WEB DEV SEMINAR

DAY 06 - JAVASCRIPT

WEB DEV SEMINAR

Table of contents

The next few days will be dedicated to the JavaScript language. Let's get comfy with it.

We will use `bun` to run your JavaScript files.

This day is tested by the autograder!

Task 01

Delivery: `task01.js`

Prototype: `drawTriangle(height)`

Write a function `drawTriangle` that:

- ✓ takes one parameter ;
- ✓ draws a triangle on the standard output ;
- ✓ uses one parameter corresponding to the triangle height (see below) ;
- ✓ is exported and contained in a `task01.js` file.

```
T-WEB-500> cat task01.js
export function drawTriangle(height) {
//
// your code here
//
}
```

Your function will be tested the following way:

```
T-WEB-500> cat task01_tester.js
import { drawTriangle } from 'task01.js';
drawTriangle(5);
```

```
T-WEB-500> bun task01_tester.js
$
$$
$$$
$$$$
$$$$$
```

Task 02

Delivery: `task02.js`

Prototype: `arraysAreEqual(arr1, arr2)`

Write a function `arraysAreEqual` that:

- ✓ takes two arrays as arguments ;
- ✓ returns true if both arrays are equal, false otherwise.,
- ✓ is exported and contained in a `task02.js` file.

Terminal

```
T-WEB-500> cat task02.js
export function arraysAreEqual(arr1, arr2) {
//
// your code here
//
}
```

Your function will be tested the following way:

Terminal

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

Terminal

```
T-WEB-500> bun task02_tester.js
False
```

Task 03

Delivery: `task03.js`

Prototype: `countGs(str)`

Write a `countGs` function that:

- ✓ takes a string as parameter ;
- ✓ returns the number of uppercase 'G' characters it contains.
- ✓ is exported and contained in a `task03.js` file.

Task 04

Delivery: `task04.js`

Prototype: `fizzBuzz(num)`

Write a `fizzBuzz` function that:

- ✓ takes a number as parameter ;
- ✓ prints all the numbers from 1 to this number ;
- ✓ is exported and contained in a `task04.js` file ;
- ✓ complies with the following requirements:
  1. print "Fizz" instead of the number if it is divisible by 3 ;
  2. print "Buzz" instead of the number if it is divisible by 5 and not by 3 ;
  3. print "FizzBuzz" instead of the number if it divisible by both 5 and 3.

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

Terminal

```
T-WEB-500> bun 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

Delivery: `task05.js`
Restriction: only ES5 is allowed
Prototype: `range(start, end, step)`

Write a `range` function that:

- ✓ takes 3 integers as arguments (`start`, `end` and `step`) ;
- ✓ returns an array containing all the numbers from `start` up to `end` included ;
- ✓ with an increment corresponding to the optional third argument `step` (increment = 1 if `step` not provided)
- ✓ is exported and contained in a `task05.js` file.

```
Terminal
T-WEB-500> cat example.js
// ...
console.log(range(1, 10, 2));
console.log(range(19, 22));
console.log(range(5, 2, -1));
```

```
Terminal
T-WEB-500> bun example.js
[1, 3, 5, 7, 9]
[19, 20, 21, 22]
[5, 4, 3, 2]
```

You code should cover all types of integers!

Task 06

Delivery: `task06.js`
Prototype: `objectsDeeplyEqual(cmp1, cmp2)`

Write an `objectsDeeplyEqual` function that:

- ✓ takes two arguments ;
- ✓ returns true only if:
  - they have the same value ;
  - or they are objects with the same properties whose values are also equal when compared with a recursive call to `objectsDeeplyEqual` ;
- ✓ is exported and contained in a `task06.js` file.

```
Terminal
T-WEB-500> cat example.js
// ...
const 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
T-WEB-500> bun example.js
true
false
true
```

The word is out: you will be using recursion.

Your function should figure out whether to compare two things by identity or by looking at their properties.
Your function is not supposed to be too complex, keep in mind that this is only the first day of Javascript.

‘null’ is also an “object”.

Task 07

Delivery: task07.js
Prototype: arrayFiltering(array, test)

Write a arrayFiltering function that:

- ✓ takes two arguments, array and test ;
- ✓ test is a function returning a boolean ;
- ✓ returns a new array, containing some filtered values ;
- ✓ calls the test function for each element contained in array ;
- ✓ returns only values for which test returned True ;
- ✓ is exported and contained in a task07.js file.

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

```
T-WEB-500> bun example
[3,6,9]
```

Your function should NOT care about the implementation of the test function.

v 4.0

EPITECH. TECHNOLOGY