Challenge 4 / Javascript Quiz Game

https://stephcambria.github.io/JavascriptDeveloperQuiz/

Unfortunately, since I had such a chaotic week, I neglected to document my progress on building this application as it was happening, so parts of this README might be a little lacking.

For the most part, setting up my files was surprisingly straightforward; my html and css remained pretty bare bones, and the bulk of my work went into the javascript file. My first step was to get as many variables as I could, and add to the list as needed:

```
const startButton = document.getElementById('start-btn');
const nextButton = document.getElementById('next-btn');
const questionContainerElement = document.getElementById('question-container');
const questionElement = document.getElementById('question');
const answerButtonsElement = document.getElementById('answer-buttons');
const scoreContainer = document.getElementById('result-container');
const mostRecentScore = localStorage.getItem('mostRecentScore');
const finalScore = document.getElementById('finalScore')
const username = document.getElementById('username');
const saveScoreBtn = document.getElementById('saveScoreBtn');
```

Next, I wrote out some placeholder questions while I built the functionality of the actual quiz application. All of the questions the quiz would cycle through would be organized in an array, and just to add some flavor, I decided to cycle through these questions randomly each time the quiz is played:

```
let shuffledQuestions, currentQuestionIndex


startButton.addEventListener('click', startGame)
nextButton.addEventListener('click', () => {
   currentQuestionIndex++ //++ means to add 1
   setNextQuestion()
} )
```

While building the other initial functions and testing everything, I noticed that my randomized questions were repeating instead of stopping once the cycle had finished, so I implemented this for loop I found someone suggest on a forum:

```
//for loop to prevent shuffled questions from repeating!
let visited = [];
```

```
let len = questions.length;

questions.sort();

let _temp;

for (let i = 0; i < len; i++) {
   if (questions[i] !== _temp) {
       visited.push(questions[i]);
       _temp = questions[i];
   }
}
```

And it worked seamlessly! Now that my placeholder questions were cycling and randomizing correctly, I finished the functions I had written for displaying the questions, selecting the answers, and what happens visually if the player's answer is right or wrong:

```
function setNextQuestion() {
   resetState()
   showQuestion(shuffledQuestions[currentQuestionIndex])
}

function showQuestion(question) {
   questionElement.innerText = question.question
   question.answers.forEach(answer => {
       const button = document.createElement('button')
       button.innerText = answer.text
       button.classList.add('btn')
       if (answer.correct) {
           button.dataset.correct = answer.correct
       }
       button.addEventListener('click', selectAnswer)
       answerButtonsElement.appendChild(button)
   } )
}


function resetState() {
   clearStatusClass(document.body)
```

```
    nextButton.classList.add('hide')
    while(answerButtonsElement.firstChild) {
        answerButtonsElement.removeChild(answerButtonsElement.firstChild)
    }
}
```

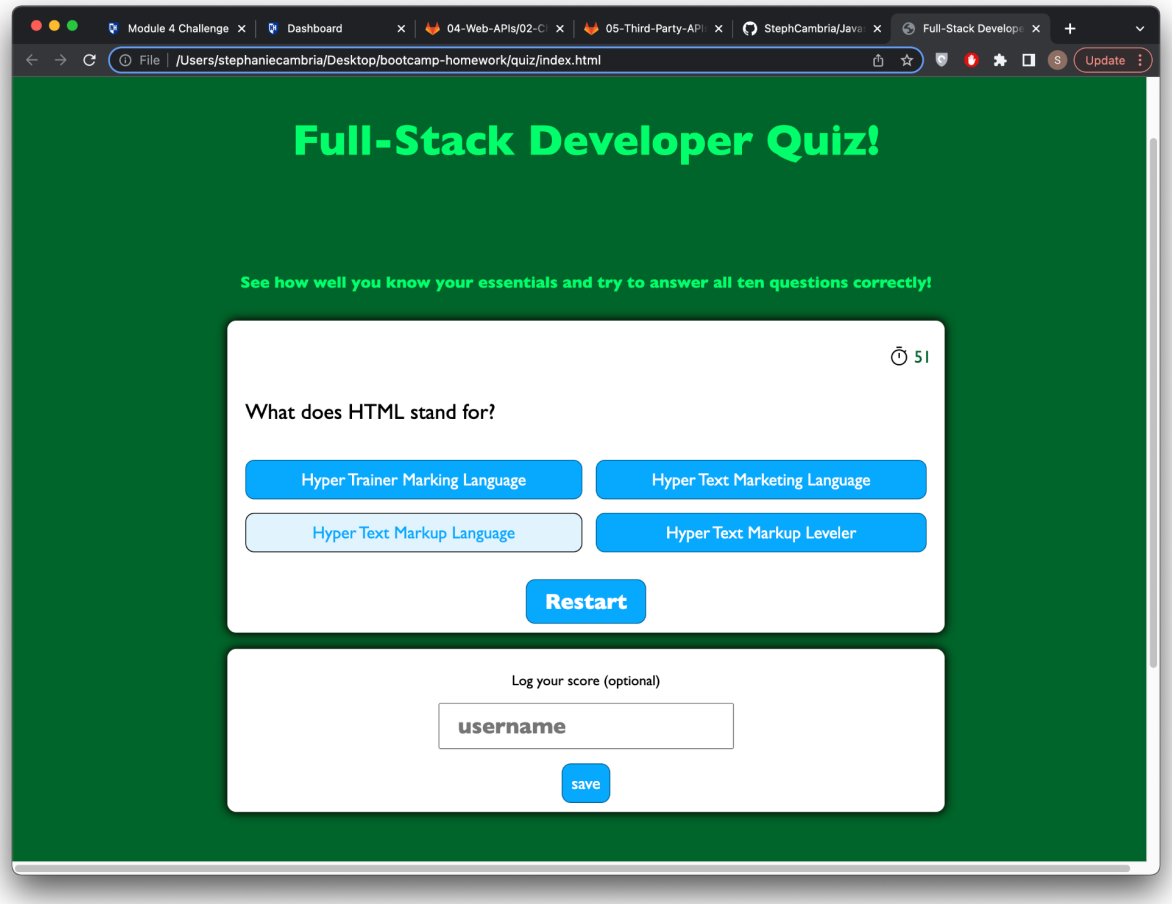
While I was building my core file structures, I more or less lived on both Youtube and various forums to see how others had built similar applications. I wanted to keep my interface extremely simple, with as little distractions as possible (so no pop-up alerts, etc), and one idea I found was to have the background change color to visually indicate whether the answer was right or wrong, which I thought would be a perfect addition! To achieve this, I used a resetState function, and classLists, to tell my code which css property to use depending on the input. To determine what the correct answer to each question is, I used booleans:

```
const questions = [
    {
        question: 'What does HTML stand for?',
        answers: [
            { text: 'Hyper Trainer Marking Language', correct: false},
            { text: 'Hyper Text Marketing Language', correct: false},
            { text: 'Hyper Text Markup Language', correct: true},
            { text: 'Hyper Text Markup Leveler', correct: false}
        ]
    },
    {
        question: 'What are the elements of the CSS Box Model?',
        answers: [
            { text: 'Content, Padding, Borer-box, and Margin', correct: false},
            { text: 'Content, Padding, Border, and Margin', correct: true},
            { text: 'CSS, Padding, Border, and Margin', correct: false},
            { text: 'Content-box, Padding, Border-box, and Margin', correct: false}
        ]
    },
```

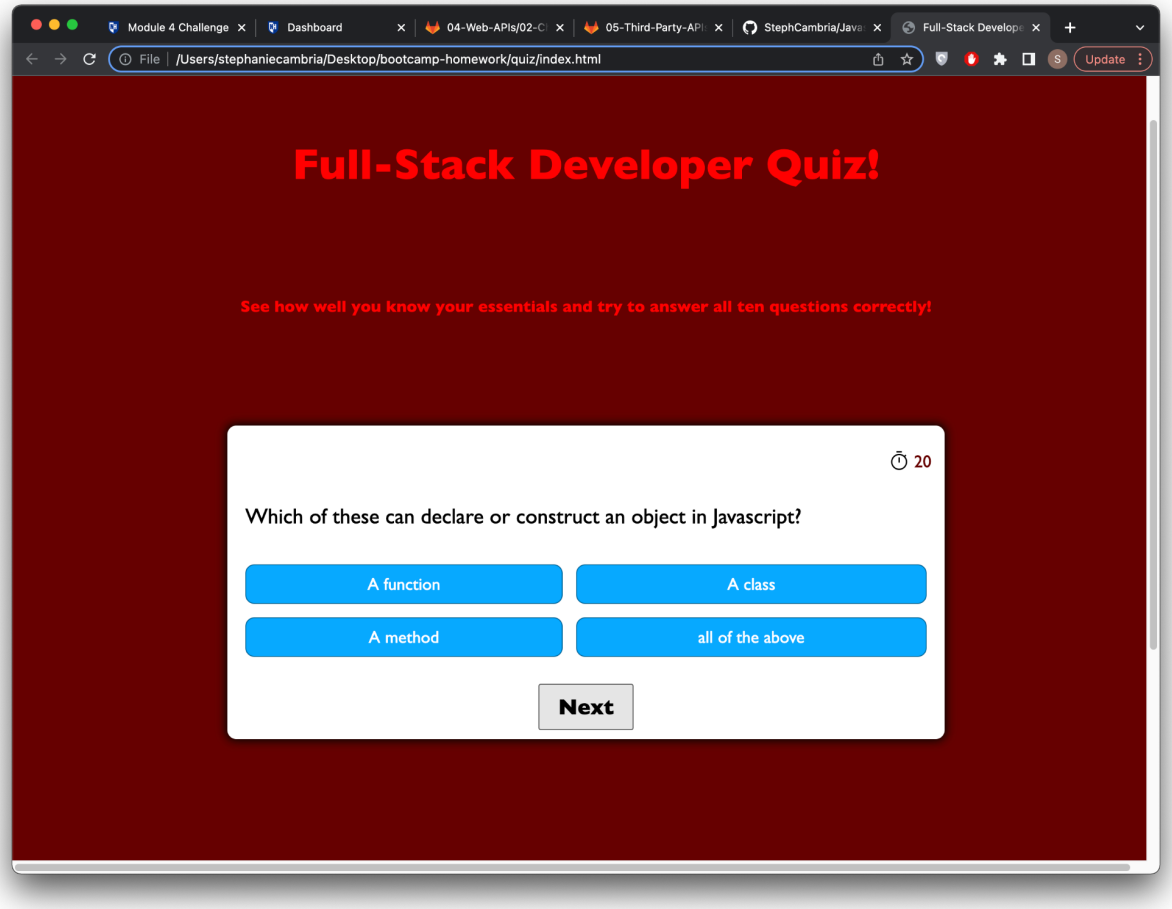
```
function setStatusClass(element, correct) {
    clearStatusClass(element)
    if(correct) {
        element.classList.add('correct')
        setTimeout(secondsLeft++);
```

```
        timerEl.textContent = secondsLeft++;

    } else {
        element.classList.add('wrong')
        //reducing a few extra seconds if you pick the wrong answer
            setTimeout(secondsLeft--);
            timerEl.textContent = secondsLeft;
        }
    }


function clearStatusClass(element) {
    element.classList.remove('correct')
    element.classList.remove('wrong')
}
```

In the live preview, this would be how it looks if the selected answer is correct:



And this is how it would look if the answer is incorrect:



And upon loading the next question, the css would revert to its neutral state.

I had also already added the start button in at this point, which, through using classLists, also doubled as my restart button at the end of the quiz:

```
let shuffledQuestions, currentQuestionIndex

startButton.addEventListener('click', startGame)
nextButton.addEventListener('click', () => {
   currentQuestionIndex++ //++ means to add 1
   setNextQuestion()
} )



function startGame() {
```

```
    startButton.classList.add('hide')
    shuffledQuestions = questions.sort(() => Math.random() - .5)
    currentQuestionIndex = 0
    questionContainerElement.classList.remove('hide')
    scoreContainer.classList.add('hide')
    setNextQuestion()
    clearInterval(timerInterval);
    setTime();
    secondsLeft = 60;
}
```

So next was the option for the player to store their name and score in the local storage. This part stumped me– while I was able to successfully store the input itself into local storage, I could not figure out how to calculate an actual score. Since I was running out of time by this point, I decided to use Math.random to at least calculate *something* to log:

```
//local storage
const highScores = JSON.parse(localStorage.getItem("highScores")) || [];
    console.log(highScores);
const MAX_HIGH_SCORES = 5;
function saveHighScore() {
    scoreContainer.classList.remove('hide')
    clearInterval(timerInterval);

    username.addEventListener('keyup', () => {
    saveScoreBtn.disabled = !username.value;
});
    saveHighScore = (e) => {
    e.preventDefault();

    const score = {
        score: Math.floor(Math.random() * 100),
        name: username.value
    }

    highScores.push(score);

    highScores.sort( (a,b) => b.score - a.score)

    highScores.splice(5);
```

```
    localStorage.setItem('highScores', JSON.stringify(highScores));
}
}
```

Because I wasn't able to make a function to calculate an actual score (and because the deadline was coming up and I still had more to add), I opted out of displaying them outside of local storage. There are probably some remnants of unused variables in my code that I missed when I was cleaning it up. On that note, I should add that I definitely think my javascript code is repetitive and excessive, which is partly from inexperience and partly by choice: while I'm still in these early practicing stages of javascript, I would rather have too much going on in my code than not enough going on. If my script is too wordy, for example, I can always comment parts out to double check what everything is doing. Though I am hoping that my working code will get cleaner and cleaner with experience.

Lastly we had the timer. This is easily the most unfinished part of my application in terms of functionality. This is also easily the part of the challenge that frustrated me the most. Initially setting up a countdown function was simple enough. I was able to get it to restart when the player clicks the restart button, I was able to reduce a few extra seconds when the player chose a wrong answer, and I was able to get it to pause when the player answered the tenth question. However, if the timer is left running, it will not stop at zero, and will just continue into negative numbers. Here is the code I ended up with:

```
var timerInterval
```

```
//TIMER

var timerEl = document.getElementById('count-down');

var secondsLeft = 60;

function setTime() {
  timerInterval = setInterval(function() {
       secondsLeft--;
       timerEl.textContent = secondsLeft;
   }, 1000);
   if(timerInterval === 0) {
       clearTimeout();
       gameOver();
   }
}
```

I'm not sure I can verbalize how long I spent troubleshooting this one specific problem. I'm also kicking myself for completely forgetting to archive all of the different syntax I used in my efforts to fix this. *Please* tell me any possible solution (I'm sure it's something mind-numbingly simple) so I can go back into this project and fine-tune it.

You may also notice a gameOver function that isn't actually used. This is because I had every intention of adding a proper game over / restart / log scores screen, but ultimately was not able to.

I'm leaving this challenge with mixed feelings. On one hand, building this application skyrocketed my confidence with javascript! On the other hand, I was not able to build some of the core functionality of this application, and my troubleshooting led me nowhere. Again, please give me some feedback on how to fix the timer function, as well as a way to calculate an actual score rather than a randomized one, as I would love to come back to this application and properly finish it!