PW 1
EFREI 3A WEB

Louis Cherel - 08/2020

Preamble

Please read **all the instructions**, they are written to be read.

Purpose of these sessions

This Advanced Web course is designed to introduce you to the different technologies that make up the web **front-end**, **browser-side** and **back-end, server-side**. HTML, CSS and Javascript will be the languages whose mysteries you will have to understand.

These three technologies are very broad and represent a field of application that this course **cannot fully cover**. The number one objective of these different sessions is to get you to **find the information on your own**. The coaches are there to support you in this process, and to direct you to the most relevant resources.

You should know that it is possible to learn the web in total autonomy, so if you find it difficult to search for information by yourself, search better, and search mostly **in English**. Indeed, at your level of study, and given the field you are studying in, it becomes counter-productive to continue googling information in French. Even if it is possible to find good resources in your native language, it will be much easier to find the same ones in English, and it will be **much more efficient in the long run**.

The advantage of the web regarding development is that you can:

- check very quickly if something you found on the internet works;
- use the inspector (Right Click > Inspect/Examine on Firefox & Chrome) to examine what your browser has understood of what you have written;
- use this same inspector to modify the code live and see how the page is progressing! (caution: unsaved modifications).

The PWs are all picked up

All PWs are picked up and must be returned no later than 11:55 pm on the evening of the session in the Moodle space provided for this purpose.
Each day that you are late will remove 1 point from your PW score, with a limit of 3 days late, which, after that time, will earn you a score of 0 on your PW.

Only one of the PWs will be graded, but not giving back the others will still be worth 0 to you.

The TPs and the project are to be done in pairs or trinomials, and must be registered with your teacher before the end of the first session.

Conduct of the PWs

You have 5 sessions, divided as follows:

- TP1: Getting back to basics on HTML, CSS and Javascript;
- TP2: First step with Vue.js;
- TP3: Vue.js: components and routing;
- TP4: Discovery of Node.js, interaction with Vue.js;
- TP5: Authentication and BDD with Node.js;
- Project (session 1): discovery of the subject, design of your site;
- Project (session 2): Realization of the project

Recommended resources

So your weapons will be the Mozilla Developer Network (MDN), Stack Overflow, and English research. **Avoid as much as possible** forums (not courses) on sites like OpenClassrooms or Comment Ça Marche, which offer solutions that are often unreliable. W3Schools can be a good resource, but best practices and standards are not their forte. Prefer MDN when possible.

Many blogs are very good, including alsacreations (in French) or css-tricks (in English).

Exercises

Exercise 1 (🕒 5 min read + 🕒 15 min practice)

Preliminary reading required: Using the viewport meta tag to control layout on mobile browsers (MDN)

1) Create a .html file with your prefered editor
    a) Recommended editors (linux, windows, mac) for the Web Programming course are **VSCode** or Sublime Text
2) Create a web page with the basic elements (html, head, body)
3) Give it a title with the tag **<title>**
4) Open your html file with your browser (Open File > …). Your title should appear in the browser tab
5) On your newly created page, right-click (anywhere) and click on **"Examine"** or **"Inspect"**, depending on your browser
6) Find your tag <title> in the interface that appears before your eyes
7) Modify the content of the tag, and see the change made
8) In the head tag, add the meta tag to properly manage mobile devices
9) Add a header, a main and a footer, with ad-hoc tags and in the right place in the page structure
10) Validate that your document is syntactically valid using the online W3C validator
    a) Fix all errors that the validator returns, until the document has exactly 0 errors and 0 warnings

Exercise 2 (🕒 5 min read + 🕒 20 min practice)

Preliminary reading required: Liste complète des balises HTML (MDN)

1) In the header of the page of exercise 1, add a level 1 title with free content

2) In your footer, indicate your name

3) Put a background color to the header of your page, and make it take the whole width of the page

    a) *Be very careful to use either a css file or a css tag, but especially no style="..." attribute !*

4) Add several articles in the <main> content of your page, each with an image, a title and a short text

    a) *Here, it may be relevant to use background-image instead of <img>. This is one of the cases where using the style="" attribute can be a lifesaver*

5) Make sure your items have a consistent style: the same width, same height, with a little space in between. Use flexboxes for this. Each item should be no more than 300px wide on the screen

6) Add a form at the bottom of the main content of your page. This form must contain:
    a) a title text field
    b) a multiline text field content
    c) a text field to contain the url of the image
    d) a send button

7) Each field must contain a placeholder when the user has not entered anything.

8) Validate that your document is syntactically valid using the online W3C validator

    a) Fix all errors that the validator returns, until the document has exactly 0 errors and 0 warnings

Exercise 3 (🕒 5 min read + 🕒 20 min practice)

Preliminary reading required: Build your own function (MDN)

Start with your exercise 2.

1) At the end of the body tag, add a script tag

2) Create an "addArticle" Javascript function that retrieves the values of the title, content and url fields, creates an article and inserts it at the end of the list of articles on your page

    a) You can use the textContent property, but **not** innerHTML

3) Make sure this function is triggered when the user clicks the "Send" button on the form

    a) Note: add the onsubmit="return false" attribute to the form tag to prevent the page from reloading when the user clicks the Send button

4) Make sure that the new articles have the same style as the articles that were present at the initial loading of the page

5) Validate that your document is syntactically valid using the online W3C validator

    a) Fix all errors that the validator returns, until the document has exactly 0 errors and 0 warnings

Exercise 4 (🕒 3 min read + 🕒 5 min practice)

Preliminary reading required: opacity

1) Make sure that your items are all squares, and that the text is visually **in** the image (superimposed on the image)

2) Make sure that the item is very faded in its normal state, and that its opacity increases as the mouse passes over it

3) Use CSS transitions so that the opacity change effect is progressive.

Exercise 5 (🕒 20 min read + 🕒 40 min practice)

Preliminary reading required: keyup event (MDN), event.target (MDN), Node.parentElement (MDN)
Note: we give you little information in an assumed way. You will have to think about what you are doing and interact a lot with your teachers to solve this exercise.

1) Add a "❌" icon (you can use another one if you wish) at the top right of each item

2) Create a deleteArticle(evt) function that deletes the article in which the click event was triggered

3) When the user clicks on the cross, the article under the cursor should disappear thanks to the deleteArticle(evt) function

a) *If you want to pass object "event" to deleteArticle, use syntax onclick="deleteArticle(event)", because "event" will be recognized and replaced by the real object in the parameters of deleteArticle*

4) We want that when the user clicks on Ctrl+Z, the article reappears

   a) Store the HTML Node representing the article and its position in a Javascript list just before deleting it

   b) Create a function that runs every time a user types a key, and display in the console the key that has been typed, as well as the information if the Ctrl key has been pressed

   c) When the Ctrl and Z combination is pressed, return the last deleted element to its original position in the HTML page

Exercise 6 (🕒 10 min read + 🕒 20 min practice)

Preliminary reading required: fetch (MDN)

1) In your page, add a "download articles" button
2) Create a downloadArticles() function that will retrieve the content of https://my-json-server.typicode.com/musinux/fake-json/articles.
3) When the button is clicked, it should trigger downloadArticles()
4) After retrieving the content, this function should add all the retrieved items to the page
5) The user must also be able to delete these items like the others, as well as be able to do Ctrl+Z with them

To go further (Bonus)

Exercise 7 (🕒 10 min read + 🕒 30 min practice)

Preliminary reading required: Anchors and links (W3C), location: hash (MDN), detect hash changes (Stack overflow)

1) In your hand tag, surround the items and the form with a tag containing

2) Add a div containing a few lines about you

3) We will make sure that depending on the URL, either "about" will be displayed, or the list of articles and the form

    a) At the end of a URL, there is the hash part (after the #) that can be read directly by the Javascript code. Make sure to display the hash part of your url in the console. You can add anything after the # and test, for example: TP1.html#j'aimisn'importequoietçamarchequandmême

    b) Design a showPage() function that, depending on the hash part of the url, displays/hides the divs "about" and "articles": if the hash part is #about, display "about", if the hash part is #articles or is empty, display the articles

    c) Add in your footer two links to #about and #articles of your page, and find a way for the showPage() function to be triggered when the URL changes

    d) Make sure that the URL is also checked the first time the page is loaded