01UDFOV/01TXYOV – WEB APPLICATIONS I

MORE JAVASCRIPT IN THE BROWSER

During this fourth lab, you will expand the web app functionality by continuing to manipulate the DOM, and you will re-structure your JavaScript code by practicing object-oriented JavaScript and callbacks use.

EXERCISE 1 – OO JAVASCRIPT

Starting from the outcome of the previous lab, restructure the JavaScript code to take advantage of the object-oriented nature of JavaScript and ES6 classes.

In particular, we suggest that you follow this file structure/class decomposition:

- `main.js` - the script that starts off the app. It just creates an *App* object.
- `app.js` - the App object encapsulates the entire state of the application. It creates a task list, and the next object to represent the filtering functionality.
- `filters.js` – it generates and handles the various views displayed when the user clicks on a specific filter.

You might want to create a *Task* object as well, which represents the property of a single task. The task list, with the desired methods, can be either in a separate *TaskManager* class or in *App*.

Finally, use the `moment.js` library (https://momentjs.com) to manage dates and times: download the library from the website and include it in your project, among the other scripts.

Beware: you cannot use *modules*, at least not yet. Modules work only when the web application is served via a server.

Hint: You can use the solution available for Lab 3 as a starting point, if you prefer: http://github.com/polito-WA1-2020/lab3-javascript-browser

EXERCISE 2 – ADDING TASKS

Update the web application developed so far to allow users to *dynamically* and *interactively* insert new tasks.

To do this, use the Bootstrap's Modal component: when the user clicks on the '+' button, open a modal and ask for a new task (with all its properties) by filling a form. By submitting the form, add the newly inserted task to the JavaScript tasks list.

Beware: form fields should be validated, to reinforce mandatory field and to avoid having not admitted values.

Hints:

1. *The documentation of the Modal component can be found at: https://getbootstrap.com/docs/4.4/components/modal/*
2. *HTML5 defines different properties to help with validation: https://html.spec.whatwg.org/dev/forms.html#client-side-form-validation*

OPTIONAL EXERCISE – PROJECTS

Make the "projects" selection work. In particular, you should generate the list of projects (in the left sidebar) by *reading* the task list in JavaScript. Clicking on a project name will update the content of the main area, as it happens with filters.

EXERCISE 3 – EDIT EXISTING TASKS

Add the edit functionality to your web application. Each task (in the main area, on the right) should have an “edit” button, now. By clicking on that button, the same modal used in Exercise 2 opens, pre-filled with the task properties. On submitting the form contained in the modal, the task is updated with the new values.