Assignment 3: Math Quiz

Coding Guidelines

- The assessment will be tested using **Chrome** web browser. Before submitting, ensure that all links and images are working by opening your index.html file in a Private Browsing window (incognito mode).
- In addition to implementing the required functionality, grading is based on appropriate usage of coding conventions demonstrated in class, clear organisation of code, clarity of variable naming, etc.
- You must use let/const for variable declarations, and arrow function syntax
- You are **not permitted** to use 3rd party HTML, CSS, or Javascript libraries

Starter Code

- You must use the provided HTML and CSS files
- You are **not permitted** to modify any of the HTML or CSS.

Submission Checklist:

- ❑ Create a folder called **A3-firstname-username**. Replace firstname and username with your name and username.
- ❑ Place all required files in the folder (Index.html, styles.css, app.js)
- ❑ Create a zip file of the folder, and rename the zip file to **A3-firstname-username.zip**. Replace firstname and username with your name and username.
- ❑ Submit your zip file to the dropbox by the specified due date.

Academic Integrity

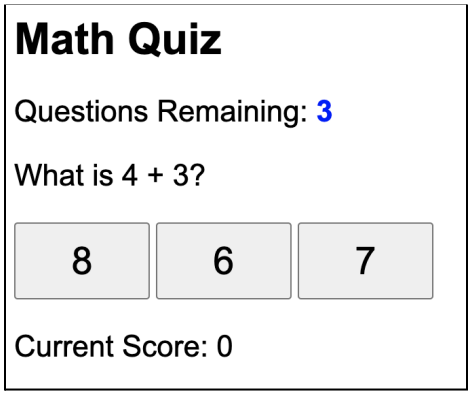
- This is an individual assessment
- You may use the course materials
- You may use the Internet to look up code syntax. However, following tutorials is not permitted
- You are responsible for familiarising yourself with the college's Academic Integrity Policy.
- Actions that constitute academic dishonesty include, but are not limited to:
    - Reposting any part of the assessment to online forums or homework help websites
    - Contract plagiarism: Purchasing a solution, or completing a solution for compensation
    - Sharing or receiving source code, references, or assistance from others

App Description

Using Javascript, create a quiz application that tests the user on their general knowledge of math.

1. The quiz consists of 3 **randomly generated** math questions, displayed one at a time:

Example UI



2. For each question, the quiz displays 3 possible answer choices, as **buttons**.
3. Answer choices are displayed in random order
4. The user selects an answer by tapping on the appropriate button. The app must evaluate the selected answer, and output the results to the screen.
5. For every correctly answered question, the user receives 100 points. Incorrect questions receive 0 points.
6. The game is over when all questions are answered.

Technical Requirements:

1. Generating Questions and Answers

The app must randomly generate a question, one at a time.

A question is generated when:

- The user starts the game (the first time the page loads)
- After the previous question's results have been evaluated an a new question is needed

The *app* will always ask the user to calculate the *sum* of two numbers (a+b). Each numbers must be a randomly generated value between 1 and 10, inclusive.

A question must be *programmatically generated* at the time it is needed. You are **not permitted** to use an array of pre-existing questions, or use any statically coded questions.

In addition to generating a question, the app must *programmatically* generate 3 possible answers for the question.

The possible answer choices always consist of:

a. The correct answer - 1
b. The correct answer
c. The correct answer + 1

For example, suppose the question is: What is 3+4? The possible answer choices would be 6, 7, or 8.

2. Displaying the Answer Choices and Detecting Clicks

The app must *programmatically* populate the text of the <button> elements with the possible answer choices. Specifically, you must use the document.querySelectorAll() function to retrieve the buttons, and then iterate through the results to programmatically populate update the button text with possible answers.

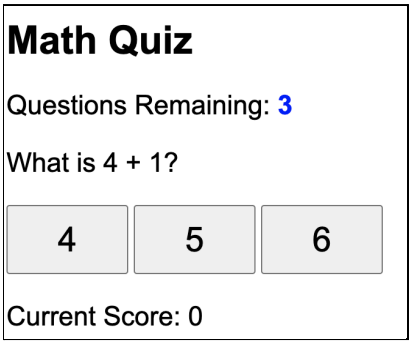
The order of the answer choices must be random. HINT: use the shuffle() function provided in the starter code.

You must use the *event delegation* pattern to detect when a button is clicked. You are **not** allowed to attach individual click handlers to each button.

User Interface Behaviour

1. Starting the Game

When the game starts, generate the first question and possible answer choices. Populate the UI with the appropriate information.



At all times, the app must display:

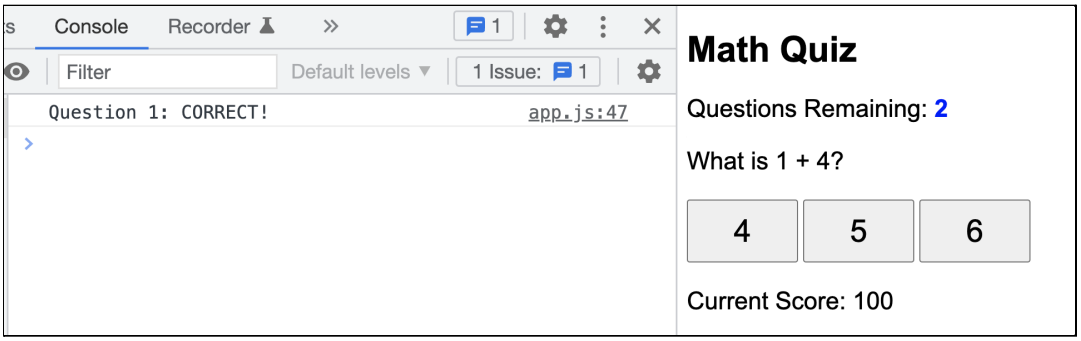
- The current question
- The current possible answer choices
- The number of questions remaining
- The current score

2. Evaluating the Answer

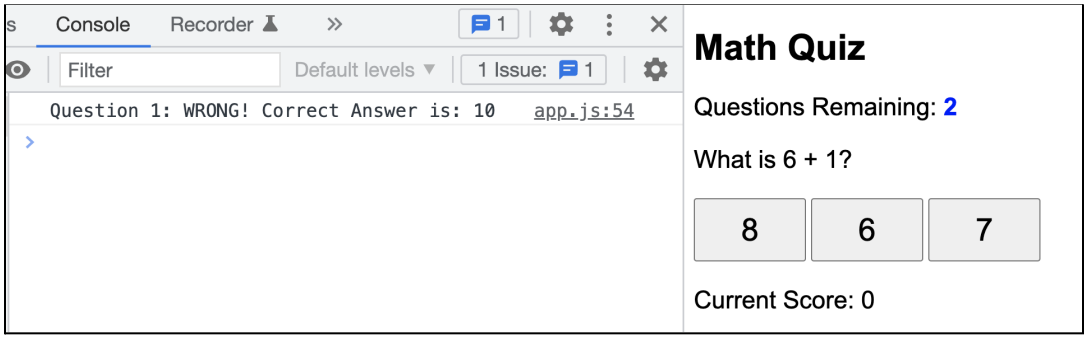
After the user selects an answer, the application should determine if the user selected the correct answer. Based on the answer provided:

- Output a message to the console, indicating the results of the question (see screenshots). The message must display the question number and results of the question. If the selected answer was incorrect, then also show the correct answer.
- Decrease the number of questions remaining
- If applicable, increase the score
- Update the UI to show a new question.

Example 1:Console output when correct answer provided



Example 2; Console output when wrong answer provided

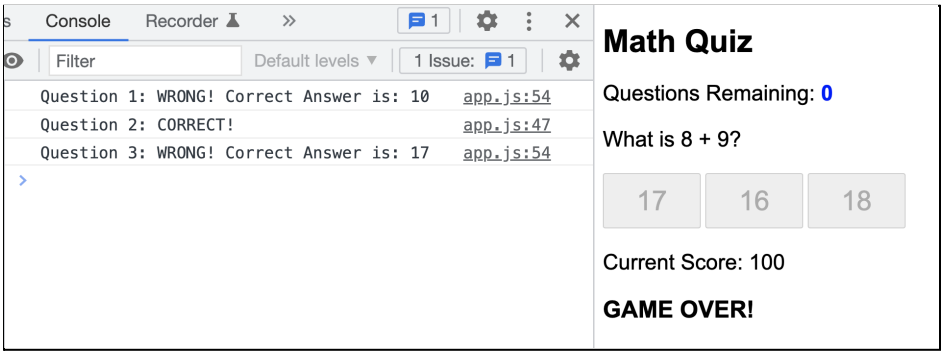


3. Game Over

The game is over when there are no questions remaining. When the game is over, update the UI to:

- Disable all buttons
- Display a game over message

Example of game over



4. Restart the Game

The game can be restarted by refreshing the website.

— END OF ASSESSMENT —