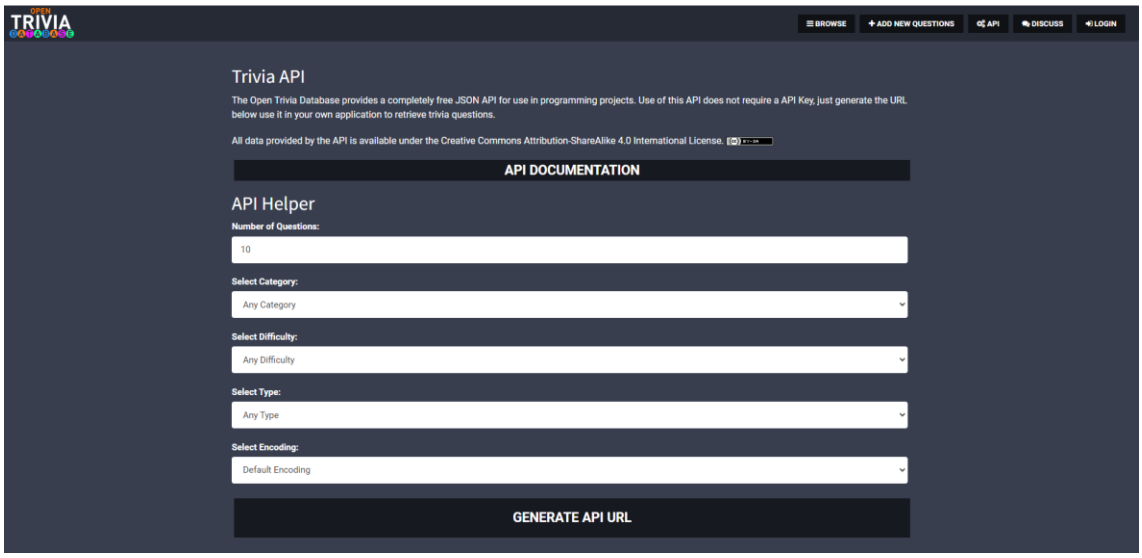
## Design and Functionality Process

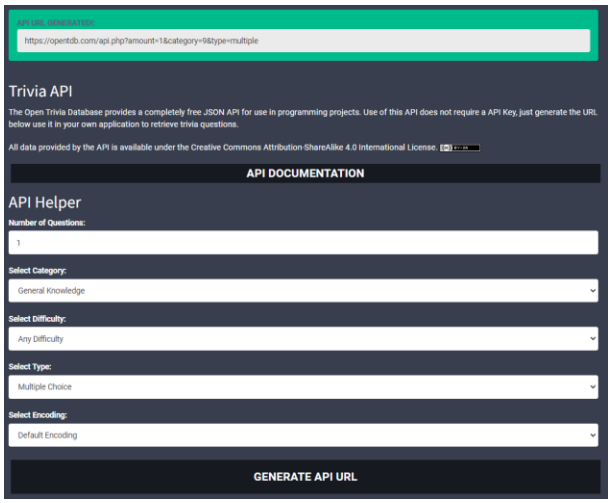
# Exercise 2.2: Personal API to Project 2: Trivia App

**Sara Cameron 139-5067** **16 August 2023**

I am a trivia nut so I wanted to make a Trivia App. I found https://opentdb.com/, which is a trivia api that does not require a key. I was able to see the possible options:




I chose:
**Number of Questions: 1** (because I only wanted one question at a time to appear on the screen)
**Select Category: General Knowledge** (I was worried the database is not large enough to tolerate specific topics)
**Select Difficulty: Any Difficulty**
**Select Type: Multiple Choice** (b/c the other option was T/F, which I thought would be boring. In hindsight, it would have been MUCH easier….especially when I needed to figure out how to shuffle/randomize the answers).

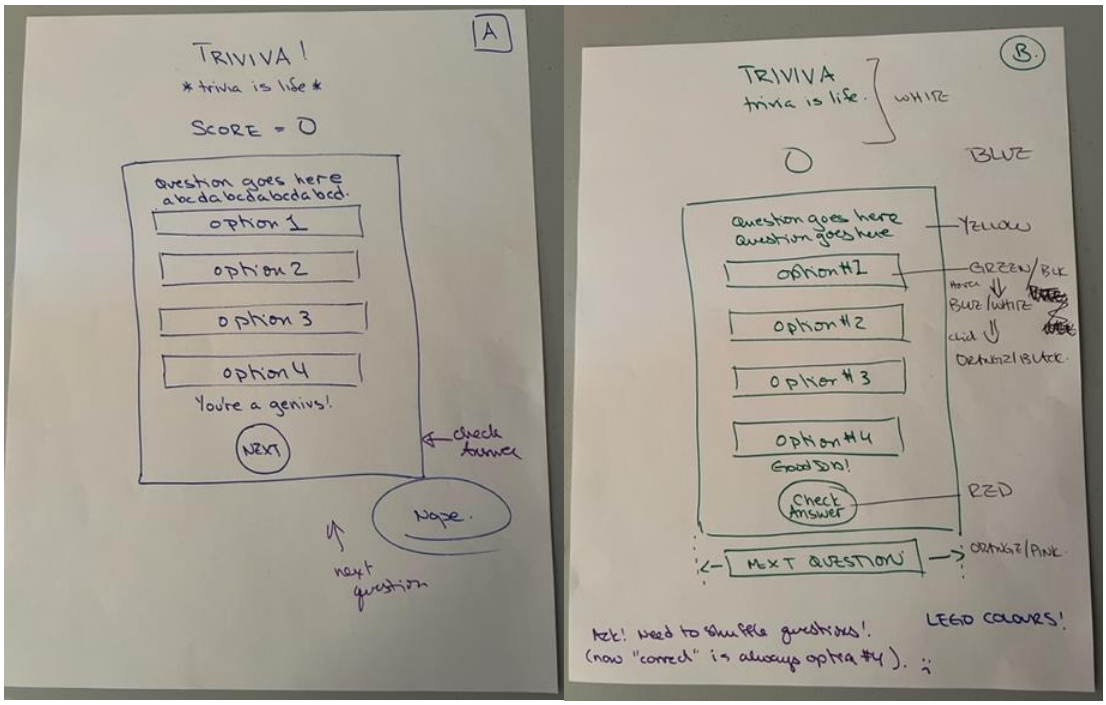


**Functionality and design:**

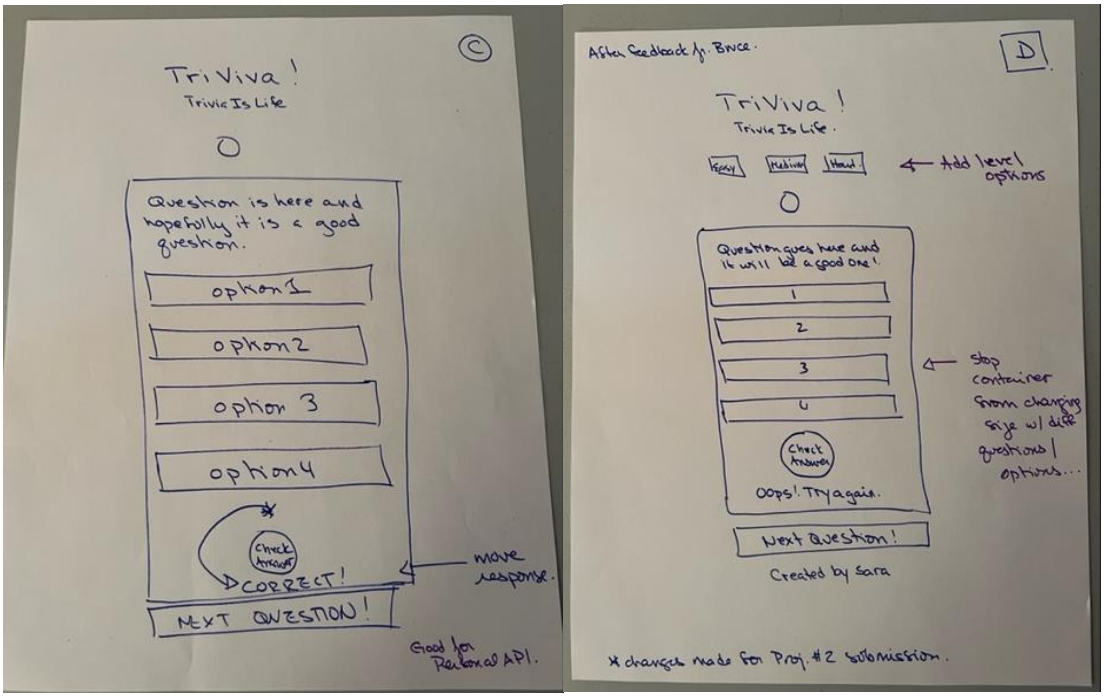
There were several concerns I had in designing the functionality: the app needed to be instinctive to use, it had to be interesting to capture the user's attention, and it had to look and feel fun to use.

I decided to make it as visually simple as possible: a running score at the top (that I was eventually able to code so that the score could go up OR down, which is evil but fun. No one likes to get a score of -6…), a button to verify your answer, a response (good or bad) depending on the answer you select, and a button to go to the next question.

Once I knew the options for this API, I designed my Trivia APP. This went through several iterations as the project advanced[1]:

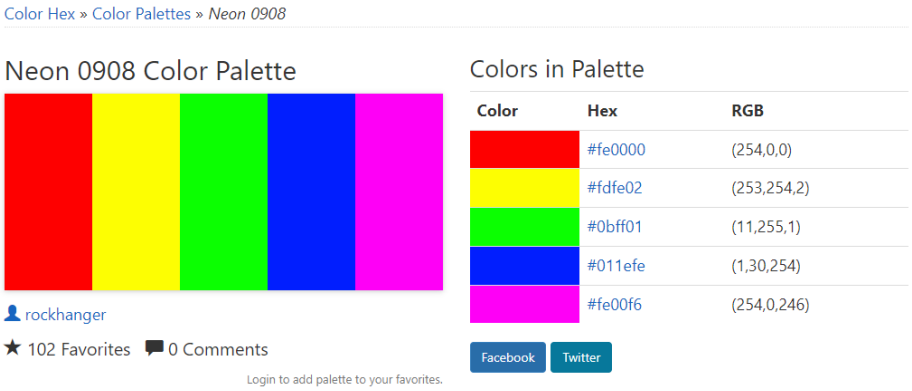



[1] I know it is good practice to decide on the basic structure before starting (h1, h2, buttons, etc.), but I was so out of my depth that I decided to figure it out as I went. I now realize that I would have saved myself a lot of trial-and-error if I had sat down and really thought it through before starting (I know, I know). I won't make this mistake in the future.




**Colour Palette:**

I wanted my API to look like a child's toy: bright colours like Legos or primary colours. I found this palette:




Added an orange because I needed a 6th colour: #ff8c00

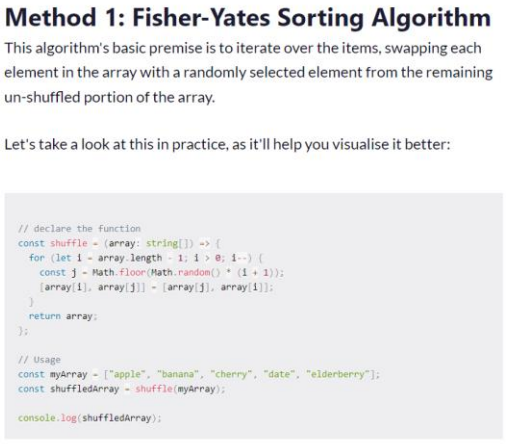
(fyi: this colour palette is WAY out of my comfort zone)

I also wanted to use a fun font, so I went with Comic Sans MS for the TriViva (h1). It was hard (a bit annoying, actually) to read for the questions/answers, though, so I just used a simple Calibri font for the rest of the App.

**Challenges**[2]:

I thought it was important to allow the user to attempt to get the correct answer multiple times, but this originally resulted in the user being able to get a point even if they took 4 tries to get the correct answer. Once I was subtracting points for incorrect answers/guesses/tries, the user gets penalized if they need to guess, or they choose incorrectly, but they can try again. I guess I believe in second (third, fourth) chances! This, however, changed again once I received feedback for my MVP, and the number of tries has been limited.

There were a lot of challenges in this project, but the major problem I ran into was that the correct answer was always appearing as the fourth answer in the options. I am not proud to say it took me about 5 days to notice that this was an issue. I had NO IDEA how to approach this, so I searched to see if others had encountered this problem or if it was an issue specific to this API. I discovered that I need the *Fisher- Yates Sorting Algorithm* to randomly sort the multiple choice answers.

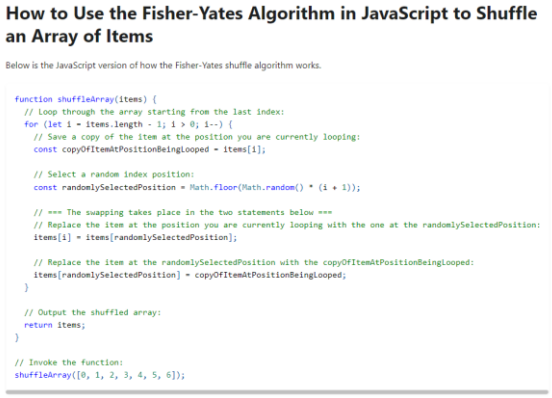


**Method 1: Fisher-Yates Sorting Algorithm**

This algorithm's basic premise is to iterate over the items, swapping each element in the array with a randomly selected element from the remaining un-shuffled portion of the array.

Let's take a look at this in practice, as it'll help you visualise it better:

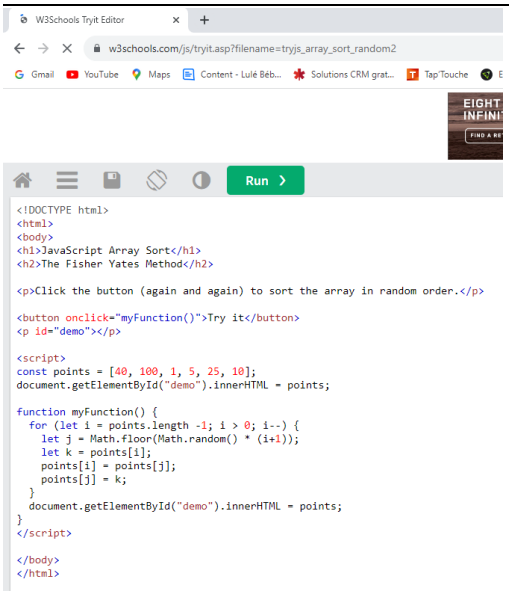
```
// declare the function
const shuffle = (array: string[]) => {
  for (let i = array.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [array[i], array[j]] = [array[j], array[i]];
  }
  return array;
};

// Usage
const myArray = ["apple", "banana", "cherry", "date", "elderberry"];
const shuffledArray = shuffle(myArray);

console.log(shuffledArray);
```



**How to Use the Fisher-Yates Algorithm in JavaScript to Shuffle an Array of Items**

Below is the JavaScript version of how the Fisher-Yates shuffle algorithm works.

```
function shuffleArray(items) {
  // Loop through the array starting from the last index:
  for (let i = items.length - 1; i > 0; i--) {
    // Save a copy of the item at the position you are currently looping:
    const copyOfItemAtPositionBeingLooped = items[i];

    // Select a random index position:
    const randomlySelectedPosition = Math.floor(Math.random() * (i + 1));

    // === The swapping takes place in the two statements below ===
    // Replace the item at the position you are currently looping with the one at the randomlySelectedPosition:
    items[i] = items[randomlySelectedPosition];

    // Replace the item at the randomlySelectedPosition with the copyOfItemAtPositionBeingLooped:
    items[randomlySelectedPosition] = copyOfItemAtPositionBeingLooped;
  }

  // Output the shuffled array:
  return items;
}

// Invoke the function:
shuffleArray([0, 1, 2, 3, 4, 5, 6]);
```




[2] I wrote extensive notes in my JS code for the MVP (which I cleaned up for Project 2) as I was learning and trying new things. I am not sure I understand everything yet, but it seems to be slowly sinking in…

I used chatGPT to customize the Fisher-Yates Sorting Algorithm to my code, because I was about to lose my mind…I'm still not sure I understand it.

It was at about this time that I realized that using True/False questions, although less interesting, would have been much easier to understand and code. Almost anything else seemed like it would have been easier at several points during the process...

**Ongoing issues:**

I don't like including code that I don't understand, so using the *Fisher- Yates Sorting Algorithm* is bothering me a lot. However, I need to live with it since my Trivia App doesn't work without it.

On some questions (but not always the same questions), all the answer choices are shown to be false. I believe that this is an issue with the API.

**Improvements for the Project 2 submission:**

When you *reviewed* my Exercise 2.2: MVP, you suggested that I prevent the question container from re-sizing and that I include an additional choice for the user. I didn't want to give the option of topic choice because, as mentioned before, I was concerned that the library of questions was too small. I decided to include the option of a level of difficulty. The added bonus is that the styling for the difficulty buttons makes the Trivia App look more balanced.

Once my Exercise 2.2: MVP was *graded,* you noted a bug that allowed the user to hit the "check answer" button multiple times to accumulate points (if they had chosen the correct answer). I changed the code to prevent this from happening, but now players only get one chance to choose an answer. I read a lot about trivia apps, and a possible fix was to create an array to keep track of answers to prevent accumulating points, but it was way out of my depth, so I just limited players to one chance. It seems to work.

**References**:

I used A LOT of references and trial-and-error to create my Trivia App. The most significant references I used are:


**https://opentdb.com/api_config.php**

https://docs.idew.org/code-trivia-app/

Learn JS Async/Await, Fetch Requests & APIs by Building a Trivia Game - DEV Community

CodeCast 61-learn-api-s-building-a-trivia-game-w-javascript/playlist

https://www.youtube.com/watch?v=zgHim4ZDpZY (Trivia App in JavaScript)

https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_sort_random2

https://www.tutorialspoint.com/what-is-fisher-yates-shuffle-in-javascript

https://www.freecodecamp.org/news/how-to-shuffle-an-array-of-items-using-javascript-or-typescript/

https://www.color-hex.com/color-palette/1131