# Challenge No.6 - Specification

Events & Introduction to Objects in JavaScript

**Description**

After the presentation of the challenge and receiving the starter files, each student is required to complete the assignment within the given deadline. The solution should be uploaded to GitLab, with the assessor added as a maintainer to the repository. Additionally, the student is instructed to add the solution link to the learning platform.

## Level 1: Beginner

Student should know JavaScript variables, functions and event listeners to be used in DOM manipulation.
Student should know how to get and set values of HTML elements using methods like document.getElementById().

## Level 2: Intermediate

Student should know JavaScript variables, functions and event listeners to be used in DOM manipulation.
Student should know how to get and set values of HTML elements using methods like document.getElementById().
Student should know how setTimeout() and setInterval() work for handling time-based functions.
Student should know how to dynamically update DOM elements through JavaScript conditions.
Student should understand JSON structure for storing and retrieving data from localStorage.

## Level 3: Advanced

Student should know JavaScript variables, functions and event listeners to be used in DOM manipulation.
Student should know how to get and set values of HTML elements using methods like document.getElementById().
Student should know how setTimeout() and setInterval() work for handling time-based functions.
Student should know how to dynamically update DOM elements through JavaScript conditions.
Student should understand JSON structure for storing and retrieving data from localStorage.
Student should know how to retrieve information from localStorage and display it in the DOM, as well as save certain information in localStorage.

**Description: Study Timer**

Create a personalized study timer with a visual progress bar, where users can set up personalized study sessions with breaks. The application should visually display the progress of each session (study time + break) using a progress bar.

**Exercise 01**

Create a function to get from the user input the study and break duration in minutes. Save the study duration and break duration on local storage. Persist the study and break duration in the local storage upon refresh.

**Exercise 02**

Use JavaScript timers (setTimeout and setInterval) to count down the session and break durations. Show the progress via a dynamic progress bar that fills up as the time for the study session + break elapses.

**Exercise 03**
Notify the user via a simple alert when it's time to take a break and when to resume studying.

**Exercise 04: Bonus**
Display a log of completed study sessions and breaks including their durations.
Hint: Use LocalStorage.

**Evaluation system**

| ***Criteria*** | Excellent 2.5p | Proficient 2.0p | Good 1.5p | Fair 1.0p | Poor 0.5p |
|---|---|---|---|---|---|
| Solution Correctness | *The solution is flawless, meeting all requirements and functionalities.* | *The solution is correct for the most part, with minor errors that do not significantly impact the functionality.* | *The solution is mostly correct, but there are some major errors affecting the overall functionality.* | *The solution has several major errors, making it partially functional, but shows a basic understanding of the problem.* | *The solution is incorrect, incomplete and lacks understanding of the problem.* |
| Code Quality | *Code is exceptionally well-organized, follows best practices, and demonstrates a deep understanding of all required concepts.* | *Code is mostly clear and well-organized and follows good practices, with only minor improvements needed.* | *Code is mostly organized, but there are notable areas that could be improved for better readability and maintainability.* | *Code lacks organization, readability, and structure, and could be significantly improved.* | *Code is messy, unreadable, poorly organized, and does not adhere to coding standards.* |
| Adherence to Specifications | *The solution precisely follows all specified challenge requirements.* | *The solution mostly adheres to the specifications, with only minor deviations or oversights.* | *The solution deviates from specifications in some aspects but still fulfills the main requirements.* | *The solution deviates significantly from the specifications, but some elements are implemented correctly.* | *The solution disregards most or all of the specified requirements.* |
| Documentation, Comments, Cleanliness | *The code is well-documente d, clear and exceptionally clean. Comments explain the logic behind any complex parts.* | *Good documentation and comments. Code is generally clean with a few areas for improvement.* | *Basic documentation and comments, with room for improvement in clarity. Code cleanliness is acceptable but needs* | *Limited documentation and comments. Code lacks clarity and cleanliness and is not well-organized.* | *No documentation or comments. Code is disorganized and challenging to understand.* |

| | | | | | |
|---|---|---|---|---|---|
| | | | *improvement.* | | |

**Deadline**
1 week after its presentation, at 23:59 (end of the day).

**Assessment Rules**

- ❖ Fair Assessment: ethical considerations
  - ➢ Assessors should ensure that the assessments are conducted in a fair and ethical manner, respecting the principles of academic integrity and honesty.
- ❖ Reliability and Validity: enabling consistency
  - ➢ Assessments should be consistent and reliable, meaning that they yield consistent results when applied repeatedly to the same task or performance.
  - ➢ Assessments must accurately gauge the knowledge, skills, or abilities they are intended to evaluate, ensuring their validity as indicators.
- ❖ Feedback: as a method for continuous improvement
  - ➢ Assessors should offer constructive feedback that identifies strengths and areas for improvement. The Feedback should be specific and actionable, it should include thought provoking guides and should challenge the student to become better at a specific task.
- ❖ Late Submission Policy: assessing assignments after their deadline
  - ➢ Students should be allowed a grace period of 3 days (72 hours) to make a late submission on any assignment, with the notice that they will be deducted 20% from the total possible points.
- ❖ Plagiarism Policy: assessing assignments with matching solutions
  - ➢ In the event of suspected plagiarism, the assessor is required to promptly collaborate with the Student Experience Coordinator/Team as the initial step. Together, they will draft a notice to remind students of the strict prohibition against plagiarism, with potential repercussions for recurrent violations. The following actions will be considered in cases of repeated plagiarism:
    - ■ If a submitted solution exhibits substantial similarity, exceeding 60%, with another student's work (individually or within a group), the respective challenge will incur a 50% reduction from the maximum points attainable.
    - ■ In cases where a solution is identified as more than 90% identical to another student's work (individually or within a group), the challenge in question will receive a score of 0 points.
  - ➢ The use of generative AI is encouraged as a learning tool in our educational programs; however, students must engage with the material and contribute with original thought. Reliance on AI for complete content generation is strictly prohibited and will result in point deductions, official warning, or other academic penalties.
  - ➢ Upon completion of the assessment process for each challenge/project, the assessor is tasked with selecting the most complete, optimal, and creative solution and to showcase it by publishing it on the platform together with the assessment results
- ❖ Timeliness: timeframe for delivering results and feedback to students
  - ➢ Feedback on challenges should be provided within 7 days after the deadline

has passed.