A02 - Code JavaScript with browser

You are to code JavaScript in the browser, utilizing various ways of structuring your code. You may use plain vanilla JavaScript constructs such as the JavaScript Module design pattern and you may use ES6 Classes and modules, you decide the structure of your code.

The aim is to get aqauinted how to organise your code and to learn the basics of integrating with the browser as development environment.

Grading

This assignment is graded as Fail (U) or Pass (G) and it is worth 3 credit/hp.

Preconditions

You have access to the course environment on GitLab.

You have basic knowledge in HTML and CSS.

You have knowledge in JavaScript and the browser environment.

Prepare

The lectures 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 also contains examples on code structure for:

- ES6 Classes
- ES Modules
- Snowpack builder

The example repo contains a `package.json` **(https://gitlab.lnu.se/1dv525/content/example/-/blob/master/package.json)** with all details on the linters and minifiers you can use.

Get going

Get going with the assignment.

1. You have a git repo on GitLab named "a02-javascript", 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/a02-javascript.git
cd a02-javascript
```

2. Create a Snowpack App in the root of the repo using the following `npx` command. We use `--force` since the directory already exists and till overwrite the `README.md` file.

```
npx create-snowpack-app . --template @snowpack/app-template-blank --force
```

3. The following commands shall work with your application.

- `npm run build` shall build the app.
- `npm start` shall start the Snowpack development server with the application running.
- `npm test` shall execute your test suite with the static code validators.

4. You are strongly adviced to use linters to do static code validation for your code. You may for example use the setup available in the repo "**Development tools (https://gitlab.lnu.se/1dv525/template/development-tools)**". The tools are configured to work in a Snowpack environment.

    - `npm test` shall execute your test suite with the static code validators.

Requirements

In this assignment, you will use asynchronous communication to create a quiz application. All questions are served to your application using HTTP in a **RESTful (https://en.wikipedia.org/wiki/Representational_state_transfer)** way, and your application should send the users answers back to the server for validation.

The backend (server-side code) of this assignment is given and already published online, and your assignment is to write the client-side code that communicates against that server.

The assignment

You should create a single page client application in which the user can answer, by the server given, quiz-questions. The user must do this within a specific time frame. If the user provides the correct answer, the application should present the next question to the user. If the user provides the wrong answer or do not answer in time, the quiz is over and a high score list is presented to the user.

You are responsible for the presentation of the questions (retrieved from the server), the handling of the client application logic and the user interface. The user shouldn't have any problem understanding the UI. Keep it simple; keep it beautiful.

At the start of the game, the user should be able to write a nickname she/he wants in the quiz game. The game must have a timer that gives the user a maximum of 10 seconds to answer each question. If the user doesn't answer during the time or the user gives a false answer the game is over and the user should be able to start over.

If the user answers all of the questions correctly, the game is over and the nickname is displayed with a proper "victory message" and showing the time it took to complete the game.

If the user answers all of the questions correctly, the game should save that user:s total time and present it in a high-score list showing the five quickest tries. The high-score will be saved in the browsers Web Storage.

The questions and answers

The questions will be public to the client application through a RESTful Web API. The first question (starting point of the application) is at the URL:

- **https://courselab.lnu.se/quiz/question/1 (https://courselab.lnu.se/quiz/question/1)**

The response of the API will tell you what kind of question you should show to the user and where to send the answers. In other word, you have to analyze the response from the server to know how to display the questions and how to send new requests for answering the questions. Hyperlinks are provided by the server response in a RESTful way. The server responses will also give clues about what HTTP-methods to use and how to send the answers back.

The server will put out two different types of questions. Simple text questions and questions with alternatives where the user should answer with the right key "alt1", "alt2" etc. You can watch the server responses and decide what is what. Questions with alternatives should be presented with html radio buttons. The user should not need to write anything at these questions.

The last question answered will not return a new link to a new question and that means that the quiz is over.

Non functional requirements

The client application must be written in vanilla javascript using ES modules and/or ES6 classes to create well structured and organised code. The application should be well documented. The application is comfortable for the end-user to understand and use.

Documentation

Use the `README.md` to provide some documentation of your application.

1. Add a representative image for the assignment and start with a short description of your solution.
2. Explain how a user can download and start your game.
3. Shortly explain the rules of the game, so the user knows how to play it.
4. Explain if you are using any linters to validate the code behind the game.

Hints

Before you start writing code think about:

- How to present the question the user should answer?
- How to get the answer from the user and how to send it back to the server? **Postman (https://www.postman.com/)** or **Advanced REST Client (https://install.advancedrestclient.com/install)** are great tools if you want to play around with HTTP-calls and do tests without writing code.
- Be sure to tell the server that the POST-request is a "application/json"-request by setting the "Content-type" of the HTTP-header (check up **fetch api (https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API)**)
- The API (Server part) will make use of HTTP. Make sure to use the correct verb (POST or GET), headers and look at the status codes. (200, 400...)

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. Explain how the scope works in JavaScript, how hoisting affects it and how a closure works.
    2. Describe how you opted to organise your JavaScript code in the assignment and your observations and learnings from your code structure.
    3. How do you feel about asynchronous programming in JavaScript, can you relate it to some other programming you done previously?
    4. Elaborate on the lint (test) and build sequence of your code. Do you see pros and cons and can you relate it other development work you have done?
    5. What is your TIL for this course part?
4. When you are done, assign the issue to the teacher, to show that you are ready for grading.

← Previous **Exercise: BartBoard demonstration** **(/courses/introduction-to-web-programming/20-part-2/10-Exercise_Bart-board)**

Next **Guide** → **(/courses/introduction-to-web-programming/80-guide)**