T5 - Web Development Seminar

T-WEB-500

Day 10

AJAX

2.0

What is AJAX

AJAX is for **A**synchronous **J**Avascript and **X**ML.
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

So far, you always programmed is in a synchronous way. It means that each statement is carried out in the order you specified.
The **asynchronous** way uses callbacks.
It is 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 the "old school" way of doing things for server side code.
You'll be asked to work with echo and PHP's JSON encoder.

You later in your studies see that this way is totally out of the picture.
Instead of echoing JSON data, many frameworks allow you to create objects such as a JsonResponse object that override the practice that we're going to ask you to use.
Feel free to go and check these out.

Exercise 00

You've Got Mail

First thing first.
Create a skeleton of an html page you will be using this whole day with jQuery.
You need to use the plugin **Bootstrap**.

Bootstrap is going to make notification appear for us.

Include the JavaScript file in your following html page: *bootstrap/js/bootstrap.min.js*.

Feel free to check the documentation to learn how to use it.

Look at this page and use some of the javascripts components in an new javascript file (for instance *js/-main.js*) that you will include in your html page too.

Exercise 01

I said it hundred times and I will say it even more

1. Read the jQuery documentation about **$.ajax()**.
2. Read it twice… Really, I mean it!

Exercise 02 3pts

Let's do some AJAX

Turn in: ex02.html, ex02.php

The PHP script contains an 'echo' of an array containing **name** as key and your name as value.
The whole array must be encoded with Json.

php.net may help you for the encode thing…

You MUST specify in the header that you are going to communicate JSON data.

Your script MUST be in script tags.
It is a tiny code, so you don't need a file for it.

In the HTML code, create a button.
When pressed, an Ajax call using a GET method is done in the PHP file.
Upon success you will create a notification containing your name.
Either way (successful or not), when the Ajax call ends, create a notification too.

EXERCISE 03 — 5PTS

Hello Server

Turn in: ex03.html, ex03.php

In your HTML, add a form containing an input **email**.

When the form is submitted, it sends a POST request to the server containing the serialized data of the form.
In your PHP script, check the email address validity.

In case of an error or if the email address is not valid, change the HTTP status code to 400.

You must display a notification whether the email address is valid or not.

EXERCISE 04 — 2PTS

Get that?!?!

Turn in: ex04.html

Now, load script on the fly with **$.getScript()**.

Read jQuery's documentation.

You must modify the HTML skeleton you created earlier, just remove the line where you included the Bootstrap script.
Create a button **load bootstrap** that loads the Bootstrap file script and launches a notification when pressed.

EXERCISE 05 — 2PTS

Loading…

Turn in: ex05.html

We gave you a file *countries.json*.

We are going to make use of this file.
In your HTML page, create an empty table which contains 2 columns one for the **country-code** and another one for the **name**.

The page mut have a “load countries” button, that fills the table with the data contained in the *countries.json* file.

In order to make this feature you MUST use **$.getJson()** function.

EXERCISE 06 — 3PTS

Turn in: ex06.html, ex06.php, ex06.js

Create an HTML page with 2 fields that sends 2 strings.

The goal is to display if a brand can be added in the database.
The database must be created with *ajax_products.sql*.

You must always have two validation messages displayed, one for each field.
You need to display these validation messages in real-time when you have a modification in fields (dynamic display).

Requirements

- **Type** allows only alphabetical characters and '-' character.
  Its length MUST be between 3 and 10 included.
- **Brand** field allows alphanumeric characters, '-' and '&'.
  Its length MUST be between 2 and 20 included.
- Both fields are case insensitive.
- The request to your database needs to be a GET request, made using Ajax and jQuery in a *ex06.js* file.

Validation messages

- $type: this type does not have enough characters.
- $type: this type has too many characters.
- $type: this type has non-alphabetical characters (different from '-').
- $type: this type doesn't exist in our shop.
- $type: this type exists in databases.
- No type sent yet!
- $brand: this brand does not have enough characters.
- $brand: this brand has too many characters.
- $brand: this brand has invalid characters.
- $brand: this brand already exists in databases.
- $brand: this brand is valid for the type $type.
- No brand sent yet!

These messages MUST be highlighted with red color for an error message and green when it's valid.

EXERCISE 07 — 3PTS

Turn in: ex07.html, ex07.php, ex07.js

Starting from the previous exercise, add 2 fields (*price* and *number*).
The goal is to perform a research and display the result in an array.

Requirements

- **Type** field allows only alphabetical characters and '-'characters.
  Its length MUST be between 3 and 10 included.
- **Brand** field allows only alphanumeric characters, '-'and'&' are valid.
  Its length MUST be between 2 and 20 included.
- **Price** field needs to be protected with only numeric characters and '>', '<' or '='. The length needs to be between 2 and 5. (e.g.: '>100', '<42' or '=42').
- **Number** field only allows positive numbers.
- These fields will all be case insensitive.
- The request MUST be a POST request made with Ajax and jQuery in product.js file.

You have to display one error message if the request doesn't succeed after having clicked on the button.

Display the proper error message among:

- Error ($type): this type does not have enough characters.
- Error ($type): this type has too many characters.
- Error ($type): this type has non-alphabetical characters (different from '-').
- Error ($type): this type doesn't exist in our shop.
- Error ($brand): this brand does not have enough characters.
- Error ($brand): this brand has too much characters.
- Error ($brand): this brand has invalid characters.
- Error $(brand) : this branddoesn'texistindatabases.' * Error($price): this price does not have enough characters.
- Error ($price): this price has too many characters.
- Error ($price): we cannot define a price - string invalid.
- Error ($price): No products found for this price.
- Error ($number): Sorry, we haven't enough stock, we have only $stock in stock.
- Error ($number): It's not a positive number.

If the request is valid, display the product in a 5-column-array (type, brand, price, number, stock) without refreshing the HTML page.

The content of the array must be cleared when a new search is done.
The response MUST not contain HTML but JSON.

EXERCISE 08 — 4PTS

Chit-Chat

Let's create a mini-chat with Ajax.
Your chat will contain 2 text boxes: one for the name, and another one for the message.
You must display the older messages and the sender's name.

Messages must be refreshed every 4 seconds.

Be creative, you are free to use everything you've just seen.