BOM

DEADLINE: 05/06/2021

FOLDER STRUCTURE

FL16_HW15/ [*]
- task [**]
- homework/ [*]
  - index.html [*]
  - styles.css [*]
  - app.js [*]
  - .eslintrc.js

[*] - required
[**] - restricted (no need to submit)

TASK

Implement a simple “Twitter – like” application.
NOTE: All styles are up to you (don’t waste much time on design).

When the application is loaded I can see the section with tweets (it should be empty on first load). In the top-center, I can see the button which allows me to add new tweet. Each tweet could be edited, liked and removed. When I press the "Remove" button then the tweet should be removed from the list. When I click on the tweet then application provides me possibility to edit the tweet. When I click ‘Like’ button the tweet should be marked as liked (you can do it the way you prefer by adding icon or text ‘liked’ or changing the text of the button to ‘unlike’). When I press the "Like" button on tweet which is already marked as liked – the tweet is unmarked as liked. When there are liked tweets on the page, I can see a button ‘Go to liked’ on the top of the page near the ‘Add tweet’ button’ it should not be shown when there are no liked tweets. When I press ‘Go to liked’ I can see the page that contains only liked tweets.

Application consists of four pages:

- Main page
- Add tweet
- Liked tweets
- Edit appropriate tweet.

The Main page shows us list of tweets. Features:

- Add new tweet

- Edit existing tweet
- Mark tweet as liked
- Unlike tweets
- Remove tweet from list
- Show error alert
- Show liked alert

Upon clicking the "Add tweet" button system should redirect to the "Add Tweet" page.

The “Add tweet” (URL hash should have **#/add** suffix) page shows empty input field, "Save " and "Cancel" buttons. Tweet can be added when it's not empty and it is not a duplicate. The tweet length is up to 140 symbols. If it is duplicate system should show custom styled alert with text “Error**!** You can't tweet about that”. If item was added, the application redirects user to main page.

Upon clicking on a tweet – the system should redirect to the "Edit tweet" page (The URL hash should consist of **#/edit/:item_id** suffix). The “Edit tweet” shows input field with the appropriate text of the tweet user is editing, "Save " and "Cancel" buttons. Item can be edited and saved when text is not empty and it is not a duplicate. If it is duplicate system should show alert with text “Error**!** You can't tweet about that”. After successful modification, application redirects to main page. And the edited item should be located in the same position as it was before the editing.

Upon clicking the "Go to liked" button system should redirect to the "Liked Tweets" page (URL hash should have **#/liked** suffix). There are only liked tweets on this page. There should be a back button which allows me to navigate back to main page.

"Cancel" button only redirects user to main page in both cases – adding new tweet and editing existing one.

“Like/unlike” button. Upon liking tweet, the tweet becomes marked as liked an alert should be shown ‘Hooray! You liked tweet with id ${id}!’. Upon pressing ‘Like’ on already liked tweet – the tweet is unliked, it is unmarked, and the alert is shown ‘Sorry you no longer like tweet with id ${id}’.

All alerts should disappear(hide) after 2 sec.

It will be really nice if you mark liked tweets, so they differ from not liked.

Pay attention:

- browser *back/forward* buttons must work correctly
- you can edit *app.js* and *styles.css* files but NOT index.html
- all DOM manipulations must be done in *app.js* file
- After refreshing the page all previously added tweets should be shown. (Please use localStorage for storing data)

RESTRICTIONS

- Editing index.html is forbidden
- You don’t need to use any frameworks or libs for routing, just pure JS
- You don’t need to spend much time on implementing design of the application

BEFORE SUBMIT

- Remove all unnecessary files that you might have included by mistake
- Verify that all functionality is implemented according to requirements
- Make sure you code is well-formatted, and validated via validator (w3org Markup Validation Service)
- Add comments if the code is difficult to understand
- Fix warnings/errors in the browser console
- Verify that the name of the folders and files meet the requirements
- Make sure there are no errors/warnings in the browser console
- Run the linter and fix all warnings and errors.

SUBMIT

- The folder should be uploaded to GitLab repository 'FL-16' into master branch

USEFUL LINKS

- https://developer.mozilla.org/en-US/docs/Web/API/History_API
- https://developer.mozilla.org/en-US/docs/Web/API/URL
- https://developer.mozilla.org/en-US/docs/Web/API/Window/popstate_event
- https://javascript.info/settimeout-setinterval
- https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage