01> node arrayIsEqual.js
False
```

Exercice 03

Count G's

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)

Exercice 04

Fizz Buzz !?

File to hand in: ex_04/fizzBuzz.js

Write a function **fizzBuzz** that prints 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 replacement, should be comma separated.

```
Terminal
~/Javascript_D01> 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$
```

Prototype: fizzBuzz()

Exercice 05

Range

File to hand in: ex_05/range.js
Restriction: Only ES5 is allowed.

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

Your range function's 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 stepping behavior.

Make sure it also works with negative step values for that range.

Terminal

```
~/Javascript_D01> 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

```
~/Javascript_D01> node range.js
[1, 3, 5, 7, 9]
[19, 20, 21, 22]
[5, 4, 3, 2]
```

Prototype: range(start, end, step)

Exercice 06

Objects Deeply Equal

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 should find out whether to compare two things by identity or by looking at their properties.

Tip 2: null is also an "object".

ip 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.

Terminal

```
~/Javascript_D01> 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

```
~/Javascript_D01> node objectsDeeplyEqual.js
true
false
true
```

Prototype: objectsDeeplyEqual(cmp1, cmp2)

Exercice 07

Array Filter

File to hand in: ex_07/arrayFilter.js
Prototype: arrayFilter(array, test)

Write a function **filter** 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, and this return value determines whether an element is included in the returned array or not (if the test succeeded it should be included in the returning array).

The arrayFilter function returns a new array containing filtered values.

Terminal

```
~/Javascript_D01> 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 a decision

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

Terminal

```
~/Javascript_D01> node arrayFilter.js
[3,6,9]
```