תרגיל 3

Javascript, strings, functions, objects, arrays, DOM basics

הגשה:
נביאים: https://classroom.github.com/a/zPFLBS7Y
תאריך הגשה: 29/01/2023 23:59

שטראוס נשים: https://classroom.github.com/a/WL06Q5DG
תאריך הגשה: 30/01/2023 23:59

שטראוס גברים: https://classroom.github.com/a/7gKvsDpu
תאריך הגשה: 01/02/2023 23:59

סרטון הדגמה:
https://hac.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=8853038e-a551-41b3-80b9-b0f90112051c

The exercise consists in one page that allow recording user information and displaying a list of users. This list will be saved in your JS program (this is not persistent data, when reloading the page, data will be lost).

The page should present various input fields to record the user, and upon completion the user will be added to the list. The list will be displayed below the form inputs. At the beginning the list is empty.
These are the input fields required:

1. **Last Name:** nonempty, should only contain alphabets (a-z)
2. **First Name:** nonempty , should only contain alphabets (a-z)
3. **Email:** nonempty, text field that should contain a valid email address from an Israeli academic institution (*.ac.il). Email should be unique, one cannot record more than one user with same email.
4. **Password:** nonempty, Password field with at least one uppercase letterm(A-Z), one lowercase letter (a-z), one digit, and minimum eight characters.
5. **Confirm Password:** nonempty, Password field that must match the first password field.
6. **Date of Birth:** nonempty, date field that must be a valid date and the user must be at least 18 years old.
7. **Gender:** a choice male/female
8. **Comments:** A text with a maximum of 100 characters, can be empty.

General rule for all exercises: all input should be trimmed before validation! (remove spaces at beginning and end) For example input such as “ “ is considered as empty (“”).

The input form will be presented in 2 steps:

1. First the user will be presented with input for first name, last name and email
2. then upon clicking on a “next” button:
    a. the remaining input elements will be shown

b. first name, last name and email will not be shown.
c. A "previous" button should allow returning to step 1

The UI will appears to the user as a 2 consecutive pages (while it really happens in the same page – you are basically working on the DOM to produce the wanted result).

The next button should first validate the input fields. If the input fields are mistaken it will display relevant error messages and not go to step 2. If the input is valid, it will go to step 2, displaying the remaining input fields and a "save" button to record new user information.

The "save" button will perform validation on remaining input (password, date, gender, comments), and if validated, it will add the user information to the list of users and reset empty all the input elements.

Upon completion, the updated list of users should be displayed with the first part of the input (step 1).

The list of users should appear as a HTML, alphabetically ordered by last name (a-z), where all input should appear (the password should of course appear once), one user per row, all columns labeled at the header of table.

Requirements:

1. all input validation must be implemented in javascript, do not use HTML builtin validation
2. validation errors should be displayed all together, for example if the first name is empty and the email is incorrect, you should display 2 errors messages at once:
   "input foo@bar is an invalid email" and "missing first name"
3. Error messages can be displayed either as a single block all together , for example such as:

   Please correct the following mistake(s):
   1. Input '123 ss' must contain a single word.
   2. Input '-2' must be positive.

   or separately next (below or to the right) to each input. If displayed as a block, error message should include the user input itself "input **xyz** is invalid email", if displayed right below/next to the input the relevant input should be obvious therefore no need to include it in the error message.
4. Beware to reset error messages each time the user clicks on next or save. You should never leave irrelevant error messages on the page.
5. Make sure to use the module pattern to organize your code as modular components, and avoid polluting the global scope. Define const variables when needed (avoid hardcoded strings), use elements caching when possible.

Good luck!