Assignment 1

Client-side programming with Javascript

Submit a single zip file called **assignment1.zip**.
This assignment has 100 marks.
You should read the marking scheme posted on cuLearn for details.

Assignment Background

In this assignment, you will develop a web page that allows the user to play a trivia game within the browser. Everything you implement for this assignment will involve client-side programming. The final part of the assignment will require you to request additional information from an online web service.

To begin the assignment, download the trivia.js file from cuLearn. This file contains a dataset with six sets of trivia questions, stored in the object 'tests'. The tests object contains six keys, each representing the name of a test, and each of the associated values contains an array of objects representing the questions associated with that test. Each question object has the following format:

```
{
      "category": string representing the category of the question,
      "type": string representing the type of the question,
      "difficulty": string representing the difficulty of the question
      "question": string representing the question text,
      "correct_answer": string representing the correct answer,
      "incorrect_answers": array of strings representing the incorrect
      answers (note: there are a varying number of incorrect answers in
      some questions)
}
```

Before starting your design for the assignment, you are encouraged to read through the entire specification. It is possible that later questions will inform your design decisions and by preparing in advance, you can avoid having to significantly refactor your code as you progress through the assignment.

Page Load (10 marks)

Create an HTML page called “trivia.html”. When the page loads, the page should contain only three elements: a drop-down list populated with each of the test names from the provided dataset (i.e., the property names of the tests object: “Test 1”, “Test 2”, etc.), a “Load Test” button, and a “Random Test” button. The dropdown list must have an item selected by default, but which one is up to you.

Load Test (20 marks)

When the user clicks the “Load Test” button, the data from the test selected in the drop-down list should be loaded. More precisely, when this button is pressed:

1. Any existing test should be cleared from the page, along with any feedback related to that test.
2. Each question within the selected test, along with each question’s possible answers, should be displayed on the page.
3. For each question, the answers should be displayed in random order, so that the correct answer is not always in the same place.
4. There should be an obvious separation between each question.
5. It should be obvious which answers belong to each question.
6. It should only be possible to select one of the possible answers for each question.
7. At the bottom of the list of questions the page should have two buttons, labeled “Clear Test” and “Check Test”.

Check Test (25 marks)

When the user clicks the “Check Test” button, you must first check that all questions have been answered. If any questions have not been answered, the text color for those questions should be set to red. Additionally, the user should be shown an alert that informs them that they must answer all questions before checking the test.

If, on the other hand, all questions have been answered, you should reset the text color of any questions that had previously been changed to red, and the following test checking process should proceed:

1. You must calculate the number of correct answers. The user’s total score should be displayed somewhere obvious on the page with the format “# of correct / # questions” (or something similar). So, if there are four questions, and the user correctly answers three of them, the score should be displayed as ‘3/4’.
2. A green checkmark should be shown beside any correctly selected answers. You may wish to use the check.png file included on cuLearn for this purpose, though you will likely have to resize the image to make it fit on your page appropriately.

3. A red X should be shown beside any incorrectly selected answers. You may wish to use the x.png file included on cuLearn, though you will likely have to resize the image to make it fit on your page appropriately.
4. There should not be any checkmarks or Xs next to answers that are not currently selected. If a previous Check Test added extra checkmarks or Xs to the page, you should remove them.

Clear Test (10 marks)

When the user clicks the “Clear Test” button, a confirmation dialog should be shown to the user asking them if they are sure they want to clear all the current test answers. If the user selects “Okay” or “Yes”, all answers within the test should be set to the un-selected state. Additionally, anything added to the page by the “Check Test” button should be removed from the page, made invisible, or otherwise reset to their original state. If the user selects “Cancel” or “No” within the confirmation dialog, then nothing should change.

Random Test (10 marks)

When the user clicks the “Random Test” button, you must use an XML HTTP request to load a new test randomly. This request should not modify the original dataset included in the file.

To do this, you can make a GET request to the Open Trivia Database (OpenTDB) API using the URL https://opentdb.com/api.php?amount=10. This will return a JSON string that should contain two keys: “response_code” and “results”. A response_code value of 0 indicates the request was successful. The value associated with the results key is an array of questions following the same format as the provided dataset.

If the request was successful, the returned test data should be loaded onto the page as usual (i.e., the same display as when clicking “Load Test”). If the request was not successful (i.e., a non-zero response code), you should display an alert indicating the request failed and the page should look as it did when initially loaded. If you are looking for additional details about the OpenTDB API, you can find information at https://opentdb.com/api_config.php.

Extension to Specification (10 marks)

Ten percent of the marks for this assignment will be allocated for an extension that you add to this specification. To receive these marks, you must decide on an additional feature (or features) to add to the application and implement them successfully. You should include a detailed explanation of your addition in your README.txt file. Your extension should not break any of the existing requirements. See the marking scheme for more details.

Code Quality and Documentation (15 marks)

Your code should be well-written and easy to understand. This includes providing clear documentation explaining the purpose and function of pieces of your code. You should use good variable/function names that make your code easier to read. You should do your best to avoid unnecessary computation and ensure that your code runs smoothly throughout operation. You should also include a README.txt file that explains any design decisions that you made, as well as any additional instructions that may be helpful to the TA.

Recap

Your zip file should contain all the resources required for your assignment to run. The TA should be able to open the trivia.html page and use your page without any additional setup.
Submit your **assignment1.zip** file to cuLearn.
Make sure you download the zip after submitting and verify the file contents.
