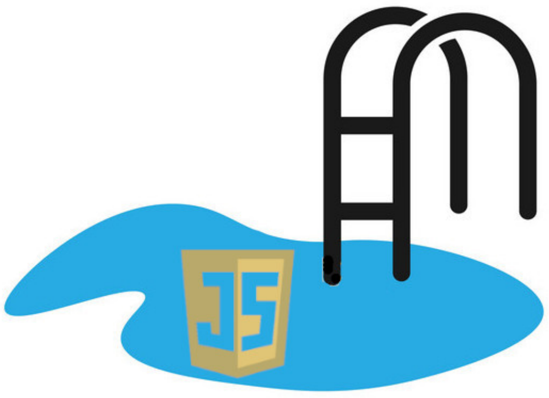
D1 - Pool JS

D-POO-100

JS pool - day 12

API - Getting further



1.01

JS pool - day 12

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.

CRUD is an acronym for Create, Read, Update, and Delete.
They are the four major functions used to interact with database applications.
It sums up the data manipulation functions needed for an application to feel complete.

SQL works similarly, with its *Insert*, *Select*, *Update*, and *Delete* paradigm.

HTTP methods also respect this pattern, and APIs often allow these data manipulations:

NAME	DESCRIPTION	HTTP
Create	Adds one or more new entries	POST/PUT
Read	Retrieves entries that match certain criteria (if there are any)	GET
Update	Changes specific fields in existing entries	POST/PUT/PATCH
Delete	Entirely removes one or more existing entries	DELETE

Throughout this day, do **not** try to edit the given API in the resources file. You will only start it and interact with it.

You are encouraged to take a peek though !

Exercise 00

File to turn in: Nothing to turn in.

Your first task is to set up properly the express form-API.

Download the resources of the day and follow these steps :

1. Open a Terminal (in the bottom window of Webstorm)
2. Ensure the displayed path points to your current day folder, and that node is installed
3. Enter the folder of the API you want to start, and ensure your terminal points to the root of the project
4. Run this command: `npm install`
5. When it is done, run `npm start`, or run directly through your IDE

Green arrow / MAJ + F10 on Webstorm

Exercise 01

File to turn in: ex_01/index.html

You are given a basic HTML form.
Add fields similar to the given “First name” **input field**:

- a “Last name” field
- an “Email” field
- a “Password” field

Their respective label and id must be “lastname”, “email” and “password”.
The password field must display stars (*) when typing.

Do not remove anything, and do not use CSS.

Exercise 01bis

File to turn in: ex_01/ex_01bis.js
Function prototype: *handleSubmit()*

Write a `handleSubmit()` function, called after submitting the form, to validate the content of the form.

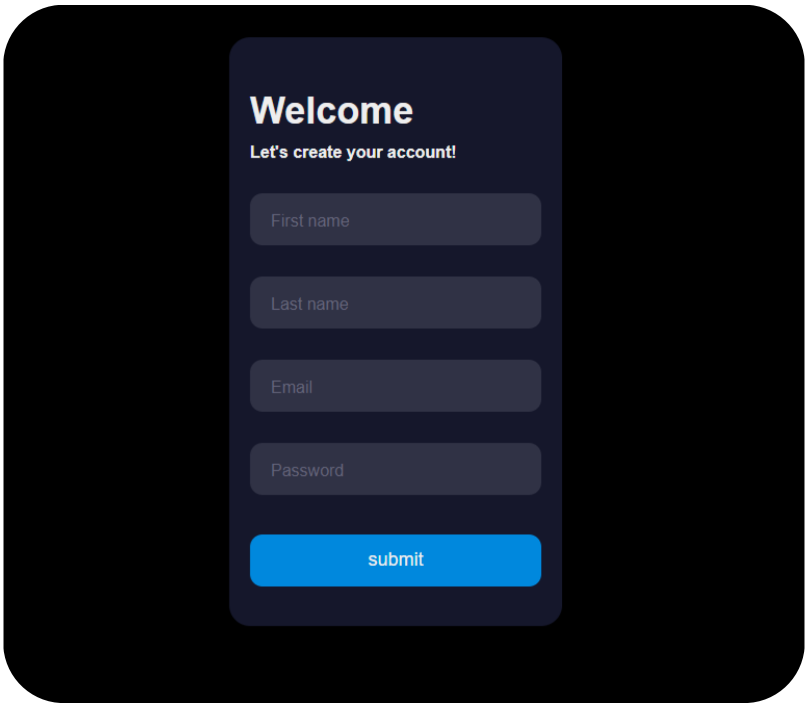
This function must alert:

- when *First Name* or *Last Name* is empty ;
- if the email does not contain at least a "*@_" and a "._*", both surrounded by characters ;
- if the password contains 7 characters or less, or if it contains only letters or numbers.

If an alert is raised, display an error in the *error* paragraph, respectively:

- "First name must not be empty"
- "Last name must not be empty"
- "Email is badly formatted"
- "Password must be at least 8 characters long and contain at least a letter and a number"

You may use **regex** to check if both password and email are valid.
Modify your html so that *handleSubmit()* is called when the form is submitted.



Exercise 01ter

File to turn in: ex_01/ex_01ter.js

We will now send our form to our API.
Your local form-API should be up and running by now.

Using javascript, POST your form to the following route : `localhost:3000/validateForm`
You must include the following information in the **body** of your **request** :

```
{
    "form": {
        "firstname": "first name",
        "lastname": "last name",
        "email": "email@domain.com",
        "password": "f02368945726d5fc2a14eb576f7276c0"
    }
}
```

First name, last name, and email are to be sent the way you received them, but don't forget to encrypt the password using **MD5!**
Specify also in the header that the content you are sending is JSON-formatted:

```
Content-Type:application/json
```

Simply display the response you get from your request in the console.

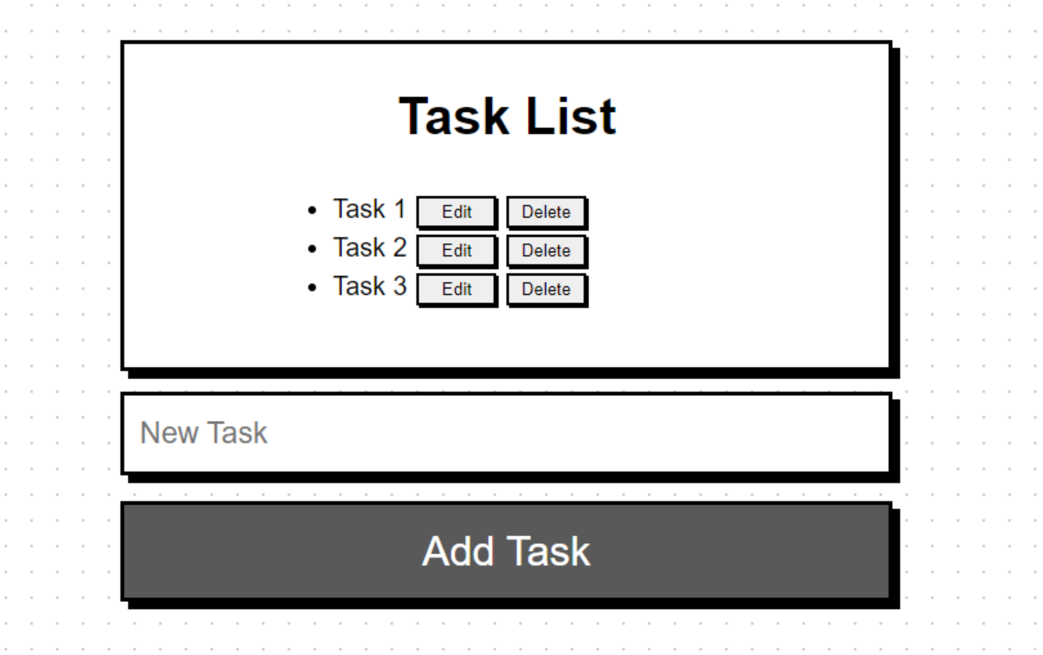
Exercise 02

File to turn in: ex_02/ex_02.js

We will now use the second API provided for the day, the *todo-API*.
Go back to the ex_00 to get it up and running, following the same steps.

Start by reading the *README.md* file.
It contains essential information about how to use the API, including an online documentation.

Your first task is to write a JS script that will **GET** and display every existing task from the API in a list, adding `<li>` element as in the given HTML.



Feel free to adapt the HTML to your needs, keeping the same structure.

You **NEED** to use the provided API.

Exercise 02bis

File to turn in: ex_02/ex_02bis.js

Get the “Add Task” button to work.
When clicked, you must **POST** a new task to the API.

Do not forget to refresh the display afterwards.

Exercise 02ter

File to turn in: ex_02/ex_02ter.js

Allow the user to **UPDATE** any task.
After clicking any “Edit” button, display an alert asking for the new value to be assigned.

Refresh the display afterwards!

Exercise 02quater

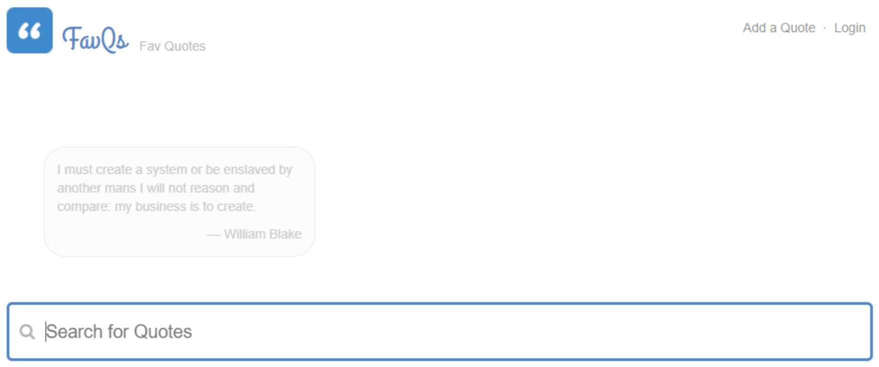
File to turn in: ex_02/ex_02quater.js

Allow the user to **DELETE** any task.
After clicking any “Delete” button, delete the associated task.

Exercise 03

File to turn in: ex_03/ex_03.js

Explore the favorite quotes API and read carefully its documentation.
Then, create a new HTML page that prints the quote of the day.



I must create a system or be enslaved by another mans I will not reason and compare: my business is to create.

— William Blake

Exercise 03bis

File to turn in: ex_03bis/ex_03bis.js

As a preamble, take some chilling time and read some of these quotes until you find one you enjoy.
Using a POST request, add this quote into the favQ API, and display the return code of your request.

Exercise 04

File to turn in: ex_04/ex_04.js

Let's play a bit with Github.
To do so, you will need a **token** to authenticate.
If you already have one, skip this paragraph.
Otherwise, go to github.com/settings/tokens and click "Generate new token".

Save your token once generated, because once you leave that page, you won't be able to see it again.

Get the last 3 commits on today's repository and print their messages in the console.

Exercise 04bis

File to turn in: ex_04bis/ex_04bis.js

Add a selfie in your local `ex04bis` folder.
Write a code that commits and pushes this image onto your repository, with the following message:

```
Javascript added the greatest selfie ever
```

Assess that everything happened correctly with your exercise 03 code by looking at your repository on github.com

Exercise 05

File to turn in: ex_05/documentation.md

Write the documentation of the provided API.
Interact with it, understand it, and write an extensive documentation using the .md format.

Here is the entry point : `story.loscil.fr/path/1`

Feel free to format the documentation as you like, but you are encouraged to take inspiration from the one you've seen until now !