Mackenzie Falla

10/08/2024

Link: https://lamp.cse.fau.edu/~mfalla2020/p4/index.html

Tic-Tac-Toe Game Project Report

This report outlines the process I followed to develop a fully functional Tic-Tac-Toe game using JavaScript, jQuery, HTML, and CSS, based on code provided in a jsFiddle example. The game was then enhanced and modified to suit the project requirements.

Step 1: Starting with the jsFiddle Example

As directed, I began by accessing the provided link: https://jsfiddle.net/rtoal/ThPEH/. From this website, I copied the pre-existing JavaScript code and pasted it into Phoenix Code for further editing. My first step was to translate the JavaScript code into an HTML file.

To build the HTML structure, I included a header to define the document type and linked the necessary CSS and JavaScript files. The game's main content was placed inside the <body>, where I used a container to house the Tic-Tac-Toe board, a status message, and a restart button.

Step 2: Understanding and Modifying the JavaScript

The core game logic was written in JavaScript and largely remained unchanged, though I made several key modifications. The original code, found on the jsFiddle website, implemented a jQuery function to ensure that the game initializes only after the Document Object Model (DOM) has fully loaded.

The game was structured for a 3x3 grid with an EMPTY variable representing vacant squares. Additional variables were defined, such as score to track player scores and moves to count the number of moves made. If no empty squares remain and there is no winner, the game results in a tie. The game begins with X's turn (turn = X), and gameOver = false indicates that the game is ongoing.

Initially, I had difficulty understanding why the coder wrote out the possible winning combinations. After consulting ChatGBT, I learned that the array represented bitwise combinations of all possible winning configurations on the 3x3 grid. This array allows the program to check for winning moves efficiently.

Step 3: Resetting the Game

I added a startNewGame variable to reset the game, allowing the players to start a fresh round without refreshing the page. When the game resets, the turn variable is set to X, and the board's squares are cleared by setting each square's content to EMPTY.

Step 4: Handling Player Moves and Scoring

Understanding the scoring logic required me to play the game multiple times. With help from ChatGBT, I researched how the game detects a winner. The program loops through each of the possible bitwise combinations, checking if a winning sequence has been achieved. If no such combination is found, the game returns a "no winner" state.

The set function is triggered when a player clicks on any square. It checks whether the square is empty or if the game has ended. If valid, it updates the square with an X or O, depending on the current player's turn, increments the move count, and updates the score. If all moves are made and no winner is detected, a message displaying "It's a tie! Try Again!" is shown, which I added as a personal touch.

Step 5: Adding a Restart Button

Another key modification I made was adding a restart button. In the original code, the page would refresh automatically once the game ended. I found this behavior inconvenient, so I allowed players to manually restart the game with the click of a button, providing a more interactive experience.

Step 6: Final Touches with CSS

For the visual styling, I edited the CSS file. I customized the background colors for both the page and the game board, adjusted the font styles and margins, and added a hover effect to the squares, ensuring that the player always knows where the mouse is. Additionally, I resized and centered the game board for a clean and user-friendly layout.

Tools and Resources

Throughout the project, I used Phoenix Code to write and edit the files, referenced the jsFiddle example as a starting point, and relied on ChatGBT to clarify specific parts of the code I found challenging.

In conclusion, the project allowed me to explore both the logic and design aspects of building a functional web-based game, while refining my skills in JavaScript, jQuery, HTML, and CSS. The final product is a dynamic and engaging Tic-Tac-Toe game that allows players to play multiple rounds seamlessly.