Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Department of Software Engineering and Automation

Web Programming

Laboratory work #3

Learn JavaScript basics

Author:
Popa Eugeniu
std. gr. FAF-202

Supervisor:
Alexei Șerșun

Chișinău 2023

1 Task

- Copy index.html, style.css and app.js to your repo
- Modify them to build an application for to-do list
- The app has to cover basic needs
  - to add to the list
  - to remove from the list
  - to mark as done
  - see ”done” and ”to-do” lists separately
- The app has to look attractive

2 Results

This laboratory work was developed using HTML, CSS, and JavaScript. There weren’t used any third-party libraries for JavaScript. First of all, the application structure was defined using HTML tags. The application was designed with CSS and the functionality was implemented using JavaScript.

The code is splitted into three parts: selectors, events, and functions.

All the necessary logic for the app is implemented, including:

- adding and removing items from the list
- marking items as done
- filtering the list of items by ”done”, ”to-do”, or ”all”

Also, the app has an attractive design.

Additional features:

- the list is preserved after page refresh, using local storage
- the app implements a notification mechanism
- the item can be edited

3 Code snippets

This is the html document:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Popa Eugeniu, FAF-202</title>
    <link rel="stylesheet" href="style.css" />
    <script src="app.js" defer></script>
  </head>
  <body>
    <header>
      <div id="add-notification" class="add-notification">
        <p>A new task has been added</p>
        <button id="close1" class="notification-close-1">Close</button>
      </div>
      <div id="delete-notification" class="delete-notification">
        <p>A task has been deleted</p>
        <button id="close2" class="notification-close-2">Close</button>
      </div>
      <h1>To-Do List</h1>
      <form id="new-task-form">
        <input
          type="text"
          name="new-task-input"
          id="new-task-input"
          placeholder="What are your plans?"
          maxlength="30" />
        <input
          type="submit"
          id="new-task-submit"
          value="Add task" />
      </form>
    </header>
    <main>
      <section class="task-list">
        <h2>Tasks</h2>
        <div id="tasks"></div>
      </section>
      <div id="filter-container">
        <div class="filter">
          <select id="filter">
            <option value="all" selected="selected">All</option>
            <option value="to-do">To do</option>
            <option value="done">Done</option>
          </select>
        </div>
      </div>
    </main>
  </body>
</html>
```

This is a code snippet of some selectors:

```
    const form = document.querySelector("#new-task-form");
    const input = document.querySelector("#new-task-input");
    const tasks = document.querySelector("#tasks");
    const filter_container = document.querySelector("#filter-container");
    const filter = document.querySelector("#filter");
    const notification1 = document.getElementById("add-notification");
    const notification2 = document.getElementById("delete-notification");
    const closeBtn1 = document.getElementById("close1");
    const closeBtn2 = document.getElementById("close2");
```

Here are declared several variables and assigned the value of specific HTML elements.

This is a code snippet of some event listeners:

```
closeBtn1.addEventListener(”click”, () => {
    notification1.classList.remove(”add−notification−show”);
});

closeBtn2.addEventListener(”click”, () => {
    notification2.classList.remove(”delete−notification−show”);
});
```

These code snippets define event listeners for two close buttons, closeBtn1 and closeBtn2, that are used to hide the notification elements notification1 and notification2, respectively, when they are clicked.

This is a code snippet of creating an item:

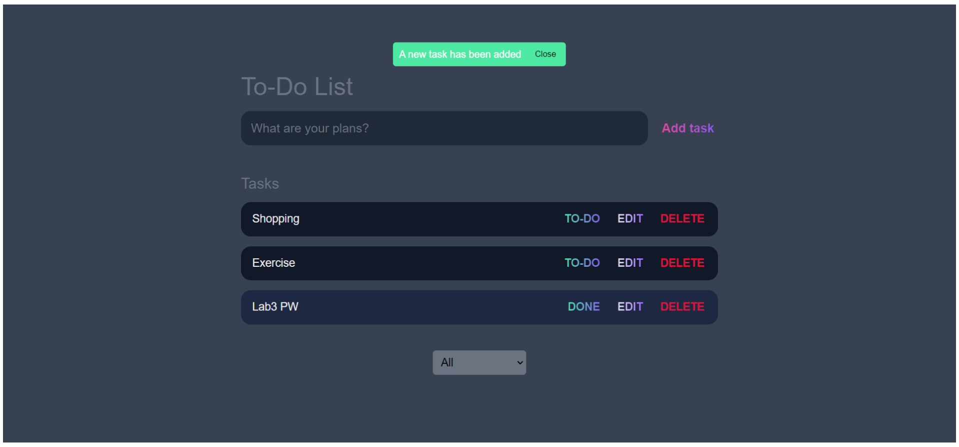
```
const task_el = document.createElement(’div’);
task_el.classList.add(’task’);

const task_content_el = document.createElement(’div’);
task_content_el.classList.add(’content’);

task_el.appendChild(task_content_el);

const task_input_el = document.createElement(’input’);
task_input_el.classList.add(’text’);
task_input_el.type = ’text’;
task_input_el.value = task;
task_input_el.setAttribute(’readonly’, ’readonly’);

task_content_el.appendChild(task_input_el);

const task_actions_el = document.createElement(’div’);
task_actions_el.classList.add(’actions’);

const task_check_el = document.createElement(’button’);
task_check_el.classList.add(’to−do’);
task_check_el.innerText = ’To−do’;

const task_edit_el = document.createElement(’button’);
task_edit_el.classList.add(’edit’);
task_edit_el.innerText = ’Edit’;

const task_delete_el = document.createElement(’button’);
task_delete_el.classList.add(’delete’);
task_delete_el.innerText = ’Delete’;

task_actions_el.appendChild(task_check_el)
task_actions_el.appendChild(task_edit_el);
task_actions_el.appendChild(task_delete_el);

task_el.appendChild(task_actions_el);

tasks.appendChild(task_el);
```

The items are created in the same way when adding a new task, or when starting the app and importing the items from local storage.

This is a code snippet of saving an item to local storage:

```
if (localStorage.getItem('tasks') !== null) {
    storageTasks = JSON.parse(localStorage.getItem('tasks'));
    storageTasks.push(task);
    localStorage.setItem('tasks', JSON.stringify(storageTasks));
} else {
    storageTasks.push(task);
    localStorage.setItem('tasks', JSON.stringify(storageTasks));
}
```

4 Screenshots



A new task has been added

To-Do List

What are your plans?

Tasks

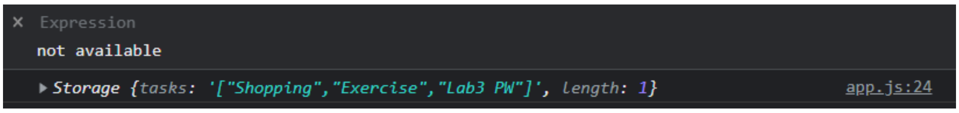
Shopping TO-DO

Exercise TO-DO

Lab3 PW DONE

All

Figure 1: Image of the implementation



```
× Expression
  not available
  ▸Storage {tasks: '["Shopping","Exercise","Lab3 PW"]', length: 1}          app.js:24
```

Figure 2: Image of the local storage

5 Conclusion

In conclusion, this laboratory work has demonstrated a basic implementation of a To-Do app using HTML, CSS and JavaScript, which allows users to add, delete, and update tasks, as well as filter tasks based on their status. The use of local storage ensures that users can access their previous tasks even after closing or refreshing the page. The functions within the app work together to provide a smooth user experience, and with further development, the app can be tailored to meet the specific needs of its users.

6 GitHub

https://github.com/eugencic/utm-pw/tree/main/lab3