PROMINEO TECH

Front End Technologies Week 5 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

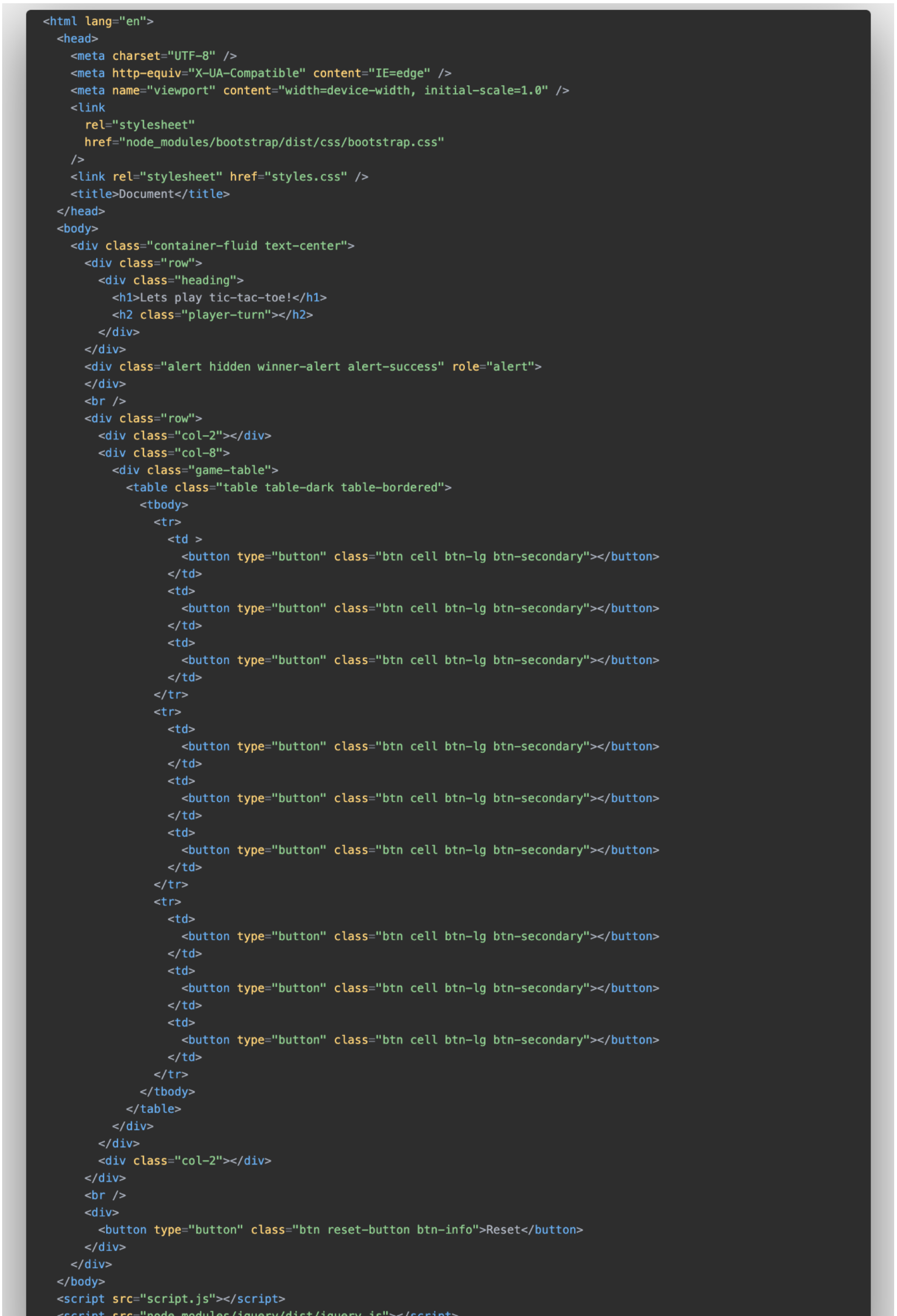
Instructions: In VS Code, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your JavaScript project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

1. Using any of the tools you've worked with so far, create a game of tic-tac-toe.
    a. A heading should say whether it is X's or O's turn and change with each move made.
    b. Create a tic-tac-toe grid using your HTML element of choice. When a cell in the grid is clicked, an X or O should appear in that spot depending on whose turn it is.
    c. A button should be available to clear the grid and restart the game.
    d. When a player has won, or the board is full and the game results in a draw, a Bootstrap alert or similar Bootstrap component should appear across the screen announcing the winner.

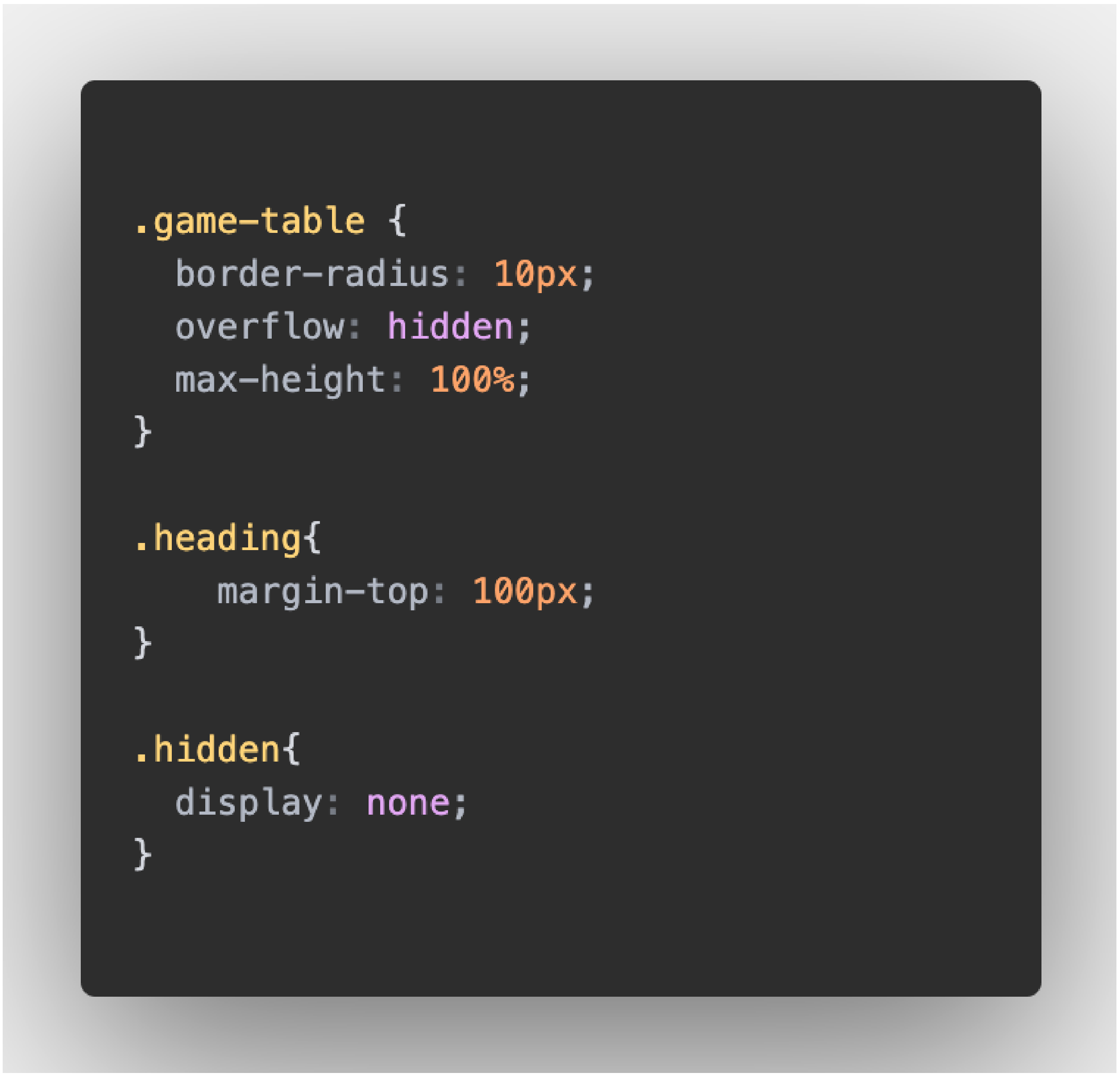
Screenshots of Code:

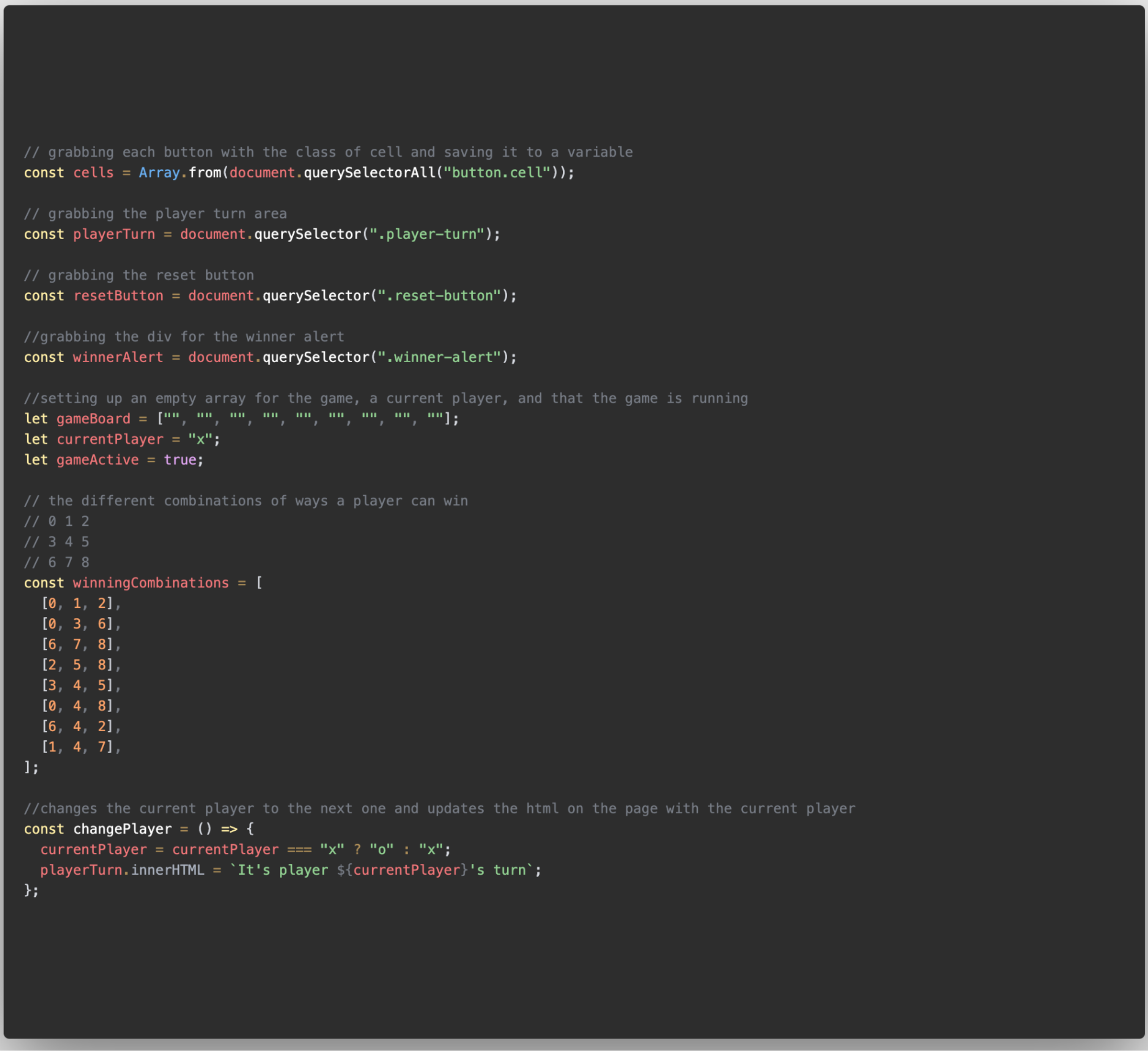
PROMINEO TECH



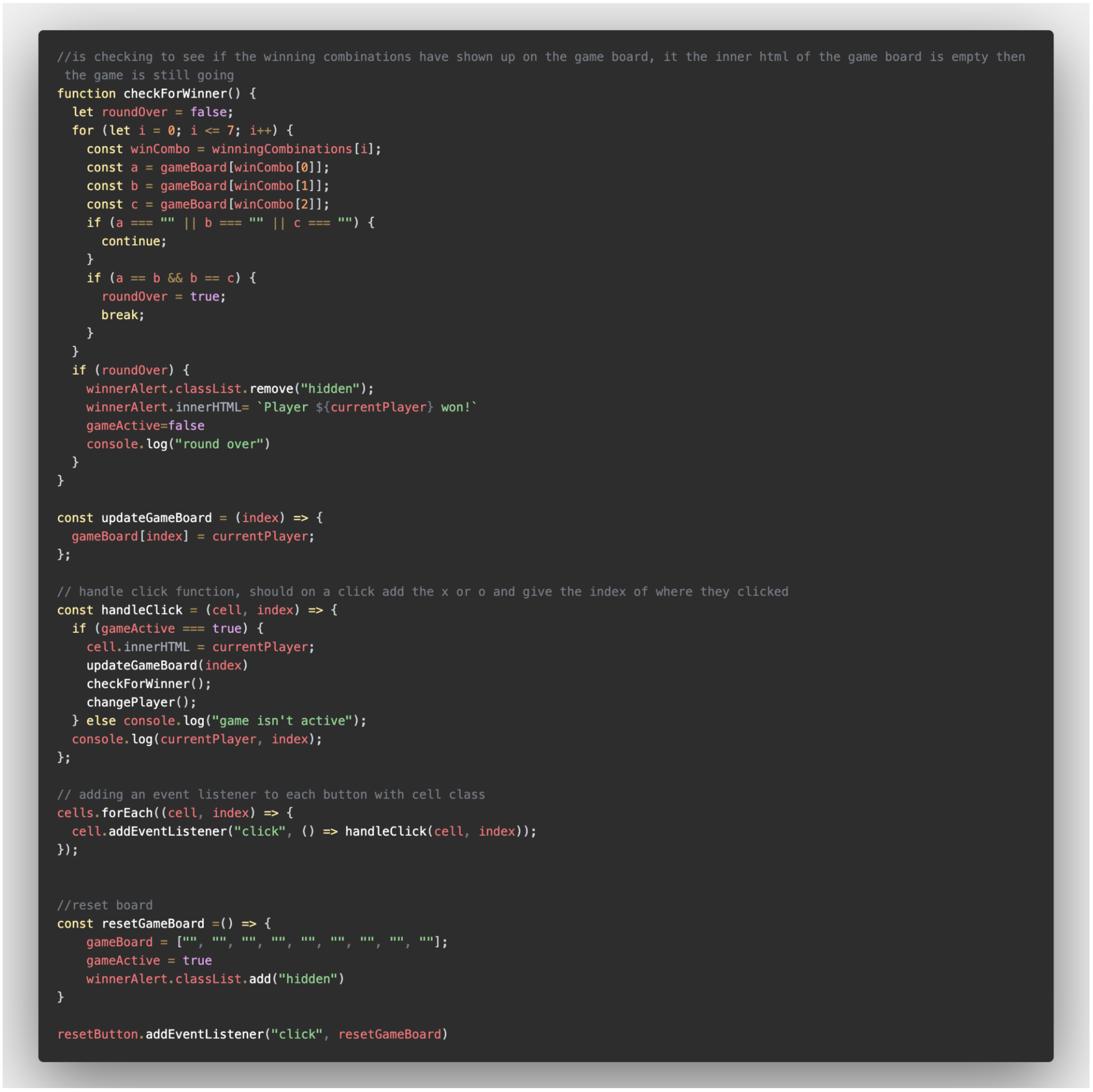
```html
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      rel="stylesheet"
      href="node_modules/bootstrap/dist/css/bootstrap.css"
    />
    <link rel="stylesheet" href="styles.css" />
    <title>Document</title>
  </head>
  <body>
    <div class="container-fluid text-center">
      <div class="row">
        <div class="heading">
          <h1>Lets play tic-tac-toe!</h1>
          <h2 class="player-turn"></h2>
        </div>
      </div>
      <div class="alert hidden winner-alert alert-success" role="alert">
      </div>
      <br />
      <div class="row">
        <div class="col-2"></div>
        <div class="col-8">
          <div class="game-table">
            <table class="table table-dark table-bordered">
              <tbody>
                <tr>
                  <td >
                    <button type="button" class="btn cell btn-lg btn-secondary"></button>
                  </td>
                  <td>
                    <button type="button" class="btn cell btn-lg btn-secondary"></button>
                  </td>
                  <td>
                    <button type="button" class="btn cell btn-lg btn-secondary"></button>
                  </td>
                </tr>
                <tr>
                  <td>
                    <button type="button" class="btn cell btn-lg btn-secondary"></button>
                  </td>
                  <td>
                    <button type="button" class="btn cell btn-lg btn-secondary"></button>
                  </td>
                  <td>
                    <button type="button" class="btn cell btn-lg btn-secondary"></button>
                  </td>
                </tr>
                <tr>
                  <td>
                    <button type="button" class="btn cell btn-lg btn-secondary"></button>
                  </td>
                  <td>
                    <button type="button" class="btn cell btn-lg btn-secondary"></button>
                  </td>
                  <td>
                    <button type="button" class="btn cell btn-lg btn-secondary"></button>
                  </td>
                </tr>
              </tbody>
            </table>
          </div>
        </div>
        <div class="col-2"></div>
      </div>
      <br />
      <div>
        <button type="button" class="btn reset-button btn-info">Reset</button>
      </div>
    </div>
  </body>
  <script src="script.js"></script>
  <script src="node_modules/jquery/dist/jquery.js"></script>
```

PROMINEO TECH


```css
.game-table {
  border-radius: 10px;
  overflow: hidden;
  max-height: 100%;
}

.heading{
    margin-top: 100px;
}

.hidden{
  display: none;
}
```

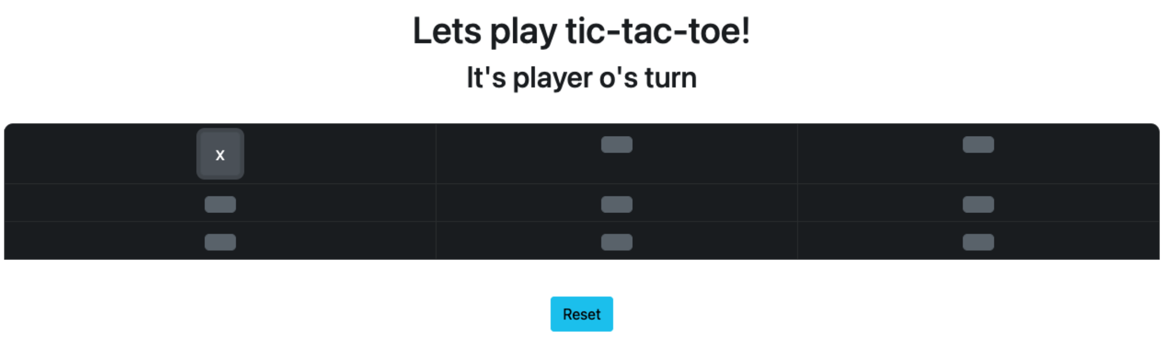


```javascript
// grabbing each button with the class of cell and saving it to a variable
const cells = Array.from(document.querySelectorAll("button.cell"));

// grabbing the player turn area
const playerTurn = document.querySelector(".player-turn");

// grabbing the reset button
const resetButton = document.querySelector(".reset-button");

//grabbing the div for the winner alert
const winnerAlert = document.querySelector(".winner-alert");

//setting up an empty array for the game, a current player, and that the game is running
let gameBoard = ["", "", "", "", "", "", "", "", ""];
let currentPlayer = "x";
let gameActive = true;

// the different combinations of ways a player can win
// 0 1 2
// 3 4 5
// 6 7 8
const winningCombinations = [
  [0, 1, 2],
  [0, 3, 6],
  [6, 7, 8],
  [2, 5, 8],
  [3, 4, 5],
  [0, 4, 8],
  [6, 4, 2],
  [1, 4, 7],
];

//changes the current player to the next one and updates the html on the page with the current player
const changePlayer = () => {
  currentPlayer = currentPlayer === "x" ? "o" : "x";
  playerTurn.innerHTML = `It's player ${currentPlayer}'s turn`;
};
```



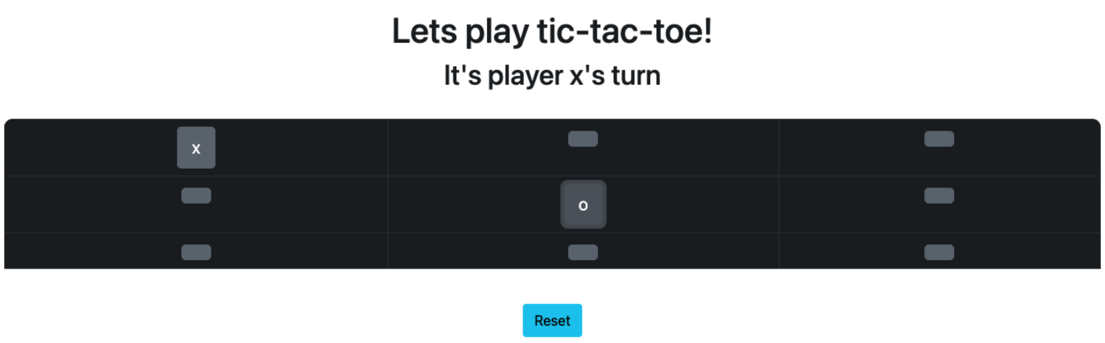
```
//is checking to see if the winning combinations have shown up on the game board, it the inner html of the game board is empty then
 the game is still going
function checkForWinner() {
  let roundOver = false;
  for (let i = 0; i <= 7; i++) {
    const winCombo = winningCombinations[i];
    const a = gameBoard[winCombo[0]];
    const b = gameBoard[winCombo[1]];
    const c = gameBoard[winCombo[2]];
    if (a === "" || b === "" || c === "") {
      continue;
    }
    if (a == b && b == c) {
      roundOver = true;
      break;
    }
  }
  if (roundOver) {
    winnerAlert.classList.remove("hidden");
    winnerAlert.innerHTML= `Player ${currentPlayer} won!`
    gameActive=false
    console.log("round over")
  }
}

const updateGameBoard = (index) => {
  gameBoard[index] = currentPlayer;
};

// handle click function, should on a click add the x or o and give the index of where they clicked
const handleClick = (cell, index) => {
  if (gameActive === true) {
    cell.innerHTML = currentPlayer;
    updateGameBoard(index)
    checkForWinner();
    changePlayer();
  } else console.log("game isn't active");
  console.log(currentPlayer, index);
};

// adding an event listener to each button with cell class
cells.forEach((cell, index) => {
  cell.addEventListener("click", () => handleClick(cell, index));
});


//reset board
const resetGameBoard =() => {
    gameBoard = ["", "", "", "", "", "", "", "", ""];
    gameActive = true
    winnerAlert.classList.add("hidden")
}

resetButton.addEventListener("click", resetGameBoard)
```

Screenshots of Running Application:


Lets play tic-tac-toe!

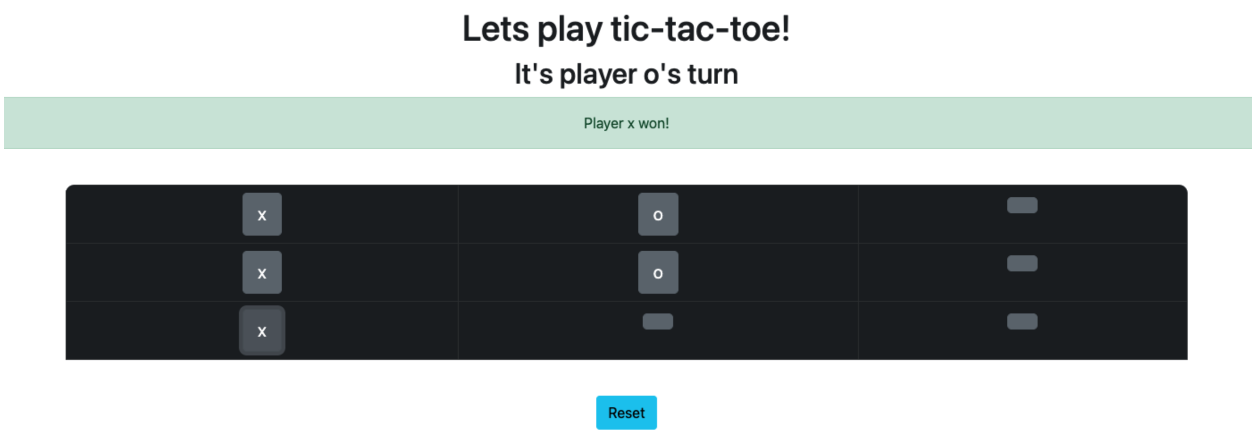
It's player o's turn

x		


Lets play tic-tac-toe!

It's player x's turn

x		
	o	


Lets play tic-tac-toe!

It's player o's turn

Player x won!

x	o	
x	o	
x		

URL to GitHub Repository:

https://github.com/DanielleByrne/tic-tac-toe