JavaScript

Exercises

MVT21

JavaScript

Goals

- Get familiar with modern JavaScript
- Be able to debug code using a debugger
- Be able to set up and implement unit tests using JEST
- Learn to set up a backend system using Node.js
- Get familiar with SQL and how to setup a database

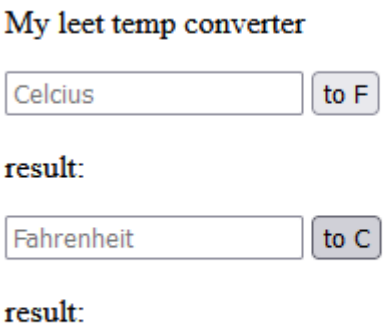
Jonathan Sönnerup
2021–12–10

Contents

Chapter 1

Front-end JavaScript

1.1 Your friend from abroad comes to visit, and she does not understand how to read a temperature scale in Celsius. Albeit being a novice programmer, you have the skills to implement a simple website converting Fahrenheit to Celsius, and vice versa. An example of this awesome project is shown below.



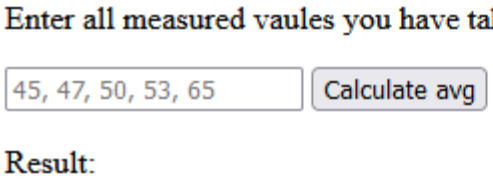
If you don't remember the formula for conversion, we give it here

$$T_C = \frac{T_F - 32}{1.8}$$

$$T_F = T_C \times 1.8 + 32$$

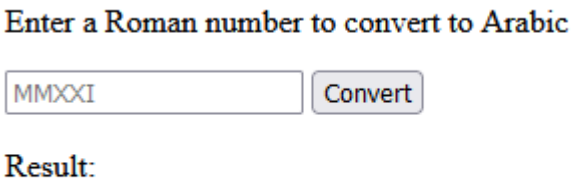
where T_C is the temperature in Celsius, and T_F is the temperature in Fahrenheit.

1.2 In a school project, you are to measure the temperature in your computer during the day, and calculate the average. You want to be able to input all values separated with a comma in a text field directly (not one at a time). Example below.



Hint: Check out the `split()` function for strings, `https://www.w3schools.com/jsref/jsref_split.asp`

1.3 Your dusty friend Pythagoras comes to visit during his trip around Europe. Being a somewhat "old school" mathematician, he is not familiar with the modern number system. You decide to make it easier for him to order a cheese burger in a drive-through by programming a Roman → Arabic converter (you do know that we are using Arabic numbers usually?). Example below.



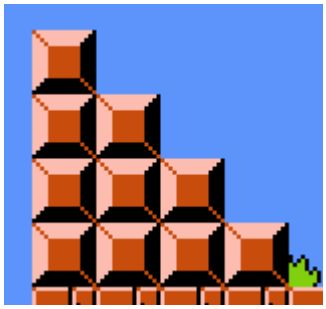
Note that this is quite a tricky challenge. To make it easier for you, only handle Roman numbers where you go from a larger number to a smaller, e.g., `XVI`, and not e.g., `XIV` as this is much harder.

Hint: You can of course create a bunch of if statements to convert all values, but try to use a map instead, example:

```
let roman = {"M": 1000, "D": 500, ...};
console.log(roman["D"]); // will print 500
// we can also do:
console.log(roman.D);
```

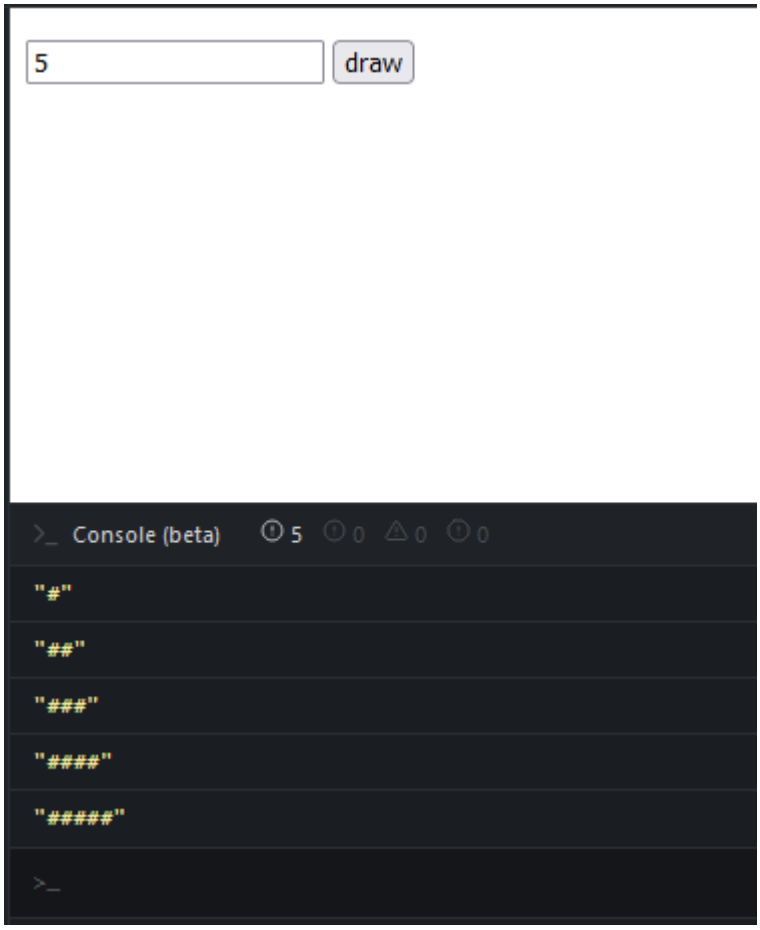
Listing 1.1: Example of using maps.

1.4 You are a fed up with all these modern games like Fartnite, Battleshit 2042, and Cyberprank. Since you have been taking a programming course, you decide to bring the old favourite, Super Mario Bros, back into life. You feel like designing the levels is a good place to start. The first objective is to implement the infamous obstacle, the stair.

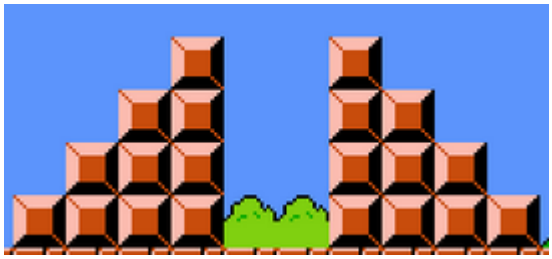


Although it is possible to work with images in JavaScript, you use hashtag "#" as a block instead. Create a simple web page where you can enter the height of the stair, and print the result in the console. Example below.

Note that you have to use nested loops to solve this. `https://sebhastian.com/nested-loops-javascript/`

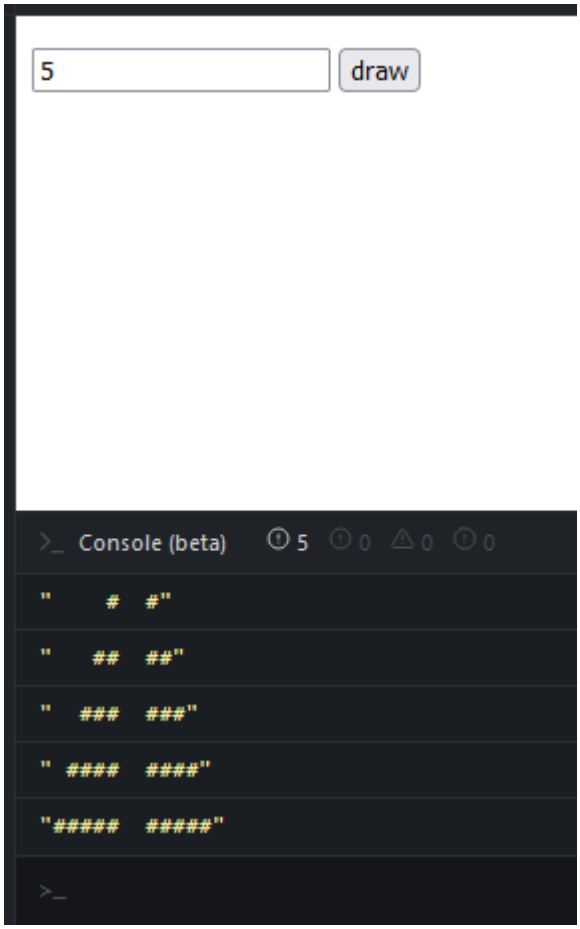


1.5 One stair is cool and all, but what is even cooler is the double stair.



This is similar to the previous exercise, but includes the double stair.

Hint: figure out how many spaces you need before the first "#" for the left stair. The right stair is the same code as the previous exercise.



1.6 The company Haribo is looking to launch a new family of sweets. Since their current best selling product MAOAM happens to be a palindrome, they believe that the next product name should also be a palindrome to be a top selling product.

Haribo knows that you are somewhat of an array expert in JavaScript, so they hire you to help with the product name. Your task is go implement a function that checks whether a string is a palindrome or not.

Make sure that both words and sentences work, such as:

- Madam
- Taco cat
- Yo, banana boy

It is common not to consider spaces, and punctuation as part of the sentence. So it would be a good idea to remove those characters first.

Hints:

- Use an array to store the letters (not punctuation etc.)
- Use the `reverse()` function on the array

1.7 Modify (or create a new version) the program from the previous example. This time, instead of using the `reverse()` function, utilize the stack functions, i.e., `push()` and `pop()`. Remember that when popping an element from a stack (array in this case) you get the latest added value.

1.8 You get hired at a startup company working with AI (Artificial Intelligence) for scanning barcodes. Your task is to detect if there are *at least* 3 consecutive identical values (3 in a row) in an image.

Another team handles the image analysis, and you are handed the input as a binary array, consisting of 1s (ones) and 0s (zeros). An example input is shown below.

$$[1, 0, 0, 1, 1, 0, 1, \underbrace{0, 0, 0}, 1, 1, 0, 1, 0]$$

$$[1, 0, \underbrace{1, 1, 1, 1, 1}, 0, 0, 1, \underbrace{0, 0, 0}, 1, 0]$$

In both of the cases, we should return true.

Hint: Use a loop and the `slice()` method to extract a sub-array of size 3. If all values are equal in this smaller array, you know that we have found our "three in a row" and may return true. See the section "Slicing an Array" here `https://www.w3schools.com/js/js_array_methods.asp`.

Chapter 2

Unit Tests

2.1 Your neighbour Dr. J. Est has decided to finally build a time machine, in order to trade Pizzas for Bitcoins. He has written a function to determine the amount of flux needed for his flux capacitor to pass through the time-space continuum.

Although he is an excellent programmer, he writes bad comments and believes that testing is equivalent to worship Satan. You are invited on this trip, so you want to make sure that everything works as intended. You take a look at his code and decides to put your testing skills into practice. The code is given below.

Write unit tests and try to get full coverage. **Note** that since the function uses random numbers, so when testing these cases, you should iterate the same test multiple times, preferably using a loop of, say, 100 iterations.

You find the code in the GitHub repository in the "Exercises" folder. The filename is `ex2-1.js`.

```
/**
 * Simple flux calculator to determine the amount of flux needed to leave
 * the schwartzchild radius in a black hole.
 *
 * Simply put:
 * - It is only worth calculating if b and d are non-zero.
 * - We are screwed (infinite flux) if c is non-zero while the array, a, is empty.
 * - The third value in the array (if given) sets the flux to be at lest 42 but no more
     than 69.
 * - If the array only consists of a single value, the flux will be 12 times this value
    .
 * - If we haven't been able to calculate the flux by now, the flux value should be c.
 */
function calculateFlux(a, b, c, d) {
    if (b == 0 && d == 0) {
        return 0;
    }

    if (c != 0 && !a.length) {
        return Infinity;
    }

    if (a.length >= 3) {
        let coeff = 0;
        switch (a[2]) {
            case 0:
                coeff = Math.floor(Math.random() * 11 + 1);
                break;

            case 1:
                coeff = Math.floor(Math.random() * 3 + 1);
                break;

            case 12:
                coeff = Math.floor(Math.random() * 27 + 1);
                break;

            default:
                coeff = 1;
                break;
        }

        return 42 + coeff;
    } else if (a.length == 1) {
        return a[0] * 12;
    } else {
        return c;
    }
}
```

Listing 2.1: Dr. J. Est's flux code, ex2-1.js