COMP 2406A
Winter 2020 – Tutorial #3

Objectives

- Practice using Javascript for client-side programming
- Practice performing asynchronous requests using XMLHttpRequest
- Implement a client that interacts with a server to retrieve new information to include in an HTML page

Expectations

The minimum problems you should complete for this tutorial are problems 1-3. Problem 4 outlines some more features you could add to the page if you are looking for more practice. If you have not completed tutorial #2 yet, it would be advisable to work on that first, as it will be more applicable to the first assignment.

Tutorial Background

In this tutorial you will write client-side HTML and Javascript that will be capable of interacting with an open web API called OpenTriviaDB. Your Javascript will load trivia question data from the API, populate the web page with question data, and allow the user to select an answer. Your code will then check the user's answer, update their score, and repeat the process. It is important to note that this overall design is different from the first tutorial – instead of having all the question data contained in your Javascript file, your Javascript code will request question data from a server on the web.

As with all the tutorials and assignments, you are encouraged to read the entire document before proceeding. Knowing the overall requirements of the system will inform the design decisions you make early in the development process.

Problem 1 (Building the Basic HTML)

Create a simple HTML page that will hold the information you need for the rest of the problems. At minimum, this HTML page will require a section for the user's score, a section to hold question data, and a button for the user to submit the answer. This HTML page should also reference a Javascript file that you will continue to add code to throughout the tutorial. When the page initially loads, it would be acceptable to only show a subset of all this information (e.g., only the score) until the first question data has been loaded from OpenTriviaDB.

Problem 2 (Loading a Question)

Now that you have the basic structure of the page, the next step will involve retrieving a question using the OpenTriviaDB API. Add code to your Javascript file that creates an XMLHttpRequest object and submits a "GET" request to the URL https://opentdb.com/api.php?amount=1&type=multiple. If this request is successful, the response text available in your callback function will be a JSON string with the following format:

```
{
    "response_code":0, //you can ignore this for now
    "results":[ //an array of objects, which in this case only has one item
        { //a question object
            "category" : "Category Name",
            "type" : "multiple",
            "difficulty" : "easy",
            "question" : "question text…",
            "correct_answer" : "Some answer that is correct",
            "incorrect_answers" : ["incorrect answer #1", "incorrect #2", "incorrect #4"]
        }
    ]
}
```

The data in this JSON string represents the question data you will need to add to the page. Since it is JSON data, you can easily convert it into a Javascript object using the JSON.parse(string) method. Write code to convert the JSON string to an object and display the question data on the page. At minimum you will need to display the question and all four of the answers. It would make sense to store the answers as radio buttons or in a drop-down list. If you are using radio buttons, ensure you set the 'name' attribute of each radio button you create to be identical. This will ensure only one can be selected at a time. If you are having trouble with this part, you can consult the AJAX lecture notes/recordings or look at the w3schools tutorial at:

https://www.w3schools.com/xml/xml_http.asp

Problem 3 (Creating a Simple Game)

Now that you can retrieve the first question and put it on the page, you will need to be able to accept an answer from the user. Add handling code that allows the user to submit their answer by clicking the submit button. Alternatively, you could automatically submit the answer when one of the answers is selected.

Once the user submits an answer, you will need to check if the answer is correct. You will have to get the value of the answer associated with the selected radio button and compare it to the correct answer of the current question. How will you do this? One of the most straightforward approaches is to set the 'value' attribute of the radio button to the answer it represents when you create the button. You can then find the radio button that is selected and access its value. Once you can retrieve the selected answer and compare it, determine if the user was correct and update their score on the page.

Now that the user can answer a single question and have their score updated, you can write Javascript code to make the game continue. Once the user has answered, request another question by making another AJAX request to the same URL. You can repeat this process indefinitely or add a limit so the game ends after X turns.

Problem 4 (Further Extensions)

If you are looking for more challenges, you can consider some of the possibilities below:

1. Add a timer to the game. You can require that the user must answer the question before the timer expires, or you can assign points relative to the amount of time it takes them to answer the question.
2. Add a feature that removes incorrect answers after a specific time period to make the question easier for the user. This also would require a timer. The score could also be based on the amount of answers that have been removed.
3. Modify your code to request X questions at a time so that multiple requests do not need to be made. This involves modifying the URL you are requesting slightly. See the OpenTriviaDB API at https://opentdb.com/api_config.php for information.
4. Add a drop-down list to allow the user to select a category and/or difficulty for their questions. This also involves modifying the URL you request. See the OpenTriviaDB API at https://opentdb.com/api_config.php for information.