Documentation

Introduction

In this project, we developed a website that allows the user to create an account and practice with a desired language. The user is able to create lists of words and make tests in order to facilitate the learning process.

Code structure

HTML

The HTML file has three main containers . A login page, a user profile page and a dashboard.

- In the login page the user can access his/her account or create a new one. Both actions will redirect the user to the user profile.

- In the profile page, there is some information displayed, namely the avatar, the username and last time being active. Also, there is an option to add a new language or click on an existing one.

- When selecting a language, the user will be redirected to the dashboard. This container has all the content regarding the actions the user can carry in relation to the language learning process. The dashboard has child containers : the main-dashboard and the test container.

CSS

The css is organized the following way. First we have the styles regarding The HTML document and the body, followed by other tag styles. Then , the style organization resembles the HTML structure. At the end, we have the 'utils' section, where some general classes used throughout the code are defined. For instance, the hidden or the invisible classes.

JAVASCRIPT

The javascript code is divided into two files. The users.js file entails the code related with local storage, the code defining the underlying data structure, and some other functions which do not directly interact with the DOM. In this file we first encounter the classes used from top to bottom (the one in the top encloses all the other ones and so on). Then we have all those functions used to revive the objects and their methods. Finally we have those functions tagged as 'utils'.

The main.js file is used for event handling , DOM interaction and API interaction. First we encounter the global variables , followed by event listeners. Then we have all the functions interacting with the DOM, and their order tries to resemble a user blueprint .First the ones concerning the login page, then the functions used in the profile, then the ones controlling the logic used in the dashboard and finally the ones needed to build the test. At the end we also have a utilities part, with functions used for the website navigation or in order to display a message to inform the user.

Git strategy

In order to control the versions of our project, we had three branches. The master branch, where already debugged and functional code was being committed and then the Bernat and Suki branches where we did the respective tasks.

Every time we both completed one or more functionalities, we would first discuss it and then we would locally merge our branches in a fourth branch created locally. After completing the merge and with all the conflicts being resolved. We would briefly test if the functionalities work indeed as desired and there isn't any bug as a consequence of the merge. After that, we would merge this branch with the master and rebase the branches 'bernat' and 'suki' so their header points to the master header.

Minimum requirements

The minimum requirements for the project are the following:

- A login frame for the users to log in or create an account

- A user profile frame where the user can click to one of their target languages he or she is learning, or create a new language. Also, some info regarding the user profile is displayed.

- A dashboard frame containing all the different options the user can interact with, in order to learn the desired target language .

In that dashboard the groups created by the user are displayed. We also have the option to create a group of words, and if clicking a given group, the words inside that group will be shown. Again, we have the option to add another word with its corresponding translation. Both groups and words can be either deleted and edited.

- A test option in order for the user to test the different words he or she has introduced.

- The user should be able within the different frames without any problem.

- API. An api will be used for obtaining the language names the user can choose from.

Extras

- Search Box for searching across the groups by name, and also across the wordslists for a specific word.

- Apart from the test, create also a small interacting game based on flashcards where the user must select the right translation for a given word.

- A statistics section where the test punctuation and rate of words included is tracked.

- Extra optional blanks when introducing a new word. For instance, definition and example.

- A third party api providing a translation for the word introduced. This is the only point among the extras we have finally carried out, although only available for a small group of languages (for instance, english, french, spanish, russian).

Lessons

- We have strengthened javaScript vanilla and jquery.
- We have improved our knowledge in data structure.
- HTML structure and element positioning using bootstrap.
- Css: first-of-type selector
- Usage of closest() in javascript.
- Object oriented programming using classes.
- Password encryption using SHA1.
- More confidence gained merging branches and solving conflicts with git.

Issues

- Project design. As it is the first time that we have a team project with no explicit requirements, the design part was even more important than with other projects and turned out to be more time consuming than expected.

- Usage of data attributes to handle user inputs. When handling user inputs, there might be some malicious code written in the input that can cause the application or website to throw an error. We discovered this possibility once we were already handling the user input with that property.

- API . The research done in order to find a suitable api turned out to be more time consuming than expected.

- Using the word name as a key to store the word object. When editing a word, the word object has to be deleted , due to the fact that you can’t edit an object key, as it resembles the identifier . Hence more time processing is spent on first deleting the word and then transferring all the object properties to the new one. The solution would have been to simply use an identifier that is not related to any properties of the object.

- Styles. Positioning elements and giving them width or height was in some cases a challenge. This was probably due to two reasons. First, we do not know what the classes we use and that come from bootstrap are exactly doing. Second, we used too many styles in this project, and hence we lost track in some cases and did not know which properties were affecting the element and which other ones were being overlapped by other ones.

Wireframe

The project design has been carried out with the help of a wireframe made with figma. The design aims to explore the functional dimension, meaning that we describe the elements that appear but not their styling.

The elements with a blueish color are part of the minimum requirements, whereas the red ones would only be implemented in case we had more time available.

The wireframe can be either found in the following link or in the 'wireframe.pdf' document.

LINK
https://www.figma.com/file/TrcmfFxp2G3Lh9BzOZ81jb/Develop-your-project-with-git?node-id=0%3A1