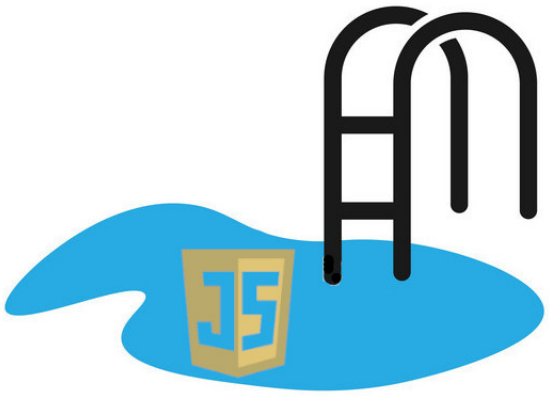
D1 - Bootcamp

JS Bootcamp - day 07

Callbacks



1.0.1

EPITECH. DIGITAL SCHOOL

JS bootcamp - day 07

delivery method: Github
language: Javascript

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

For today exercises deliver the totality of your javascript files.

As usual now, do not forget to download the resources of the day, available along with this subject. This is also true for the rest of the pool…

A **callback** is a function passed as an argument to another function, to be executed later, when a trigger event happens.
For example, a click on a button should call a given function (to display a page, add an article in the cart, pop a message for instance).

Callbacks are regularly used for asynchronous actions (initiated now, but executed later).

Mind the the Pyramid of Doom when playing with asynchronous functions…

Exercise 01

File to turn in: ex_01/ex_01.js
Function prototype: *reduceOpacity()*
Function prototype: *resetOpacity()*

Create a `reduceOpacity` function that multiplies by 0.5 the opacity of the rectangle displayed in the resource HTML page.
Create a `resetOpacity` function that resets the opacity of the rectangle to 1.

Using the previous functions, reduce the rectangle's opacity when the cursor hovers over it.

We are talking here about event listeners.

Do not forget to reset the opacity when the cursor leaves the rectangle.

EXERCISE 02

File to turn in: ex_02/ex_02.js
Function prototype: *rotateCircle(step: integer)*

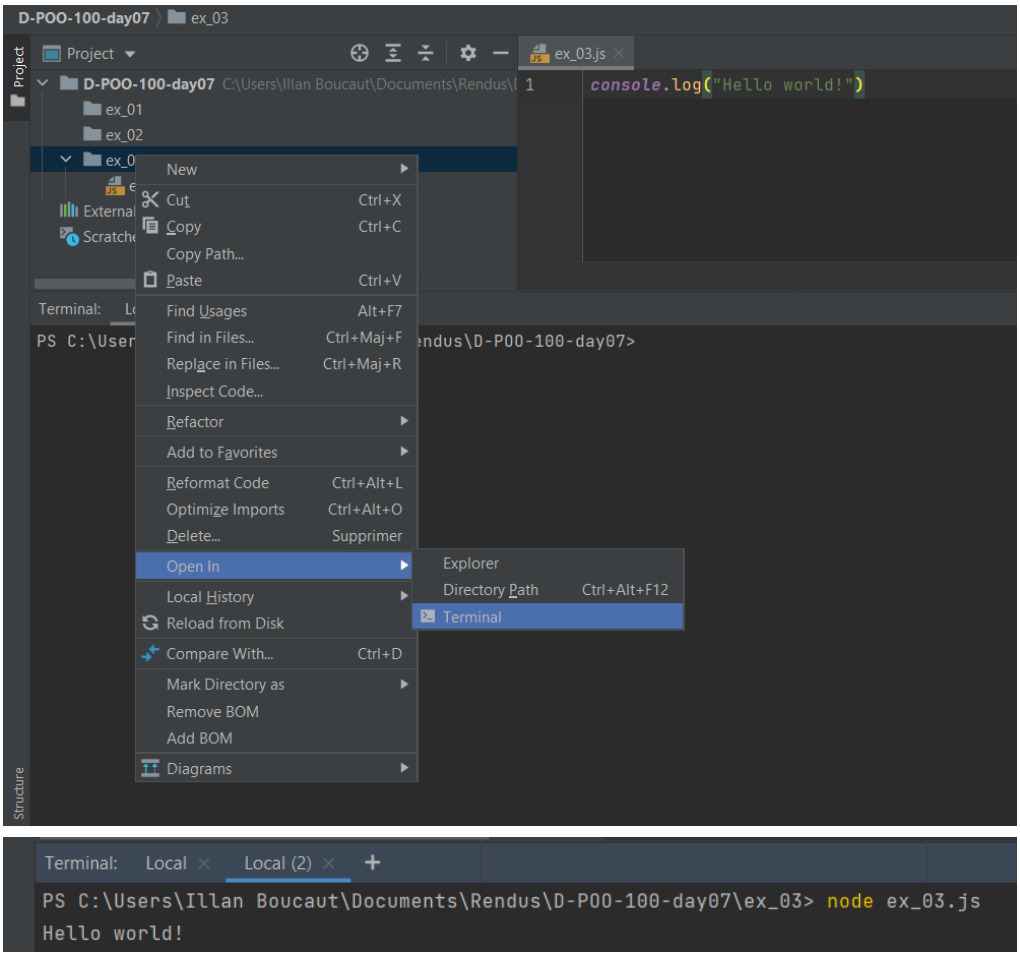
Create a `rotateCircle` function that rotates the circle by *step* degrees (*step* being given as argument). This function must be triggered when one of the top buttons of the page is clicked.

The texts on the buttons are pretty self-explanatory and give the values of the different *steps* to apply.

Interlude

The JavaScript console that you have been using since the beginning of this pool is a **terminal**.
On your computer there are also different terminals that you use from now on.

JavaScript was initially designed to be launched from a browser, until a program called **node** popped up.
It allows to launch JavaScript scripts directly from the terminal without opening a web page.
You may already have a terminal integrated in your IDE, from which you can launch your scripts:



If you wish to dig deeper into terminals, there are lots of relevant commands you can experience.
For instance,

- **cd** (for *Change Directory*) to navigate into your file system
- **ls** (for *LiSt*) to list all the files in a directory
- **man** (for *MANual*) to get all the information about any command

Take some time to get used to commands, it is a very profitable investment!!
man man

EXERCISE 03

File to turn in: ex_03/ex_03.js
Function prototype: *map(elements: array, modifier: function): array*

Rebuild the `map` method turning it into a function.
It should take as parameter an array of elements and the callback to be applied to each of its elements.
It returns a new array with the elements modified by the callback.

The following code:

```
function isEven(number) {
    return number % 2 === 0;
}
console.log(map([5, 8, 10], isEven));
```

should output `[ false, true, true ]`

If you do not know the *map* method, it is time to catch up.

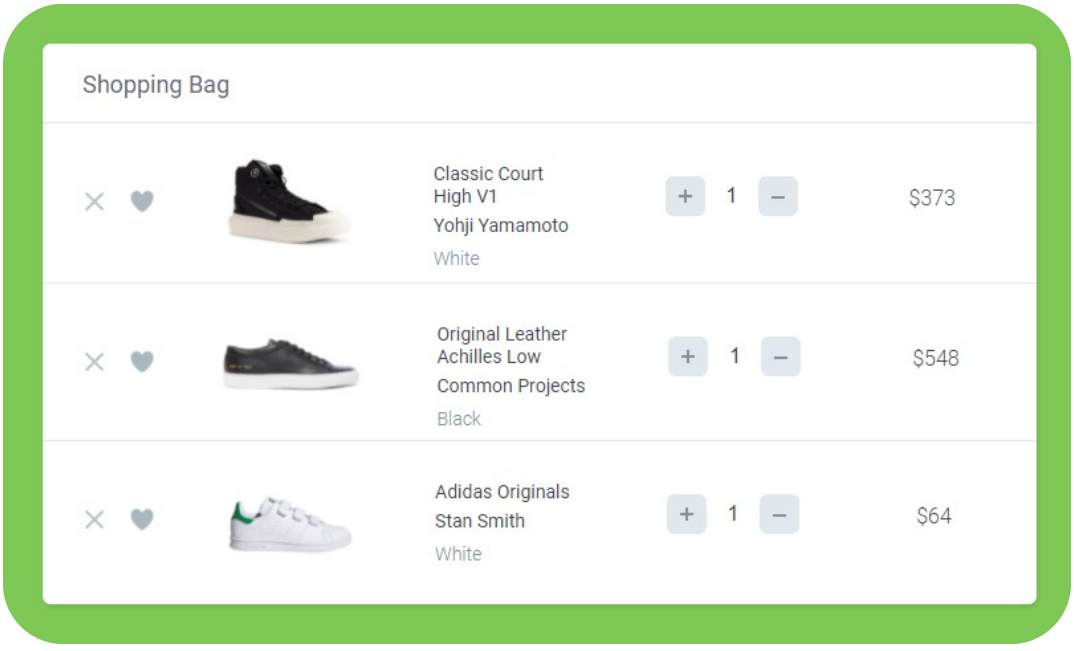
Exercise 04

Turn in: ex_04/ex_04.js

The resource is a work in progress of a shopping cart.
Make the 12 buttons functional!

Here are the expected buttons behaviors:

- `+`
  increase the quantity (no price recalculation)
- `-`
  decrease the quantity (no price recalculation)
- `x`
  delete the corresponding product
- heart
  toggle the *is-active* class to the corresponding element (if working it should animate)



Exercise 05

Turn in: ex_05/ex_05.js
Function prototype: findClosestResult(functionsObj: object, inputNumber: number, outputNumber: number): number

Create a `findClosestResult` function that takes as arguments an object (*functionsObj*), and 2 numbers (*inputNumber* & *outputNumber*).
The first arguments *functionsObj* contain functions that take a number as argument, and return another number.

Your function must return the key name (the name of the function) of the *functionsObj* object whose value (which is a function) returns the closest number to *out*, when *in* is given as input. In case of equality, return the first key found in the object.

You could test your function with this snippet of code:

```
const fObj = {};
fObj.multipleByEight = (x) => x * 8;
fObj.square = (x) => x * x;
fObj.addSixty = (x) => x + 60;

const result = findClosestResult(fObj, 5, 26);
console.log(result);
// should log: 'square' because : 5 * 5 = 25, 5 * 8 = 40, 5 + 60 = 65
console.log(findClosestResult(fObj, 10, 5));
// should log: 'addSixty'
console.log(findClosestResult(fObj, 5, 45));
// should log: 'multipleByHeight'
```

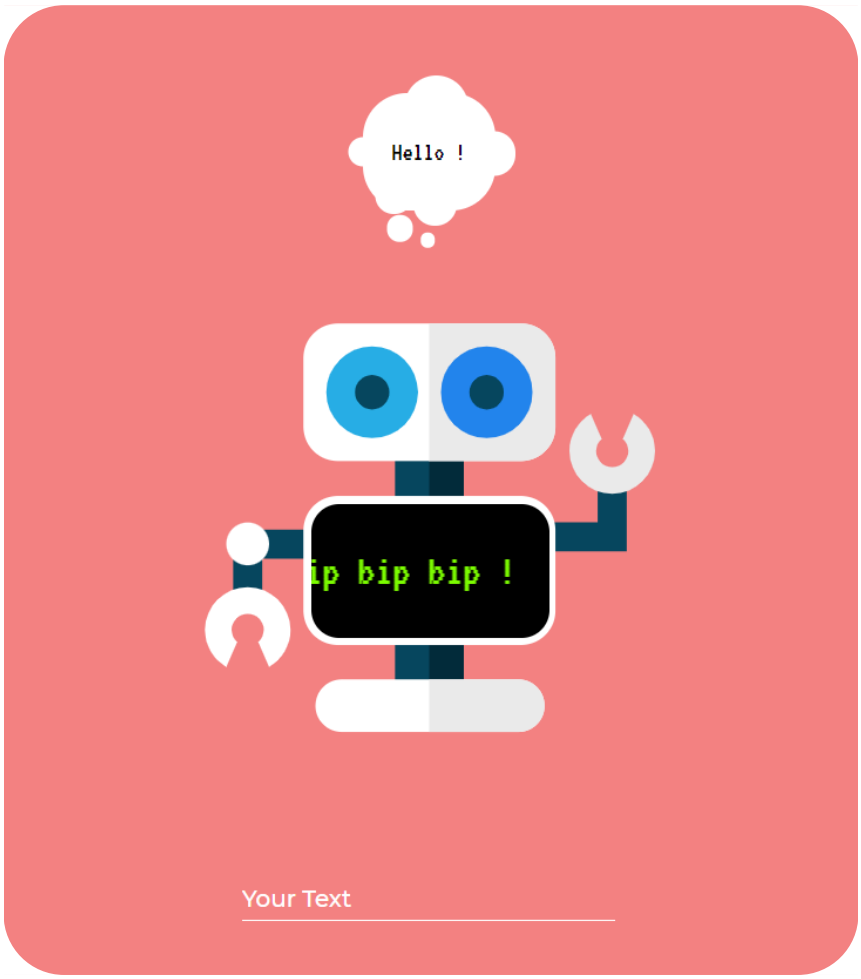
Paper and pen in hand: feel free to sketch, paraphrase or dissect the exercise step by step.

EXERCISE 06

Turn in: ex_06/ex_06.js

Interactive terminals will be set up in the various appliance stores with a mascot with which visitors can interact.
You have been commissioned to create the prototype to show its capabilities.
The prototype must behave as below:

- when the cursor clicks on the robot, display “Ooch, that hurts!” for 2 seconds on its speech bubble
- when the cursor hovers over the robot, display the coordinates of the cursor on the scrolling screen
- when a message is entered in the input, display “Don’t worry, I’ll take care of it!” for 2 secondes on the robot scrolling screen
- when the cursor clicks at least 10 times on a robot’s eye, change randomly the color of both iris



EXERCISE 07

Turn in: ex_07/ex_07.js

Write a script that:

- every 2 seconds, change randomly the order of the 4 words in the title element
- when the cursor hovers the text, it freezes
- when the user clicks on the title element, copy the text in the clipboard and the title element stops changing
- when the user clicks outside of the title element, resume the text suffling every 2 seconds
- when the ! key is pressed, an alert box displaying *42!* appears after 42 seconds

Exercise 08

Turn in: ex_08/*

From exercise 04 sources, create a basic e-commerce website.
The web page must:

- display a list of products on the left side of the page and a shopping cart on the right side
- allow adding elements into the cart, and delete them
- display the sum of the products prices in real-time
- allow modifying the quantity of a product (in or out of the cart)

Deliver everything you've created and all the resources needed for this exercice.