Mandatory assignment – JavaScript – part 2

IMPORTANT:

- All the requirements are mandatory to implement in order to pass.
- You are required to use the knowledge that have been thought at the lecture, from the syllabus book, and further resources that were suggested at the lecture. Your assignment will be evaluated against these resources. Each task has references to some of these resources.
- More information regarding for example failing marks and plagiarism are given in Blackboard, on the "Mandatory assignments" page.

General considerations

Continue to work with the page and topic from the previous assignment/s by adding to it the features/functionality required by the present assignment.

Your implementation should be similar in complexity with the examples given to you in ch. 5-6 of the syllabus book "JavaScript & jQuery" by John Duckett.

Delivery

All the files for your website should be included in **one** folder named:
IIKG1002_mandatoryAssignment3_StudentName /
IDG1011_mandatoryAssignment2_StudentName

This folder should be packaged as **a .zip file** and delivered in Blackboard.

This folder should contain:

- ☐ a JavaScript file: *script.js*
- ☐ a *styles.css* file with the CSS code
- ☐ an *index.html* file with the HTML code
- ☐ a *scriptDesign.pdf* file containing the design of your script
- ☐ a folder with images
- ☐ a *README.pdf* or *README.md* file with documentation

I. Defining a goal and designing the script

IMPORTANT: the design should cover only the new additions requested for the present assignment.

In a separate file – this can be created in any program you like, e.g., Microsoft Word, Google Sketchboard:

a) define the goal of your script,

b) break the goal in a series of tasks (in a ordered list) that have to be performed step-by-step to achieve the goal,
c) sketch out the tasks in a flowchart or in several – you are required to use the shapes presented on p. 23 of the Duckett book

Sources: Ch. 1/a in the book “JavaScript & Jquery. Interactive front-end development.” and lecture notes from week 5 (p. 16 – 21). See also the many other examples of flowcharts throughout the book to help you understand better their usage: pp. 268, 494, 498, 504, 510, 512 …

Delivery: A single .pdf file containing the flowchart drawing, the goal, and tasks.

II. JavaScript implementation

1 General requirements

a) Use an external script file. Do the same for the CSS file.

   Resources:
   For a reminder on how to link JavaScript files, see pp. 47 – 49 and 51 in the syllabus book “JavaScript & Jquery. Interactive front-end development.”

   as well as Lecture notes from week 5, pp. 47 – 50

b) The path to the file should be a relative one (not absolute); this concerns also the paths to images and the CSS file.

   Resources: p. 83 of the syllabus book (for IIKG1002 students) “HTML & CSS. Design and build websites”, by Jon Duckett.

2 Specific requirements

Create **buttons to add and delete** items (objects) created in the previous assignment from a list, such a favorite list, bookmarks list, or a shopping cart.

i.e., If you created a page displaying three food recipes, each of this recipe should have buttons for, for example, adding and removing a recipe to/from a favorite list; similarly to a favorite list you may have a shop cart list, depending on your topic.

<u>Extra challenge, not mandatory:</u> allow the users to update the contents of your page – for example for the recipes example, allow the users to add new recipes, in addition to the three original ones. This will imply that you include a form where the user can include the name for a recipe, a list with ingredients, and a preparation text.

Resources: see the examples related to the “LISTKING” application that was given as example to you throughout the ch. 5 and 6 of the Duckett book.

Detailing requirements for the implementation:

a) Adding items to your list

- Use the **createElement(), append(), prepend(), before(), after(), cloneNode(), remove(), replaceWith()** methods for working with element and text nodes – use the ones appropriate for your use case

  Resources:

  - Lecture notes from week 11 and Flanagan book, pp. 448-450, 15.3.5 Creating, Inserting, and Deleting Nodes
  - An example of **adding** items you may find on p. 285 of Duckett book on JavaScript. Pay attention that the example uses the old methods for adding elements to the DOM. You should use the new ones mentioned above.
    Another example you may find on p. 319. This example is implemented in jQuery. You must use pure JavaScript.

b) For deleting and adding elements to the list you will be using the theory on events from week 12. Use the **addEventListener()** method to handle the click event fired by the user clicking on the buttons.
Resources: Lecture notes from week 12 and Ch. 6 from the Duckett book on JavaScript.

c) When the user presses the **delete button/icon** the respective item should be deleted from the list.
Resources:
See the example on pp. 268--269 of the Duckett book for a similar implementation.

→ you do not need to check weather the browser supports addEventListener() or preventDefault()

→ you do not need to implement for IE5–8 support as explained on p. 264; so the function getTarget() in the example on p.269 will not be necessary, but just use e.target to get the target element

→ see an updated implementation of the exercise on the p. 269 in the lecture notes from week 12, pp. 43–44

Other considerations:

- For the delete/add buttons you are allowed to use the <button> tag instead of <a>. If you choose to use the <button>, please make sure that you implement the “accessibility” aspects presented here:
  https://developer.mozilla.org/en-US/docs/Web/HTML/Element/button#accessibility_concerns

→ specifically: “To give an icon button an accessible name, put text in the <button> element that concisely describes the button's functionality.”

→ When using <button>, you will not be needing to use preventDefault(), as in the case of the <a> tag, if you add the type to be button “The button has no default behavior, and does nothing when pressed by default.” (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/button#attributes)

- You may use icons from the Material design or design your own

  → https://fonts.google.com/icons?selected=Material+Icons

Delivery: Crate a *script.js* file which should be stored in the root folder.

III. Comments

a) refer to the task number the respective code is implementing, e.g., `// Implementation of the task II.2.a`
b) use short comments to explain what your code is doing, as it is done in the many examples provided to you throughout the Duckett syllabus book; you are going to give more details in the report – see the next task

Extra challenge – not mandatory: You may use a tool such as JSDoc (https://jsdoc.app/about-getting-started.html) for writing your comments. See the extensions available in the Visual Studio Code for JSDoc.

IV. Documenting your code

The documentation can be written in

- a **word file** – save it as a README.pdf for submission – or
- a **markdown file** – save it as README.md for submission
  - for a guide on how to write markdown documentation see https://www.markdownguide.org/
  - for markup tools/support in VS Code see the documentation here: https://code.visualstudio.com/docs/getstarted/tips-and-tricks#_open-markdown-preview and here: https://code.visualstudio.com/docs/languages/markdown

Important: Include references to line-numbers or/and code snippets so that I understand to which code you are referring to.

The documentation should include:

a) explanations of your reasoning around the choices you make in your implementation,

b) a presentation of how the theory that you have learned from the syllabus and lectures applies to your specific implementation
c) include references to the lecture notes, syllabus books, and Mozilla Developer (include page numbers)
d) Explain how you applied good practices in writing your code
  - refer to good practices taught in the class – include page numbers from the lecture notes – and
  - refer to the good practices from this guide https://developer.mozilla.org/en-US/docs/MDN/Writing_guidelines/Writing_style_guide/Code_style_guide/JavaScript

Delivery: Either a README.pdf file or a README.md