A03 - JavaScript SPA

This examination will focus on building a single page application with chat integration against a web socket server. The backend (server-side code) of this assignment will be given and your job is to write the client-side code.

In this examination assignment you are supposed to build what we call a "Personal Web Desktop" (PWD). First of all, have a look at this recording to get a better view of the assignment.

Demo - Personal Web Desktop (PWD) (https://youtu.be/zHFjfoUWONY).

Grading

This assignment is graded as F, Fx, E-A and it is worth 3.5 credit/hp.

Preconditions

You have access to the course environment on GitLab.

You have knowledge of HTML and CSS.

You have knowledge of JavaScript and the browser environment.

Prepare

The lectures in part 3 contains details which helps to prepare for this assignment.

The **example repo (https://gitlab.lnu.se/1dv525/content/example)** contains examples on how to use the DOM and events within the browser. It contains examples for working with a builder and linters. It also contains examples on code structure for:

- Drag and drop
- Web sockets
- Local storage

Requirements

The requriements are divided into a Startup section and a couple of Feature sections (F1, F2 and so on).

All feature sections are mandatory, except those marked with "(OPTIONAL)" which can be implemented to reach a higher grade.

Startup

These are the requirements to fullfill the assignment.

1. You have a git repo on GitLab named "a03-spa", available below your lnu student acronym in the course at GitLab. Start of by cloning the repo to your computer ("xxx" is your lnu-

username):

```
# Using ssh
git clone git@gitlab.lnu.se:1dv525/student/xxx/a03-spa.git
```

1. The repo is prepared as a Snowpack App using the template `@snowpack/app-template-blank`. Review the repo `README.md` to see what scripts are available.

F1: PWD Functional requirements

Please check that your application meets the requirements below before submitting your final version.

1. The application should be a single page application.
2. The user shall be able to open multiple windows (not browser windows/tabs but custom windows created using the DOM) within the application.
3. The user shall be able to drag and move the windows inside the PWD.
4. The user shall be able to open new windows of the desired application by clicking or double clicking an icon at the desktop.
5. The icon used to close the window should be represented in the upper bar of the window.
6. Windows should get focus when clicked/dragged.
7. The window with focus shall be on top of all other windows.

The following three window applications should at least be included in the desktop application:

- A memory game.

- A chat connected to a central chat channel using websockets.
- One, by you, designed and decided application (perhaps the quiz or another application).

F2: PWD Non functional requirements

1. A complete git commit history should be present for assessment. For this assignment somewhere between 30 and 200 commits is normal
2. The code standard standard.js should be followed. Use the development environment you learnt earlier.
3. All exported functions, modules and classes should be commented using JSDoc.
4. The linters should pass without notices when running `npm test`.
5. The code shall be organized in appropriate ES modules, at least four (4).
6. The application shall be visually appealing.

F3: Memory game window application

Read for a **description of the memory application (https://github.com/CS-LNU-Learning-Objects/exercise-memory-game)**.

These are the requirements for the Memory application that should exists as a window application in the PWD.

1. The user should be able to open and play multiple memory games simultaneously.
2. The user should be able to play the game using only the keyboard (accessability).
3. One, by you decided, extended feature for the game.

F4: Chat window application

Read for a **description of the chat application (https://gitlab.lnu.se/1dv525/template/a03-spa/-/blob/master/README_chat.md)**.

The chat application should exists as a window application in the PWD. The chat application shall be connected to other students chats via a web socket-server.

1. The user should be able to have several chat applications running at the same time.
2. When the user opens the application for the first time the user should be asked to write his/her username.
3. The username should remain the same the next time the user starts a chat application or the PWD is restarted.
4. The user should be able to send chat messages using a textarea.
5. The user should be able to see at least the 20 latest messages since the chat applications was opened.
6. One, by you decided, extended feature.

F5: An additional window application

You should add one additional window application to your PWD. It should be developed by yourself and it can for example be the Quiz application or another application you choose to develop.

F6: An enhanced chat application (OPTIONAL)

This is an optional feature that might help you to a higher grade.

Fulfill some (at least three) requirements to enhace your chat application. Here are suggestions and feel free to add your own enhancements.

- Ability to choose which channel to listen to.
- Caching message history.
- Added support for emojis.
- Added support for writing code.
- Ability to change username.
- Encrypted messages on a special channel to allow secret communication.
- Added functionality to the "chat protocol". Discuss with others in the course and agree upon added functionality to add to the sent messages.
- Use the messages to play memory against an opponent. Perferably using a seperate channel.

F7: Additional enhancements (OPTIONAL)

This is an optional feature that might help you to a higher grade. You need to have implemented F6 to be able to solve this requirement.

You may implement and document additional enhancements to your PDW to make it more personal, flexible and usable. You may define these requirements on your, document them and explain how they works.

Grading

These are a few notes on how your submission is graded.

Grade E, D

Fulfill all mandatory features and requirements to a satisfactory level where they work as expected.

Document all your features in the report page and present your solution with a proper video.

Grade D, C

As above and your application, documentation and presentation has a high level of quality.

The outcome is visual appealing and has a high level of usability.

Grade B, A

Your application, documentation and presentation has an excellent level of quality, usability and fulfillment of all the features and their requirements.

You have fulfilled at least one of the optional features, but most likely both and you have produced an outcome to an excellent level with no issues.

Presentation

You are to record a video presentation of your project where you show how your project implements the features and fullfills the requirements. You should also present a few highlights from your source code and elaborate on why you feel the code is good/bad.

The video scene should contain a cam where your face is visible. You should also present yourself using your student id so that it is visible in the cam.

The tool OBS is a free and opensource tool you can use to record the presentation video.

The video should be 5-7 minutes.

Submission

This is how to submit this assignment.

1. Add a tag `v1.0.0` to your repo when you are done. If you make updates then add another tag like `v1.0.1` or `v1.1.0` and so on.

2. Ensure you have committed and pushed all your changes, including the tags, to GitLab.

3. Create an issue on the repo and answer the following questions in it. Write freely with 15 to 30 sentences of text.
    1. For each feature (F1, F2 and so on) you have implemented, write a 5-10 sentences text about it and let the teacher know how you solved it. This is essential. There is no grade if you fail to write about the features.
    2. Write a concluding text, 5-10 sentences, about your implementation of the project. Write about hard/easy, troubles/solutions, and so on, things you encountered and thought of during the project and its implementation.
    3. What is your TIL for this course part?
4. Write a final summary of your thoughts of this course. Did you enjoy it or not, did you learn anything/something/a lot. Let the teacher know your thoughts of the course and grade it by a scale 1-10 where 10 is "quite good course" and 1 is "not so good".
    1. **What is your overall TIL for this course?**
5. When you are done, assign the issue to the teacher, to show that you are ready for grading.

← Previous **L13 - JavaScript Frameworks** **(/courses/introduction-to-web-programming/30-part-3/04-L13-js-frameworks)**

Next **Part 2 - Client-side JavaScript** → **(/courses/introduction-to-web-programming/20-part-2)**