

Module 4 - Tic Tac Toe Assignment - Model Answer

```
<!--TicTacToe.html-->

<!DOCTYPE html>
<html lang='en'>

  <head>
    <title>Tic Tac Toe Game</title>
    <meta charset='UTF-8'>
    <meta name='viewport' content='width=device-width, initial-scale=1.0'>
    <link rel='stylesheet' href='css/game.css'>
  </head>

  <body id='body'>
    <div class='center-container'>
      <h1>Tic Tac Toe</h1>
      <canvas id='win-lines' width='608' height='608'></canvas>
      <table>
        <tr>
          <td id='0' onclick='placeXOrO("0")'></td>
          <td id='1' onclick='placeXOrO("1")'></td>
          <td id='2' onclick='placeXOrO("2")'></td>
        </tr>
        <tr>
          <td id='3' onclick='placeXOrO("3")'></td>
          <td id='4' onclick='placeXOrO("4")'></td>
          <td id='5' onclick='placeXOrO("5")'></td>
        </tr>
        <tr>
          <td id='6' onclick='placeXOrO("6")'></td>
          <td id='7' onclick='placeXOrO("7")'></td>
          <td id='8' onclick='placeXOrO("8")'></td>
        </tr>
      </table>
    </div>
    <script src='js/tictactoe.js'></script>
  </body>
</html>
```

```
/*game.css*/

.center-container {
  width: 608px;
  margin: 0 auto;
  margin-top: 50px;
}

table {
  border-collapse: collapse;
  border-style: hidden;
}

td {
  cursor: pointer;
  border: 4px solid #008000;
  padding: 0px;
  width: 200px;
  height: 200px;
  filter: contrast(3);
}

canvas {
  position: absolute;
  z-index: 10;
  pointer-events: none;
}

h1 {
  text-align: center;
}
```

```

//tictactoe.js

//Variable to keep track of whose turn it is
let activePlayer = 'X';

//Array to store moves - use this to determine win conditions
let selectedSquares = [];

//Function to place x or o in a square
function placeXOrO(squareNumber) {
    //checks if the square has been selected already
    if (!selectedSquares.some(element => element.includes(squareNumber))) {
        //Variable to hold the HTML element that was clicked
        let select = document.getElementById(squareNumber);
        //Determines the active player and places the icon
        if (activePlayer === 'X') {
            select.style.backgroundImage = 'url("images/x.png")';
        } else {
            select.style.backgroundImage = 'url("images/o.png")';
        }
        //Adds the square number and player to the array
        selectedSquares.push(squareNumber + activePlayer);
        //Calls the function to check for a win
        checkWinConditions();
        //Changes the active player
        if (activePlayer === 'X') {
            activePlayer = 'O';
        } else {
            activePlayer = 'X';
        }
        //Function to play the placement sound
        audio('./media/place.mp3');
        //Checks if it is the computers turn
        if (activePlayer === 'O') {
            disableClick();
            setTimeout(function () { computersTurn(); }, 1000);
        }
        //Returning true is needed for the computersTurn() function
        return true;
    }
}

```

```

//Picks a random square for the computers turn
function computersTurn() {
    let success = false;
    let pickASquare;
    while (!success) {
        pickASquare = string(Math.floor(Math.random() * 9));
        if (placeXorO(pickASquare)) {
            placeXorO(pickASquare);
            success = true;
        };
    }
}

```

//This function parses the selectedSquares array to determine if a player has won
 //The drawLine function is called if a win condition is met

```

function checkWinConditions() {
    if (arrayIncludes('0X', '1X', '2X')) { drawWinLine(50, 100, 558, 100) }
    else if (arrayIncludes('3X', '4X', '5X')) { drawWinLine(50, 304, 558, 304) }
    else if (arrayIncludes('6X', '7X', '8X')) { drawWinLine(50, 508, 558, 508) }
    else if (arrayIncludes('0X', '3X', '6X')) { drawWinLine(100, 50, 100, 558) }
    else if (arrayIncludes('1X', '4X', '7X')) { drawWinLine(304, 50, 304, 558) }
    else if (arrayIncludes('2X', '5X', '8X')) { drawWinLine(508, 50, 508, 558) }
    else if (arrayIncludes('6X', '4X', '2X')) { drawWinLine(100, 508, 510, 90) }
    else if (arrayIncludes('0X', '4X', '8X')) { drawWinLine(100, 100, 520, 520) }
    else if (arrayIncludes('00', '10', '20')) { drawWinLine(50, 100, 558, 100) }
    else if (arrayIncludes('30', '40', '50')) { drawWinLine(50, 304, 558, 304) }
    else if (arrayIncludes('60', '70', '80')) { drawWinLine(50, 508, 558, 508) }
    else if (arrayIncludes('00', '30', '60')) { drawWinLine(100, 50, 100, 558) }
    else if (arrayIncludes('10', '40', '70')) { drawWinLine(304, 50, 304, 558) }
    else if (arrayIncludes('20', '50', '80')) { drawWinLine(508, 50, 508, 558) }
    else if (arrayIncludes('60', '40', '20')) { drawWinLine(100, 508, 510, 90) }
    else if (arrayIncludes('00', '40', '80')) { drawWinLine(100, 100, 520, 520) }
    //checks for a tie - if no win conditions are met and 9 squares have been selected
    else if (selectedSquares.length >= 9) {
        //plays the tie sound
        audio('./media/tie.mp3');
        //resets the game after a tie
        setTimeout(function () { resetGame(); }, 500);
    }
}

```

```
//This function checks for each win condition
function arrayIncludes(squareA, squareB, squareC) {
    const a = selectedSquares.includes(squareA);
    const b = selectedSquares.includes(squareB);
    const c = selectedSquares.includes(squareC);
    if (a === true && b === true && c === true) { return true; }
}
}
```

```
//Clears the board and the array to restart the game
function resetGame() {
    for (let i = 0; i < 9; i++) {
        let square = document.getElementById(String(i));
        square.style.backgroundImage = '';
    }
    selectedSquares = [];
}
```

```
//Plays the audio files
function audio(audioURL) {
    let audio = new Audio(audioURL);
    audio.play();
}
```

```
//Function to draw the line across winning coordinates
function drawWinLine(coordX1, coordY1, coordX2, coordY2) {
    const canvas = document.getElementById('win-lines');
    const c = canvas.getContext('2d');
    let x1 = coordX1,
        y1 = coordY1,
        x2 = coordX2,
        y2 = coordY2,
        x = x1,
        y = y1;
```

```

function animateLineDrawing() {
  const animationLoop = requestAnimationFrame(animateLineDrawing);
  c.clearRect(0, 0, 608, 608);
  c.beginPath();
  c.moveTo(x1, y1);
  c.lineTo(x, y);
  c.lineWidth = 10;
  c.strokeStyle = 'rgba(70, 255, 33, .8)';
  c.stroke();
  if (x1 <= x2 && y1 <= y2) {
    if (x < x2) { x += 10; }
    if (y < y2) { y += 10; }
    if (x >= x2 && y >= y2) { cancelAnimationFrame(animationLoop); }
  }
  if (x1 <= x2 && y1 >= y2) {
    if (x < x2) { x += 10; }
    if (y > y2) { y -= 10; }
    if (x >= x2 && y <= y2) { cancelAnimationFrame(animationLoop); }
  }
}

```

//Clears the board after the animation

```

function clear() {
  const animationLoop = requestAnimationFrame(clear);
  c.clearRect(0, 0, 608, 608);
  cancelAnimationFrame(animationLoop);
}
disableClick();
audio('./media/winGame.mp3');
animateLineDrawing();
setTimeout(function () { clear(); resetGame(); }, 1000);
}

```

//Disables click during the computer's turn

```

function disableClick() {
  body.style.pointerEvents = 'none';
  setTimeout(function () { body.style.pointerEvents = 'auto'; }, 1000);
}

```