Tommy Las Z23517623 Tommy-Las

February 13th 2022 [Github Repository](https://github.com/COT4930-Spring-2022-Fullstack-Web/hw2-Tommy-Las)

Full Stack Web Development COT4930

Homework #2

Objective 1:

The first objective was to take the W3 School quizzes for HTML, CSS, JavaScript, Bootstrap and JQuery. Some of them I were familiar with the material, and for some I had to study again from the tutorials of the website. Here are my results:

HTML:

Graphical user interface, text, website

Description automatically generated

CSS:

Graphical user interface, text, website

Description automatically generated

Bootstrap:

Graphical user interface, text, website

Description automatically generated

Javascript:

Graphical user interface, text, website

Description automatically generated

JQuery:

Graphical user interface, text, website

Description automatically generated

Objective 2:

For objective two, we had to implement a TicTacToe program using HTML, CSS, Bootstrap, Javascript and JQuery. This program helped apply my newly found knowledge of these languages.

For the HTML structure, I used the container attribute from bootstrap and three columns.

The left column is to store the score of the player and the computer, and also has a button to restart the scores. The middle column just has the tic tac toe game. The right column states whose turn it is and who started first. I used the “border” class from bootstrap in order to see better how each element was placed in the website. Here is a screenshot of an early version of my program. You can see a very light border. That is how also I was able to build the tic tac toe board.

Chart

Description automatically generated with medium confidence

Also, in the HTML I added a bootstrap modal, that is going to be toggled when someone wins, or there is a tie. I saw this modal in the Full Stack Web Development book by Edwin R. Torres and I decided to implement it in my game as an alert of who won.

To decide who starts the game I implemented a randomPlayer function that randomly generetes a 0 or 1 using Math.Random. In the whole program, when num == 0, then it means that it is User’s turn. If num == 1, then is the computer’s turn:

Text

Description automatically generated

To decide whether the user or the player win, I used an array that holds all the possible winning solutions. I took this idea from one of the videos shown in the lecture on Monday February 7th:

Text

Description automatically generated

I used a 2D array that holds the tiles picked by user (X) and computer (O).

At index 0 would hold the values of the user, index 1 would hold the values of computer.

The function checkWin() would basically check if the moves of either the computer or the user would match any array of the winning solution. To compare if an array is a subset of another array, I used an algorithm taken from [GeekForGeeks](https://www.geeksforgeeks.org/find-whether-an-array-is-subset-of-another-array-set-1/), all credits to them.

One thing that was tedious and took me long to figure out was to boardClicked function and to get the id of the element clicked, so I could push() it to the move array. It was hard because later I had to disable so the user could not choose that tile again. I used the .click method from jQuery to use the boardClick function when the div element was clicked. I used the .off attribute to disable the click from the given clicked element.

For hovering, I added a class to each cell called “hover\_cell”. I would enable or disable using the removeClass() and addClass() attribute of JQuery, so each time a cell was clicked, so it would not hover it since the tile was taken.

I used an arrow function in order to enable the modal when there is a win or there is a tie: 

Arrow functions is a easy and simple way to create functions.

I used for loops in a ciple functions in order to traverse arrays or the 2d array that hold the moves from the user and the player. For example, in the checkWin() function. The first loop just traverses the 2d Array. Index 0 is the array that holds the user’s move, Index 1 is the array that holds the computer moves. Then it has a nested for loop which is going to traverse the array that holds all possible winning conditions. It is going to compare each of them to the user or computer array. If the winner array is a subset array of the user or computer move array, then there is a win.

Text

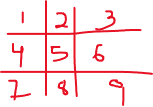
Description automatically generated

To highlight the cells that caused the win, I just created a function highlightWinTiles(win\_array) where the array that caused the win as an argument.

Text

Description automatically generated

I forgot to mention that each cell div has its own id, 1 through 9:



The README file has all functions and their explanations. The program has also a lot of comments so the code could be understanded better. The only thing that I’m not happy about of my game is that when you click a cell on the board, the cell resizes. I tried fixing this but I was not able to. There might be easier ways to implement this tic tac toe, but I used my current knowledge to implement it. Also, more functions could be created in order for the code to be more readable.