C1 - Javascript

C-COD-120

Javascript Day04

jQuery Day02 + ajax

1.2

EPITECH.

Javascript Day04

repository name: javascript_jquery_2
repository rights: ramassage-tek

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Foreword

Remember that you're coding in jQuery, so you must use the jQuery alternative to do something, and not pure JS.

Exercice 01 (1pt)

File to hand in:

- ex_01/index.html
- ex_01/ex_01.js

Create an input allowing someone to enter a text. Then add it to a bulleted list displayed under the input.

Exercice 02 (3pts)

File to hand in:

- ex_02/index.html
- ex_02/style.css
- ex_02/ex_02.js

Define a few CSS classes:

- A "note" class, putting the border of the element in **blue**
- An "email" class, putting the border of the element in **green**
- A "todo" class, putting the border of the element in **red**

It should now be possible to select one of these 3 types (note, email, todo) when validating the input. You're free on the way you implement this, and you can add more style to your css. However if an input of type "email" is created, the element displayed with the content must possess the class CSS "email". It will work the same way for the 2 other types.

Note: You may need to add some different checks to validate the inputs depending on the type.

Exercice 03 (2pts)

File to hand in:

- ex_03/index.html
- ex_03/style.css
- ex_03/ex_03.js

You will now implement a search in the elements created. As of now, you only need to handle a search by types. You will implement a search by words in the following exercise. If I search for “email”, only the elements being “email” must stay displayed. The others must not be deleted and must be shown again when the search is over (Once again you’re free for the implementation).

Exercice 04 - Bonus (3pts)

File to hand in:

- ex_04/index.html
- ex_04/style.css
- ex_04/ex_04.js

You will upgrade your search possibilities. It should now be possible to search by a word or part of a word contained in the elements. For example, if I search for “rick” and I have an email “patrick@something.com” and a “todo” with the value “Send a mail to Rick”, they both must be displayed. Moreover, you should add the possibility to combine your search with words and types, so in the previous example if I also choose to display only “todo” only the “Send a mail to Rick” should be displayed in the end.

Once again, there should be no element destroyed, and it should be possible to switch from one search to another without the need to reset everything. (For example if I selected “todo” and “rick” in my search but now decided I want all the types, I should not need to erase everything and re-do my selection).

Exercice 05 - Bonus (2pts)

File to hand in:

- ex_05/index.html
- ex_05/style.css
- ex_05/ex_05.js

Let's upgrade your input a little. Make it possible to add tags to a "todo" element. The tags must also be displayed. It should be possible to add multiple tags to an element, even after its creation.

Introduction to AJAX

+ What is AJAX

AJAX is for Asynchronous JAvascript and Xml. It is a set of technologies which enable communication between a client (your web browser) and a server (apache/nginx) without refreshing your web page.

+ Asynchronous and callbacks

The only way you programmed is in a synchronous way. It means, each statement is carried out in the order you specified. The Asynchronous way uses callback. What is it? Simply put, it's a function, called only when a particular event is triggered.

A simple example, you want to make an AJAX request to a laggy server. This request takes time. If it is synchronous, you will wait for the request to complete before doing anything else. It may take time. With asynchronous programming you can tell "execute this function when you will end, even if it takes 30 seconds, it doesn't matter" and you can continue to do your things aside.

+ Some precisions

We will use something one can say is an "old school" way of doing things for server side code. This way to create responses is not used in frameworks (such as Symfony). I'll ask that you work with echo and PHP's JSON encoder. You will see with Symfony that this way is totally out of the picture. Instead of Echoing JSON data, Symfony and number of other frameworks allow you to create objects such as a JsonResponse object that override the practice that I'm going to ask you to use. You are free to go and check these out to see what I am talking about.

Exercice 06 (2pts)

File to hand in:

- ex_06/index.html
- ex_06/style.css
- ex_06/ex_06.js

The PHP script contains an 'echo' of an array containing name as key and your name as value. The whole thing (the array) is encoded with Json. You MUST specify in the header that you are going to communicate JSON data. In the HTML you will create a button. When this button is pressed an AJAX call using a GET method is done to the PHP file. Upon success, you will create a new note containing your name.

Exercice 07 (2pts)

File to hand in:

- ex_07/index.html
- ex_07/style.css
- ex_07/ex_07.js

Using the database given with today's subject, you will add to your file the only element that you trully miss. To do so you will add a button 'Import THE element from the database' that will import the only element from the database that truly matters in your list.

Exercice 08 (2pts)

File to hand in:

- ex_08/index.html
- ex_08/style.css
- ex_08/ex_08.js

Now you will add to your list each element of the table "todo" from the database.
And of course all the tags are linked to the todo elements.
To do so, you will add a button "Import from database" that will import every "todo" elements from the database to your list.

Exercice 09 (3pts)

File to hand in:

- ex_09/index.html
- ex_09/style.css
- ex_09/ex_09.js

You will "dump" your list into the database.
To do so you will add a button to your HTML "Save into database" that will POST the content of your page using an AJAX call.

EXERCICE 10 (3 PTS)

File to hand in:

- ex_10/index.html
- ex_10/style.css
- ex_10/ex_10.js

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

In order to make your interface more ergonomic you should use the promises to display to the user the result of his actions: "Action recorded successfully" or "Communication problem with the server" if the server encounters an error or does not respond after a certain time.

The alert message should only be displayed after the server has processed the user's request

.