University of St Andrews

17/Feb/2019

CS5003 — Masters Programming Projects

Assignment 1:

Project 1

Student Id: 180029803

Word Count:

Table of Contents

Introduction

The aim of the assignment is creating a structured single-page Trivia Quiz application in JavaScript by utilizing from API as one of the components of the application. The core feature of this application is providing a knowledge game to the players where they choose the proper answers from either multiple or true/false choices regarding to the given questions and try to finish the game with a decent score.

This report deeply explains the design criteria/method of creating functions in JavaScript and it's frameworks from retrieving data from API at the beginning of the game to storing the details of the player that are saved after the player exits the game; the drawbacks of the selected codes and difficulties in implementing the requirements. Also, it supports the provided arguments by giving necessary evidence including screenshots of the tests that are done on the browser.

Design

The necessary requirements are starting with providing a start page for the game where there is a form for entering the player's name and choosing bonus category from the randomly given category which bring in doubled scores to the player if s/he knows the answer of the question correctly within the given countdown clock limit. The form obligates user to write his/her name and choose a category by alerting when there is no entry in the form area and not allowing player to start the game unless the category is selected. By doing this, it ensures that the game developer is able to store the player details with their names and allow players to keep track of their scores when they finish or exit the game. The drawback of using pop-up page for providing the form was losing the form and having a non-functional web-page if the user clicks on somewhere in the page out of the form. However, this issue was sorted by adding a button for calling the pop-up page and keeping it behind the pop-up page until the user entries are submitted through clicking the start game button and they are accepted by the system. After the entries are accepted by the system, the user becomes a player and faces with the first

question and the answer choices which s/he should choose one of it before 30 seconds are finished. The chosen answer shake, the incorrect choice turns into red and correct one returns into green when the player gives wrong answer to the question. If the time is up, the questions and answers change in order not to allow any attempt, the player is deemed to have got the answer wrong, so the system increases the number of incorrect answers by 1. Also, when the player answers the question correctly within the time limit, s/he gains score which varies by the difficulty of the question and the system increases the number of correct answers. In case of receiving 3 incorrect answers either with answering or not being able to answer before the time out, the game is over and the play again button appears on the page. Moreover, on right of the numbers correct/incorrect answers and countdown time, there are 3 buttons as lifelines which help player to find the answer of the question just for one time for each in the same game: Lifeline 50/50, Ask a Friend Lifeline and Ask the Host Lifeline. When the player clicks on Lifeline 50/50 the system takes out the two incorrect choices of the answers. Ask a Friend option allows player to ask to his/her friend or search the question on the internet by giving him/her 1 more minute in addition to the current seconds. For the third Lifeline, the system provides some estimations on the answers whereas the accuracy varies by the difficulty of the question. In other words, the system acts like real audiences and provide the similar probability for finding the right answer. While using all of these, the timer continues counting down.

When the game is finished because of reaching 3 incorrect answers or finishing all of the 10 questions, the score table which includes best 10 players' scores, new game button and save and exit button are provided demonstrated by a pop-up window under the label "Game Over". If the player clicks on the new game button, the stages of starting to a game is repeated and Bootstrap JS modal for the form comes up, if s/he chooses the save and exit button, the game finishes and the system doesn't suggest anything unless the player clicks on restart/exit button on the main page again in order to reach new game button and restart the game.

The design is supported by an external HTML, CSS, and images files to draw a more preferable looking and while at the same time, allowing developers to read the codes and use

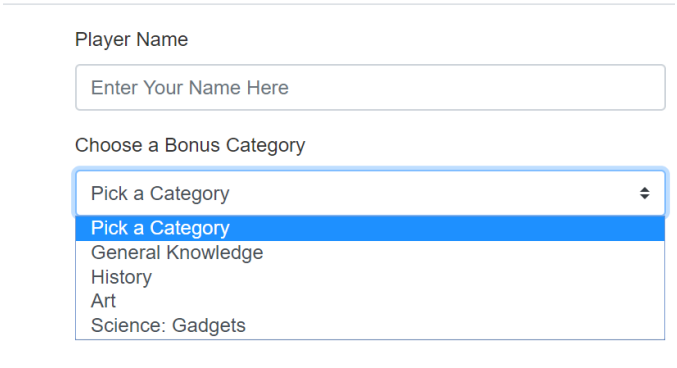
them easily in the future. Since the application requires lots of functions and convenient design, using pure JavaScript and CSS made difficult to reach the desired outcome. Therefore, utilizing from jQuery was the optimal option while developing the application for this assignment in terms of being the lightweight of JavaScript by taking a lot of common tasks that require many lines of JavaScript code to accomplish, and wrapping them into methods that you can call with a single line of code.

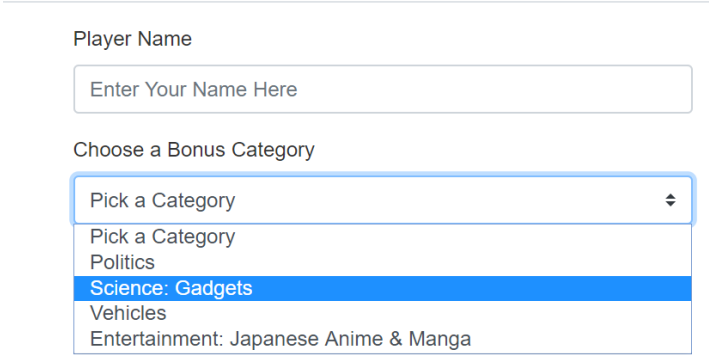
Implementation

For implementing the codes in JavaScript file for the functions, firstly, the ids that indicate which properties are going to be used, were called back from the HTML file by using DOM method. After that, the elements were received from API by fetching the API through its URL and results of the arrays were used for providing questions and answers and scoring the players. Since the incorrect, correct answers, difficulty of the questions etc. are defined by the API, the few things to reach them were naming, separating by using separator, conditioning and pushing them to the system. While taking the data from the library, the same question was prevented from coming up more than once and this can be easily checked by using console.log method. The system was developed as calling 4 random categories from the library for each game by using function “randomCats”. Therefore, the questions, difficulty of the question categories, incorrect and correct answers of them change for each game. As another requirement, the commands for scoring were arranged by 1-) indicating which difficulty level brings in how many point(s) and 2-) doubling assigned score of the question which comes from the bonus category. Moreover, having shorter than one question and incorrect answers equal to three are conditioned as finishing the game. When the player exits the game by clicking on exit button, the local storage keeps the player details as items by JSON Stringy method. The provided questions and the answers disappear not only by clicking on exit button, but also by requesting new game, so each time the local storage renews itself by adding the new data.

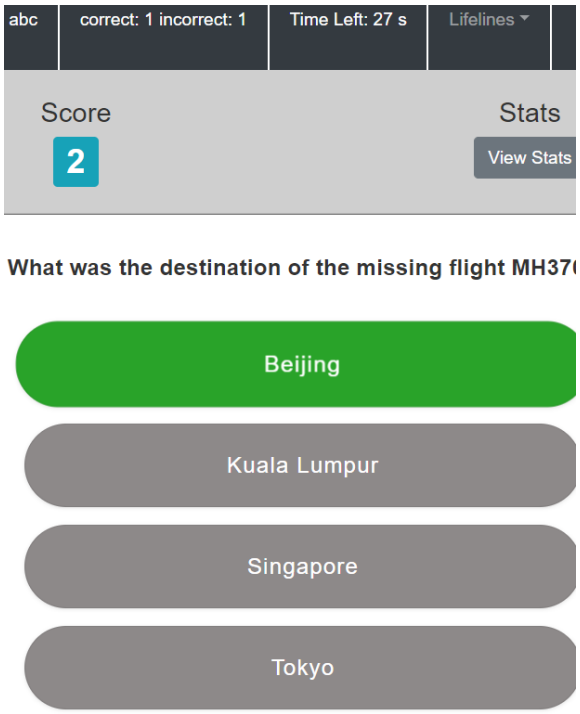
Testing

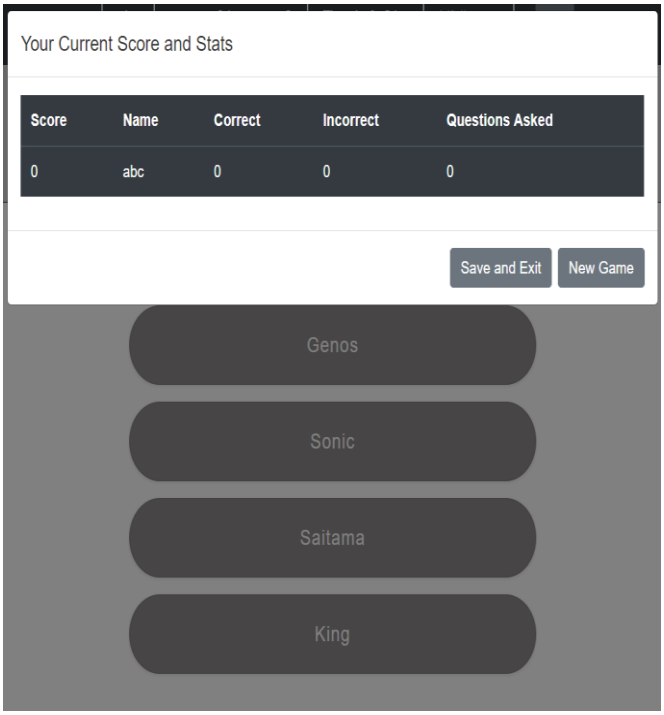
The screenshots below are demonstrating the random categories that change for each game.





The screenshot on the left is showing what happens when the player chooses the right answer. The right side demonstrates the score table and the exit and new game button.





Difficulties

4 random categories for 10 questions that are asked in the quiz, have demonstrated for each game, and the questions are received randomly, the difficulty of the questions are decided and given by the system before the game starts. Since the difficulty levels are decided before, the number of the same difficulty levels are decided together with them and the user cannot choose more than the available numbers of the same difficulty level for the next question. To Illustrate, if the user wants to answer 5 difficult questions s/he cannot do it when the system provides only 4 difficult questions. This complexity obstructs allowing player to choose the next question's difficulty level. Lastly, unfortunately, the written codes for ask a friend lifeline and ask a host lifeline are started not working for some reason at last moment.

Conclusion

Taking everything into consideration, all of the basic and intermediate requirements were implemented. Except the requirement of allowing user to choose the difficulty of the next question, all of the advance requirements were fulfilled. Since timer has to be adjusted for each function, an additional JavaScript file was used just for storing the timer objects, functions and commands. In order to provide a better view, CSS codes are implemented to the objects of API and overall webpage.