T5 - Web Development Seminar

T-WEB-500

Day 06

Javascript

During the days to come, you will have a series of exercises to get familiar with the JavaScript language.

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

This day is tested by the autograder!

TASK 01 - DRAW ME A TRIANGLE

Turn in: `task01.js`

Write a function `drawTriangle` that takes a height as parameter and draws a triangle on the standard output.
The height corresponds to the triangle height (see below).
The function must be exportable, and in a `task01.js` file.

Prototype: `drawTriangle(height)`

```
Terminal
~/T-WEB-500> cat task01.js
module.exports = {
drawTriangle: function(limit) {
//
// your code here
//
}
}
```

Your function will be tested the following way:

```
Terminal
~/T-WEB-500> cat task01_tester.js
const task01 = require('./task01');

task01.drawTriangle(6);
~/T-WEB-500> node task01_tester.js
$
$$
$$$
$$$$
$$$$$
$$$$$$
```

TASK 02 - ARRAYSAREEQUAL

Turn in: `task02.js`

Write a function named `arraysAreEqual` that returns true if both arrays passed as parameters are equal, false otherwise.
The function must be exportable, and in a `task02.js` file.

Prototype: `arraysAreEqual(arr1, arr2)`

```
Terminal
~/T-WEB-500> cat task02.js
module.exports = {
arraysAreEqual: function(arr1, arr2) {
//
// your code here
//
}
}
```

Your function will be tested the following way:

```
Terminal
~/T-WEB-500> cat task02_tester.js
const task02 = require('./task02');

console.log(task02.arraysAreEqual([1, 2], [1, 4]) ? 'True' : 'False');
~/T-WEB-500> node task02_tester.js
False
```

TASK 03 - COUNTGS

Turn in: `task03.js`

Write a `countGs` function that takes a string as parameter and returns the number of uppercase 'G' characters it contains.
The function must be exportable, and in a `task03.js` file.

Prototype: `countGs(str)`

TASK 04 - FIZZBUZZ

Turn in: `task04.js`

Write a `fizzBuzz` function that takes a number as parameter and prints all the numbers from 1 to this number.
Three requirements:

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

The function must be exportable, and in a `task04.js` file.

Every output (be it a number or a string) should be comma separated.

Prototype: `fizzBuzz(num)`

```
Terminal
~/T-WEB-500> node task04_tester.js 20 | cat -e
1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz, 16, 17,
Fizz, 19, Buzz$
```

TASK 05 - RANGE

Turn in: `task05.js`
Restriction: only ES5 is allowed

Write a `range` function that takes three arguments (start, end and step) and returns an array containing all the numbers from start up to (and including) end.
The third argument indicating the step value used to build up the array is optional. If not provided, the array elements go up by increments of one, corresponding to the classic stepping behavior.
The function must be exportable, and in a `task05.js` file.

Make sure it also works with negative step values.

Prototype: `range(start, end, step)`

```
Terminal
~/T-WEB-500> cat example.js
// ...
console.log(range(1, 10, 2));
console.log(range(19, 22));
console.log(range(5, 2, -1));
~/T-WEB-500> node example.js
[1, 3, 5, 7, 9]
[19, 20, 21, 22]
[5, 4, 3, 2]
```

Task 06 - objectsDeeplyEqual

Turn in: `task06.js`

Write an `objectsDeeplyEqual` function 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.

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

'null' is also an "object".

Your function is not supposed to be too complex, keep in mind that this is only the first day of javascript. The function must be exportable, and in a `task06.js` file.

Prototype: `objectsDeeplyEqual(cmp1, cmp2)`

```
~/T-WEB-500> cat example.js
// ...
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}));
~/T-WEB-500> node example.js
true
false
true
```

TASK 07 - ARRAYFILTERING

Turn in: `task07.js`

Write a `arrayFiltering` function that takes two arguments : an array and a `test` function.
The argument named `test` is a function that returns a boolean.
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 return values determine whether an element is included in the returned array or not (if the test succeed, it should be included in the returning array).

The `arrayFiltering` function returns a new array containing filtered values.
The function must be exportable, and in a `task07.js` file.

Prototype: `arrayFiltering(array, test)`

```
~/T-WEB-500> cat example.js
// ...
var toFilter = [1, 2, 3, 4, 5, 6, 7, 8, 9];
var res = arrayFiltering(toFilter, function (value) {
    return value % 3 === 0;
});
console.log(res);
~/T-WEB-500> node example
[3,6,9]
```