CS5003

Practical 1

University of
St Andrews

Matriculation Id: 230015450

Report

Overview

In this assignment, I used the Open Trivial DB API and basically completed all the basic and intermediate requirements. Among the basic requirements, the following requirements are implemented:

- design of html titles and subtitles, random display of the multiple-choice questions obtained, and recording the number of correct and incorrect answers.
- When the user chooses to end answering questions, they can exit the answering interface and click Start directly without refreshing the interface. The game starts a new round of answering questions. Among the intermediate requirements, the following requirements are implemented:
- the total score of the user's answer is calculated according to the different difficulty of the question.
- The fetch token obtains non-duplicate questions.
- delete half of them the function of wrong answers and only use once in one game.
- tracking and displaying the best score of the user's answer until the web page is refreshed, and the countdown function. But the

method I use is the total time of answering the question rather than the time of each question, which I will discuss later elaborate.

Technologies & Resources

Some methods are based on the content on W3schools and MDN. The relevant knowledge points of each part include:

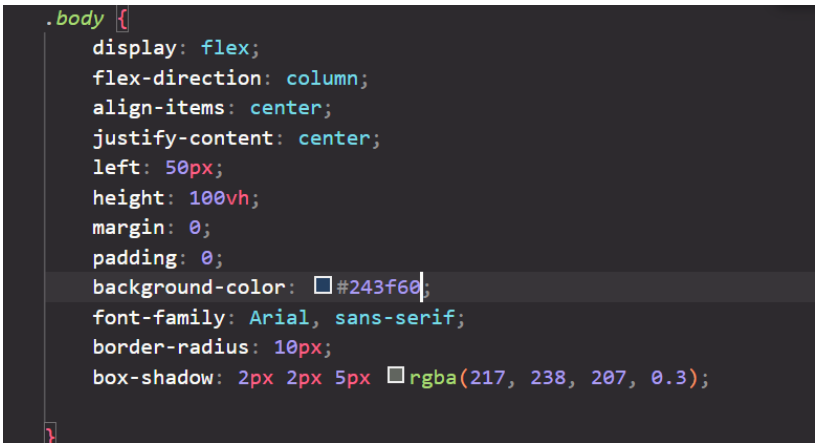
- CSS: Flexbox layout, hover
- Javascript: DOM operations, Math.max(...array1), Window Object Methods

Coding design of important processes

- design the interface and Page turning design.
    a. Use a simple HTML structure to present questions, options, and various buttons. The interface has done some DOM operation processing to display different content before and after clicking the start game button.
    b. Name a variable currentQuestionNumber to control the order of questions. Once the user answers the question, check whether it is correct or not and compare it with the total number of questions. If the current question number is less than the length of question array,

continue to answer the question, otherwise complete the question.

c. As for the CSS part of the project, I used a Flexbox layout to make the web interface more flexible and prevent elements from overflowing. Items are arranged in columns.



```css
.body {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    left: 50px;
    height: 100vh;
    margin: 0;
    padding: 0;
    background-color: #243f60;
    font-family: Arial, sans-serif;
    border-radius: 10px;
    box-shadow: 2px 2px 5px rgba(217, 238, 207, 0.3);

}
```

- API data acquisition and Use API tokens to prevent duplicate issues.

    a. Use the Fetch () function to get what the API returns and convert the values of results into a new object array. Here I am using the Open Trivial DB API, because the answers need to be displayed randomly, so I set up a new answer array here, put the obtained correct and incorrect answers into it and sort them randomly. Here I refer to the sorting method on w3schools, use the sorting random function function (a, b) {return 0.5 - Math.random() } to

disrupt the order, and then use DOM operations to write the html page.

b. In subsequent operations, the tokens are obtained respectively, and the tokens are passed to the server for verification.

- Count the number of correct and incorrect questions and calculate scores based on difficulty（when the incorrect questions number over 3,end the game and get 0 point）

a. Here I create an object to store the correct and incorrect numbers, return the obtained values to the array, and then use DOM operations to write the updated values to the web page.

```
var checkAnswer = {
    correct: 0,
    wrong: 0,
};
//update the correct and wrong count
function updateCountDisplay() {
    document.getElementById("correctCount").textContent = checkAnswer.correct;
    document.getElementById("wrongCount").textContent = checkAnswer.wrong;
};
```

b. Next, I write a function to calculate the total score based on the difficulty. In the inside of function, I use if...else statement to judge the difficulty and stack the points obtained each time. Once the number of errors reaches 3, the previous function will be triggered to end the program.

- Update best score.
    a. The method I use here is to set up an empty array to obtain the score of each game, traverse this array and compare it with the maximum value of the current array, return the newly obtained maximum value and write it to the web page.

```
//create a null array to store total point,it will reset when the game is over
let bestScore = [];
function updateBestScore() {
    bestScore.push(totalPoint);
    let max = Math.max(...bestScore);
    for (let i = 0; i < bestScore.length; i++) {
        if (bestScore[i] > max) {
            max = bestScore[i];
        }
    }
    document.getElementById('best').textContent = max;
    document.getElementById('best').style.display = 'block';
}
```

- Add a button for players to end the game at any time and remove half of wrong answers (only use once)
    a. You need to add an event listener to call the endgame function to end the game.
    b. How to randomly remove half of the wrong answers, the logic design here is slightly complicated. First, I set up a DOM operation and bind an event listener to call the function that removes the error option. Then construct a new array to store the wrong answers to the current question, and then construct a function with the removal function. The design of this function is to first determine

whether this function has been used. If not, a random number smaller than the length of the new array is regenerated via the Math.random and Math.floor functions, looping Array.splice() twice to remove incorrect answers. At this time, the logic on the main body has been designed, but the problem I encountered is that the option box cannot be hidden. This problem continues to this day.

- Countdown clock
    a. For the timer options, I refer to the timer-related content in the lecture module of the third week. Reseting the timer by clicking the start and end button or waiting for the game to end. The question requirement is to limit the answer time for each question, such as 10 seconds, but the function I implemented is some inconsistent with the question.
    b. If the timer is designed according to the question requirement, the timer needs to be called every time when answering the question, then it needs to be Add a paragraph with id = timer to the HTML page, name a countdown variable in JavaScript, design a countdown function, and reset the timer when switching questions.

c. This may require using the if else statement again inside the questionAdd() function(this function is to switch the question) to determine whether to automatically switch to the next question based on the answer time. If the user's answer time times out and no option is selected, an incorrect option should be automatically assigned to count the number of wrong questions.

Evaluation

In this assignment, I completed the basic requirements in the first part relatively smoothly, but I met several troubles in the second part. The first is that when the user chooses to delete half of the wrong answers, the option content is indeed deleted but the option box is not hidden. The second question is to design global variables and block variables, because several variables in this job need to be set as global variables. For example, if the totalPoint variable is set as a block variable, when I want to use it to find the highest score, It cannot be used and report an error that the variable is not found . The last problem is that I set the timer to count down the total time, rather than count down for each question. If I have more time, I hope to design a small timer to control the answering time every time I answer a question and try to solve some advanced requirements.

Reference

1.https://www.w3schools.com/css/css3_flexbox.asp

2.https://www.w3schools.com/cssref/sel_hover.php

3.https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/max

4.https://www.w3schools.com/jsref/obj_window.asp

5.https://www.w3schools.com/jsref/prop_node_textcontent.asp

6.https://www.w3schools.com/jsref/prop_element_nextelementsibling.asp

7.https://www.w3schools.com/jsref/prop_style_display.asp