PROMINEO TECH

Front End Technologies Week 11 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.

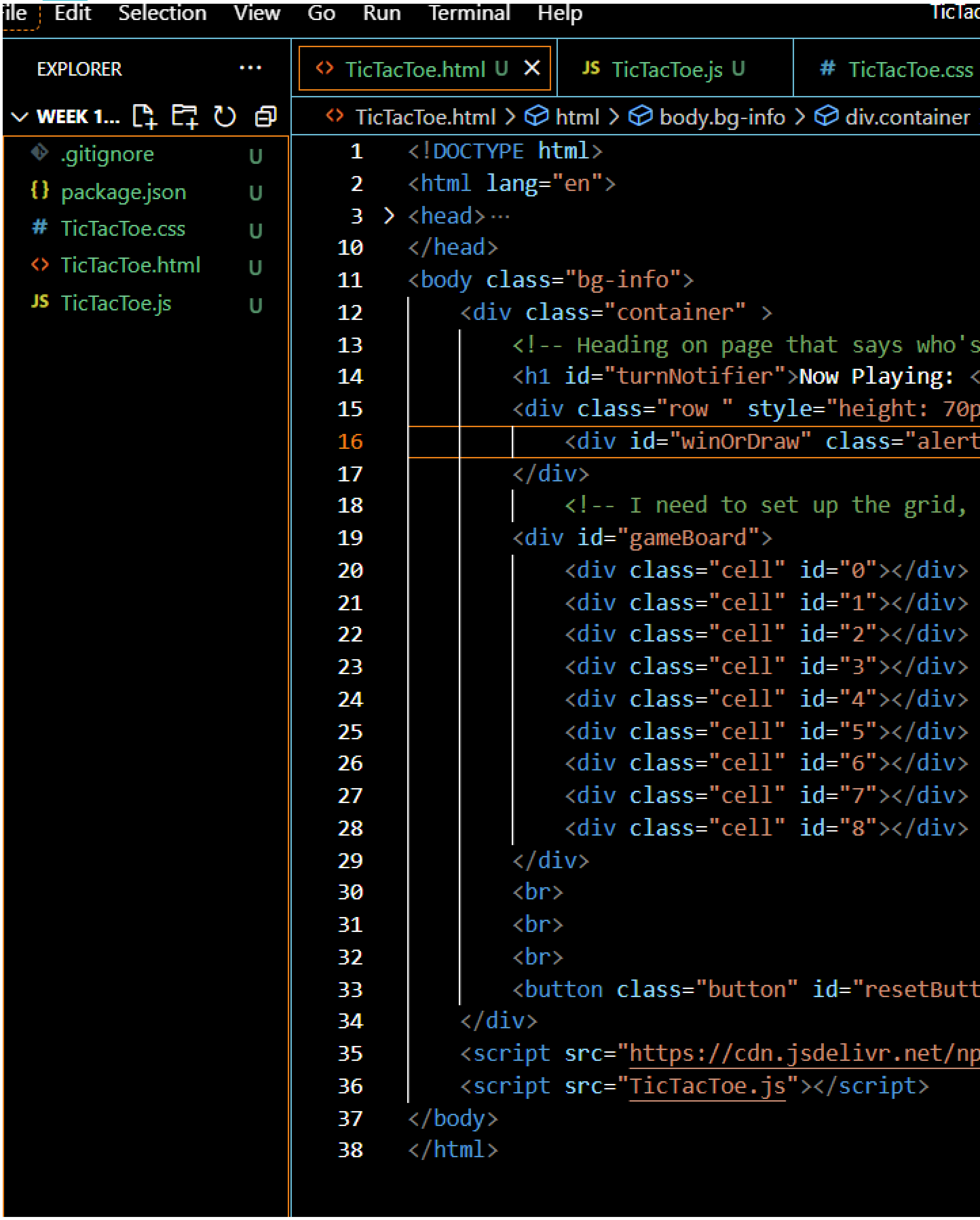
Coding Steps:

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:

HTML

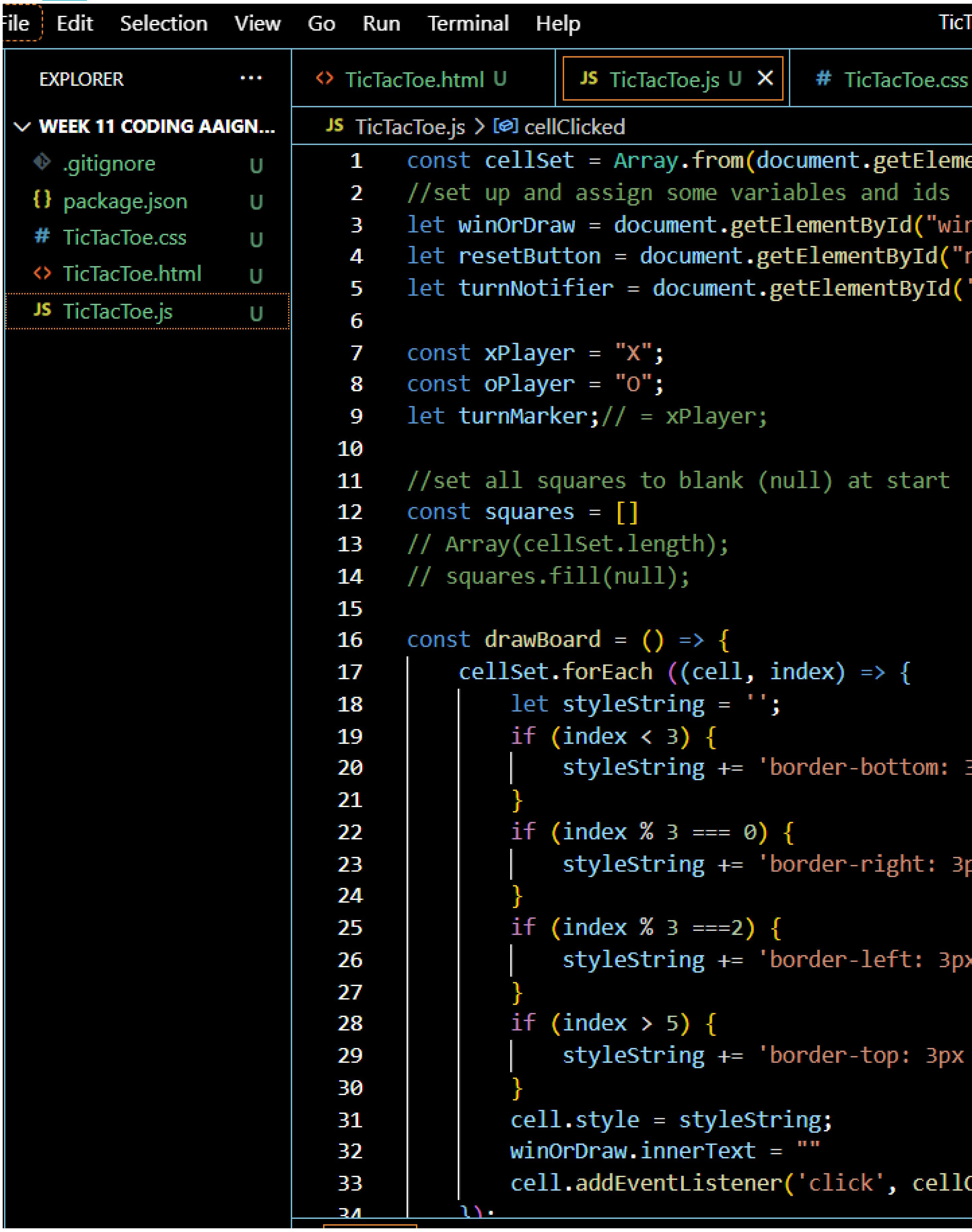
PROMINEO TECH



```
<!DOCTYPE html>
<html lang="en">
<head>···
</head>
<body class="bg-info">
    <div class="container" >
        <!-- Heading on page that says who's
        <h1 id="turnNotifier">Now Playing: <
        <div class="row " style="height: 70p
            <div id="winOrDraw" class="alert
        </div>
            <!-- I need to set up the grid,
        <div id="gameBoard">
            <div class="cell" id="0"></div>
            <div class="cell" id="1"></div>
            <div class="cell" id="2"></div>
            <div class="cell" id="3"></div>
            <div class="cell" id="4"></div>
            <div class="cell" id="5"></div>
            <div class="cell" id="6"></div>
            <div class="cell" id="7"></div>
            <div class="cell" id="8"></div>
        </div>
        <br>
        <br>
        <br>
        <button class="button" id="resetButt
    </div>
    <script src="https://cdn.jsdelivr.net/np
    <script src="TicTacToe.js"></script>
</body>
</html>
```

Java

PROMINEO TECH



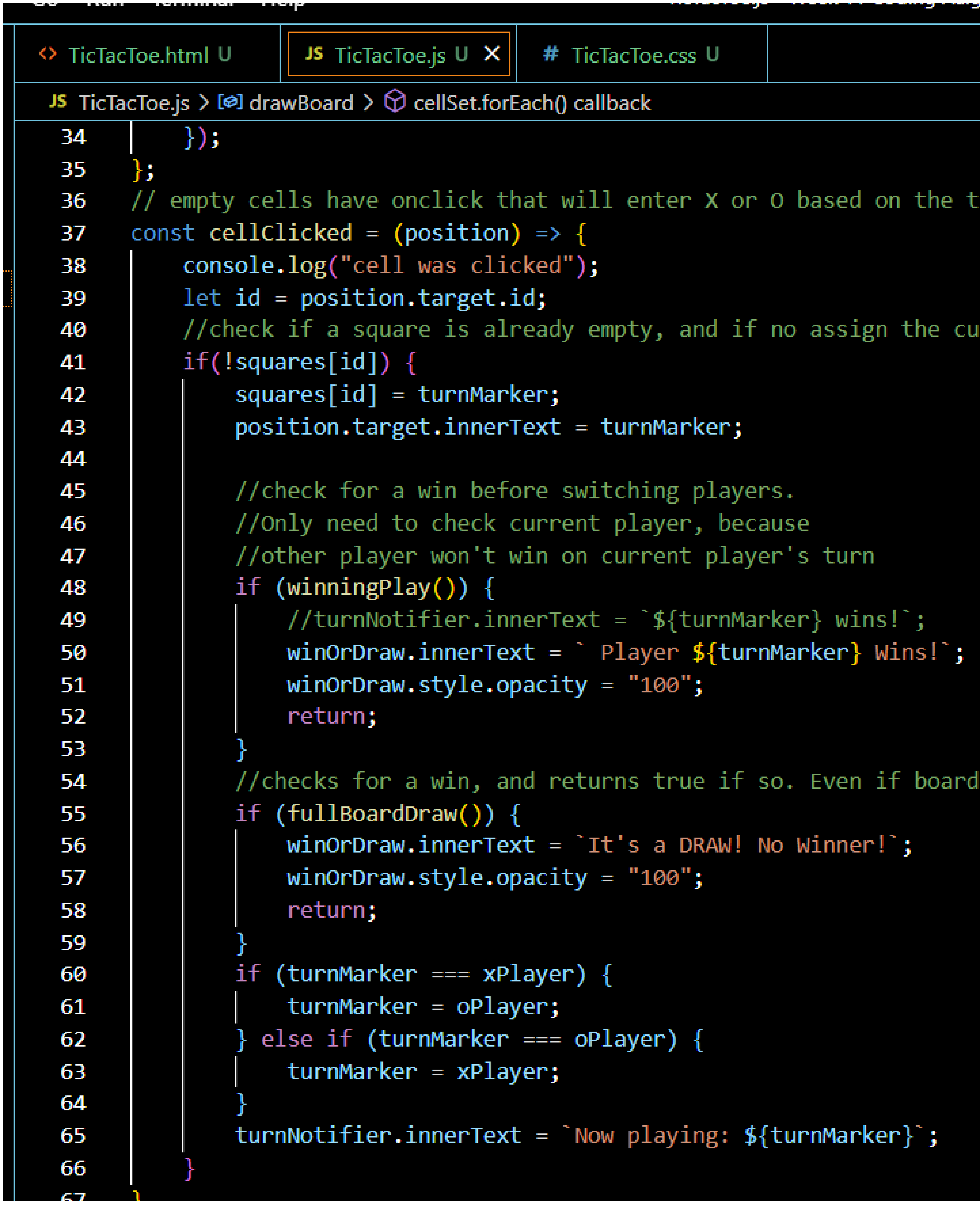
File Edit Selection View Go Run Terminal Help

EXPLORER
WEEK 11 CODING AAIGN...
.gitignore U
package.json U
TicTacToe.css U
TicTacToe.html U
TicTacToe.js U

TicTacToe.html U | TicTacToe.js U | TicTacToe.css

TicTacToe.js > cellClicked

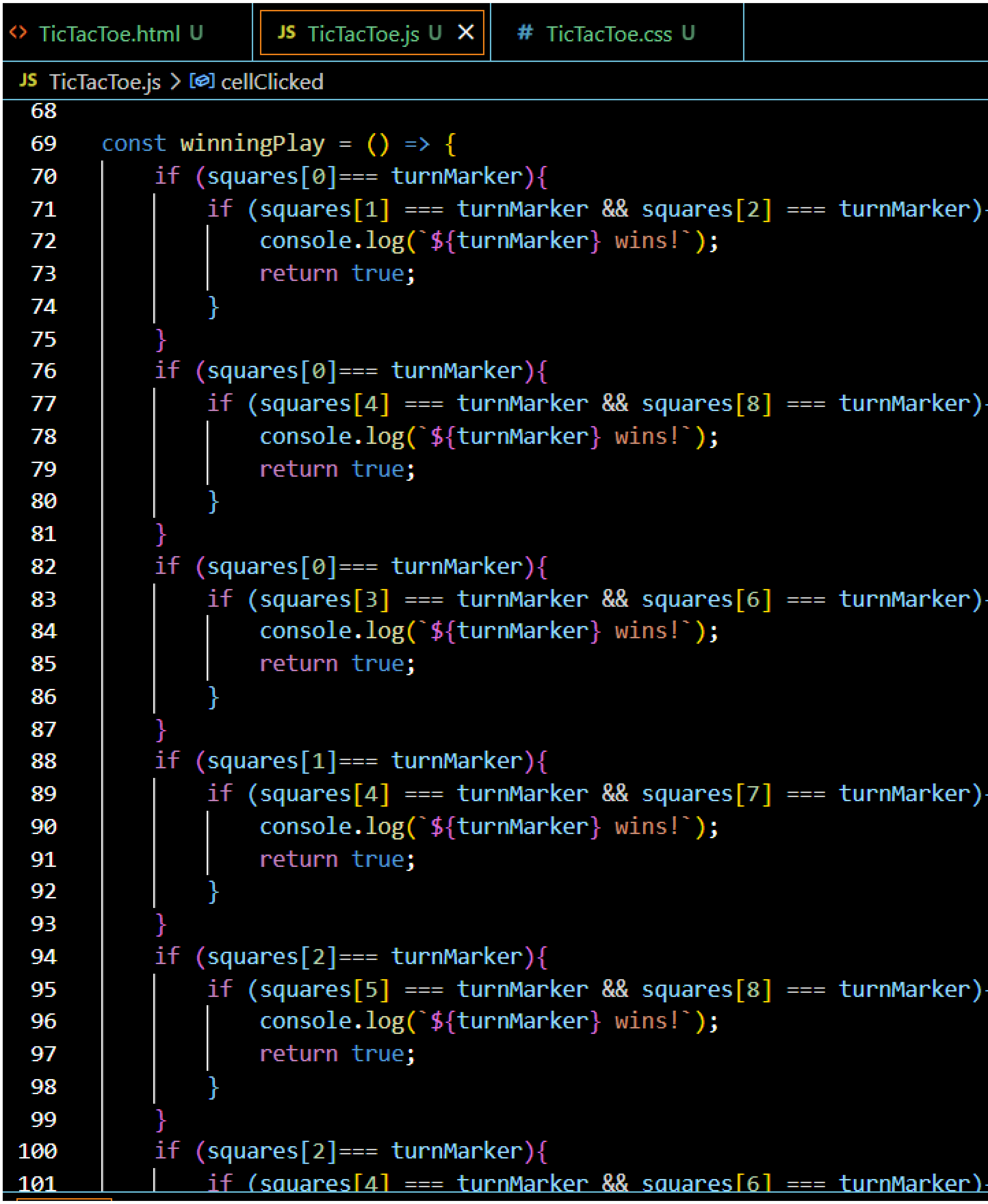
```
const cellSet = Array.from(document.getEleme
//set up and assign some variables and ids
let winOrDraw = document.getElementById("wir
let resetButton = document.getElementById("r
let turnNotifier = document.getElementById('

const xPlayer = "X";
const oPlayer = "O";
let turnMarker;// = xPlayer;

//set all squares to blank (null) at start
const squares = []
// Array(cellSet.length);
// squares.fill(null);

const drawBoard = () => {
    cellSet.forEach ((cell, index) => {
        let styleString = '';
        if (index < 3) {
            styleString += 'border-bottom: 3
        }
        if (index % 3 === 0) {
            styleString += 'border-right: 3p
        }
        if (index % 3 ===2) {
            styleString += 'border-left: 3px
        }
        if (index > 5) {
            styleString += 'border-top: 3px
        }
        cell.style = styleString;
        winOrDraw.innerText = ""
        cell.addEventListener('click', cellC
    });
```



TicTacToe.html U | TicTacToe.js U | TicTacToe.css U

TicTacToe.js > drawBoard > cellSet.forEach() callback

```
    });
};
// empty cells have onclick that will enter X or O based on the t
const cellClicked = (position) => {
    console.log("cell was clicked");
    let id = position.target.id;
    //check if a square is already empty, and if no assign the cu
    if(!squares[id]) {
        squares[id] = turnMarker;
        position.target.innerText = turnMarker;

        //check for a win before switching players.
        //Only need to check current player, because
        //other player won't win on current player's turn
        if (winningPlay()) {
            //turnNotifier.innerText = `${turnMarker} wins!`;
            winOrDraw.innerText = ` Player ${turnMarker} Wins!`;
            winOrDraw.style.opacity = "100";
            return;
        }
        //checks for a win, and returns true if so. Even if board
        if (fullBoardDraw()) {
            winOrDraw.innerText = `It's a DRAW! No Winner!`;
            winOrDraw.style.opacity = "100";
            return;
        }
        if (turnMarker === xPlayer) {
            turnMarker = oPlayer;
        } else if (turnMarker === oPlayer) {
            turnMarker = xPlayer;
        }
        turnNotifier.innerText = `Now playing: ${turnMarker}`;
    }
```

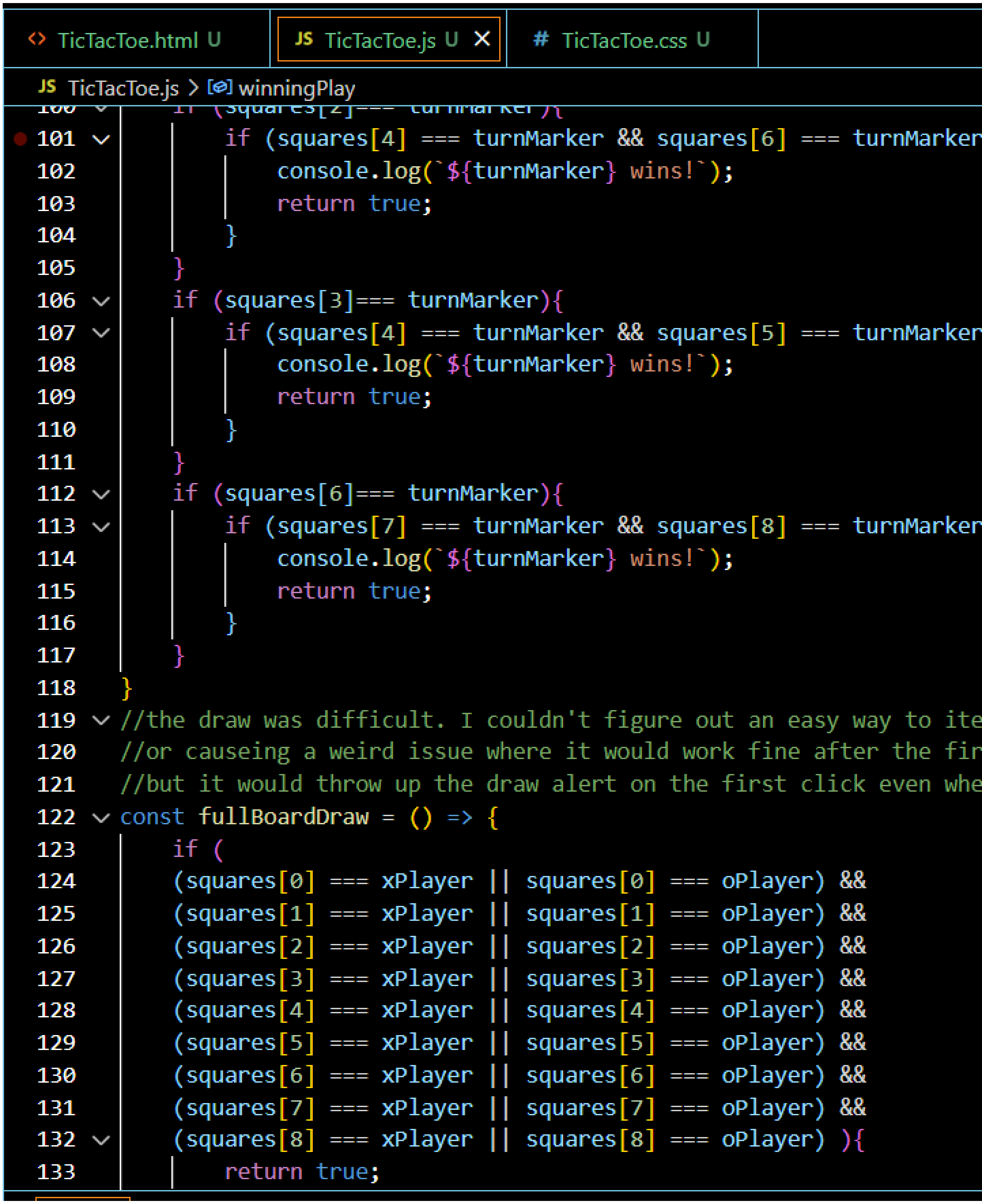


TicTacToe.html U | TicTacToe.js U | TicTacToe.css U

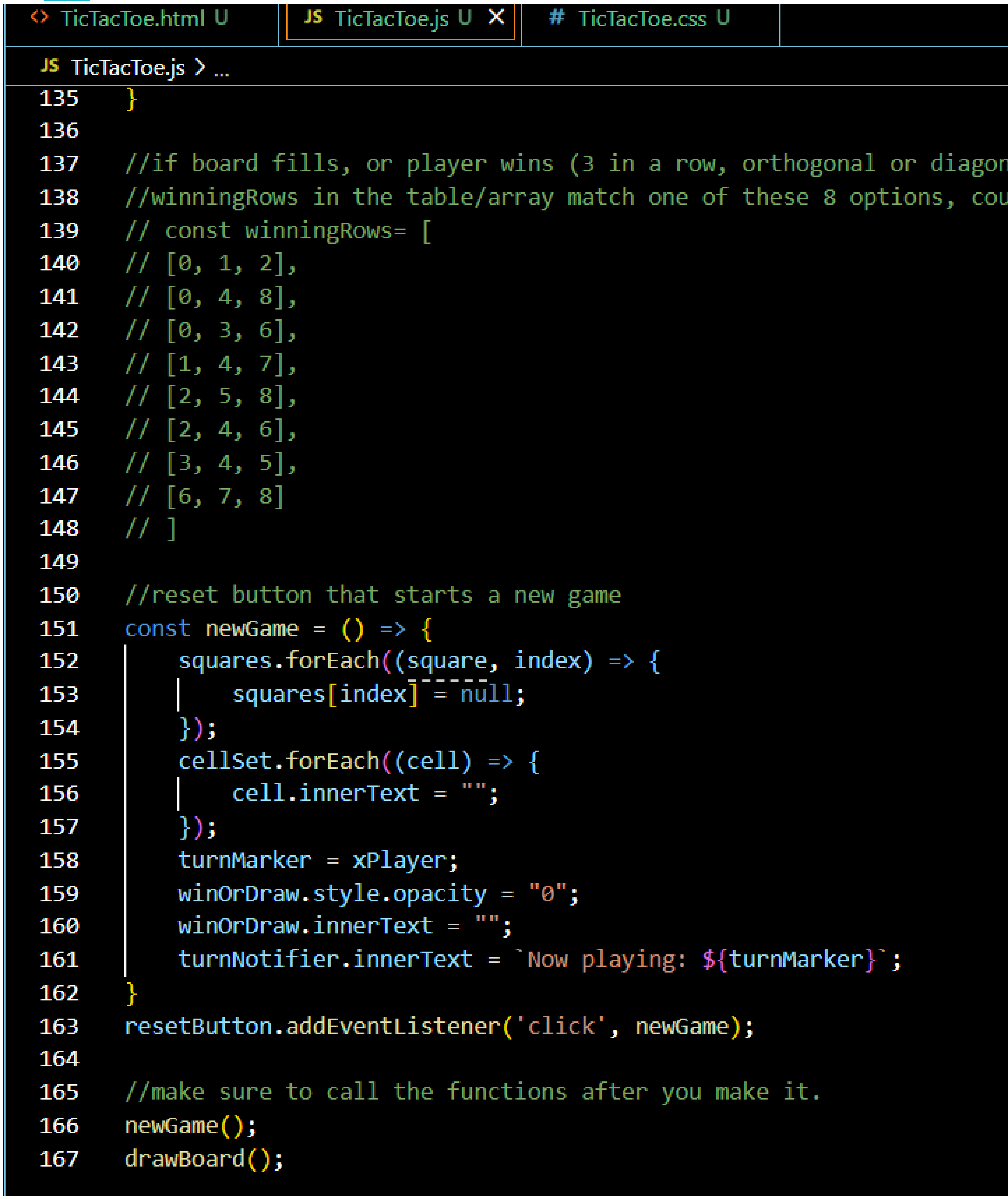
TicTacToe.js > cellClicked

```

const winningPlay = () => {
    if (squares[0]=== turnMarker){
        if (squares[1] === turnMarker && squares[2] === turnMarker)
            console.log(`${turnMarker} wins!`);
            return true;
        }
    }
    if (squares[0]=== turnMarker){
        if (squares[4] === turnMarker && squares[8] === turnMarker)
            console.log(`${turnMarker} wins!`);
            return true;
        }
    }
    if (squares[0]=== turnMarker){
        if (squares[3] === turnMarker && squares[6] === turnMarker)
            console.log(`${turnMarker} wins!`);
            return true;
        }
    }
    if (squares[1]=== turnMarker){
        if (squares[4] === turnMarker && squares[7] === turnMarker)
            console.log(`${turnMarker} wins!`);
            return true;
        }
    }
    if (squares[2]=== turnMarker){
        if (squares[5] === turnMarker && squares[8] === turnMarker)
            console.log(`${turnMarker} wins!`);
            return true;
        }
    }
    if (squares[2]=== turnMarker){
        if (squares[4] === turnMarker && squares[6] === turnMarker)
```

PROMINEO TECH



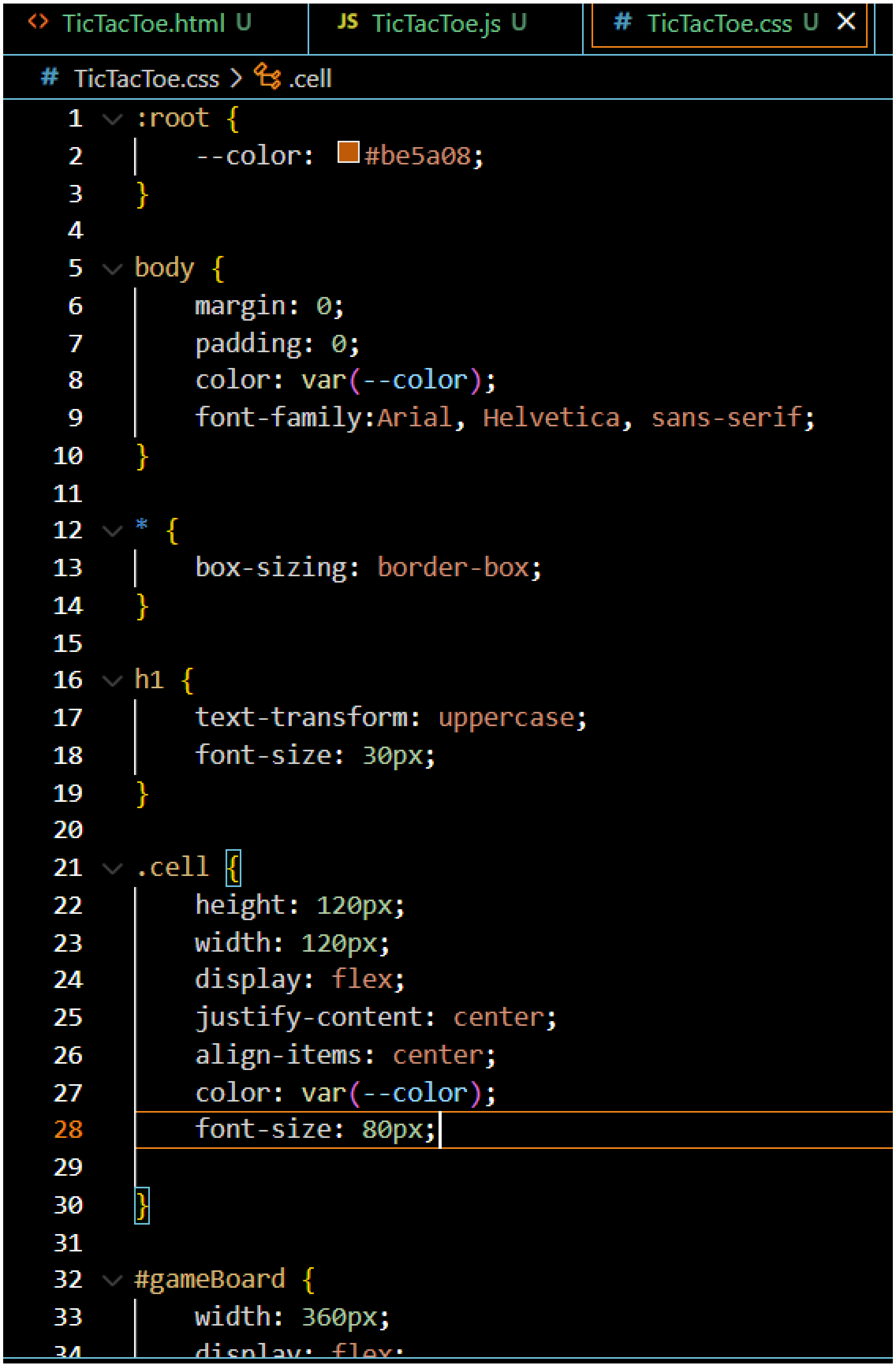
```
    if (squares[2]=== turnMarker){
        if (squares[4] === turnMarker && squares[6] === turnMarker
            console.log(`${turnMarker} wins!`);
            return true;
        }
    }
    if (squares[3]=== turnMarker){
        if (squares[4] === turnMarker && squares[5] === turnMarker
            console.log(`${turnMarker} wins!`);
            return true;
        }
    }
    if (squares[6]=== turnMarker){
        if (squares[7] === turnMarker && squares[8] === turnMarker
            console.log(`${turnMarker} wins!`);
            return true;
        }
    }
}
//the draw was difficult. I couldn't figure out an easy way to ite
//or causeing a weird issue where it would work fine after the fir
//but it would throw up the draw alert on the first click even whe
const fullBoardDraw = () => {
    if (
    (squares[0] === xPlayer || squares[0] === oPlayer) &&
    (squares[1] === xPlayer || squares[1] === oPlayer) &&
    (squares[2] === xPlayer || squares[2] === oPlayer) &&
    (squares[3] === xPlayer || squares[3] === oPlayer) &&
    (squares[4] === xPlayer || squares[4] === oPlayer) &&
    (squares[5] === xPlayer || squares[5] === oPlayer) &&
    (squares[6] === xPlayer || squares[6] === oPlayer) &&
    (squares[7] === xPlayer || squares[7] === oPlayer) &&
    (squares[8] === xPlayer || squares[8] === oPlayer) ){
        return true;
```



```javascript
}

//if board fills, or player wins (3 in a row, orthogonal or diagon
//winningRows in the table/array match one of these 8 options, cou
// const winningRows= [
// [0, 1, 2],
// [0, 4, 8],
// [0, 3, 6],
// [1, 4, 7],
// [2, 5, 8],
// [2, 4, 6],
// [3, 4, 5],
// [6, 7, 8]
// ]

//reset button that starts a new game
const newGame = () => {
    squares.forEach((square, index) => {
        squares[index] = null;
    });
    cellSet.forEach((cell) => {
        cell.innerText = "";
    });
    turnMarker = xPlayer;
    winOrDraw.style.opacity = "0";
    winOrDraw.innerText = "";
    turnNotifier.innerText = `Now playing: ${turnMarker}`;
}
resetButton.addEventListener('click', newGame);

//make sure to call the functions after you make it.
newGame();
drawBoard();
```

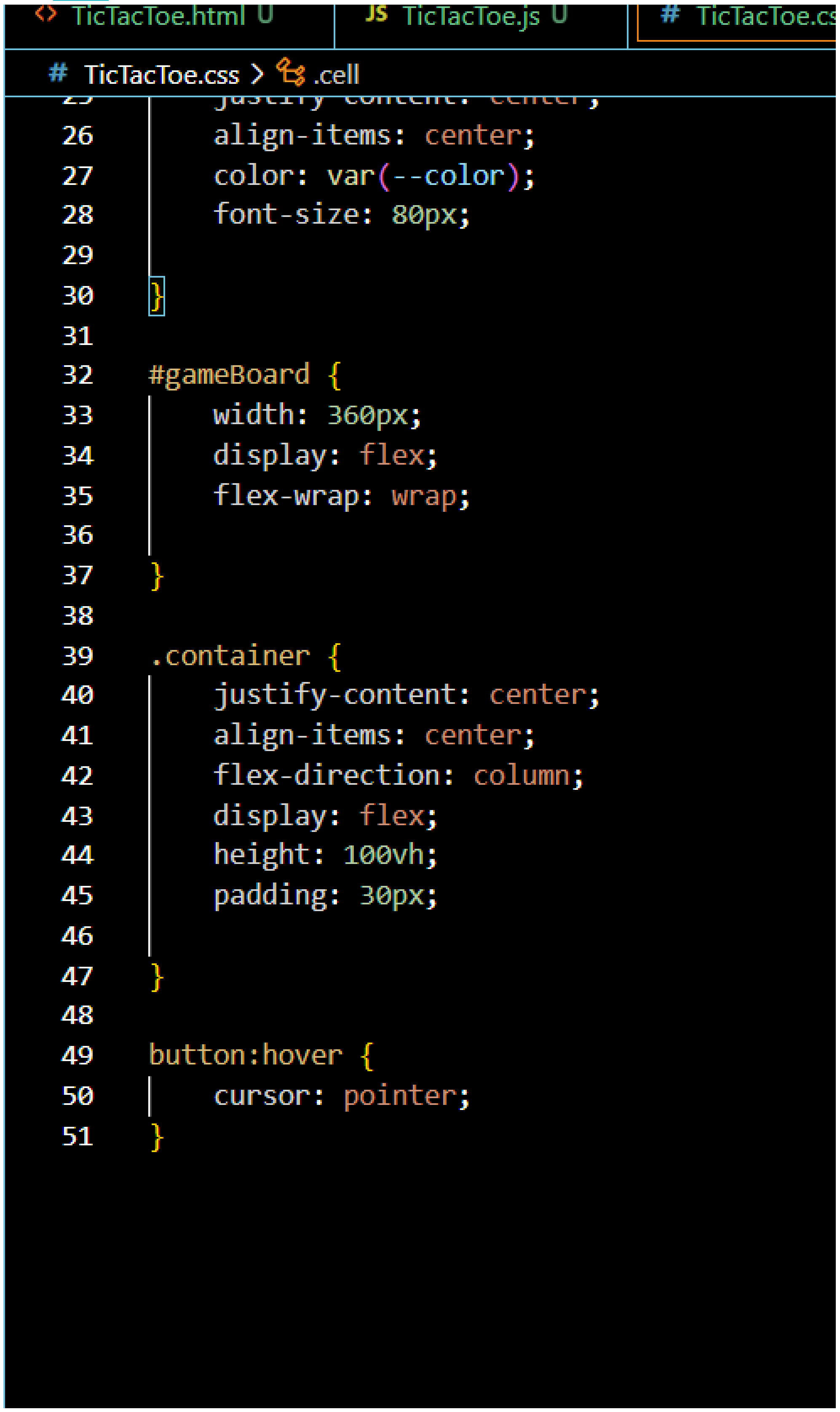
CSS stylesheet (not much, was running late)

PROMINEO TECH



```css
:root {
    --color: #be5a08;
}

body {
    margin: 0;
    padding: 0;
    color: var(--color);
    font-family:Arial, Helvetica, sans-serif;
}

* {
    box-sizing: border-box;
}

h1 {
    text-transform: uppercase;
    font-size: 30px;
}

.cell {
    height: 120px;
    width: 120px;
    display: flex;
    justify-content: center;
    align-items: center;
    color: var(--color);
    font-size: 80px;

}

#gameBoard {
    width: 360px;
    display: flex;
```

PROMINEO TECH

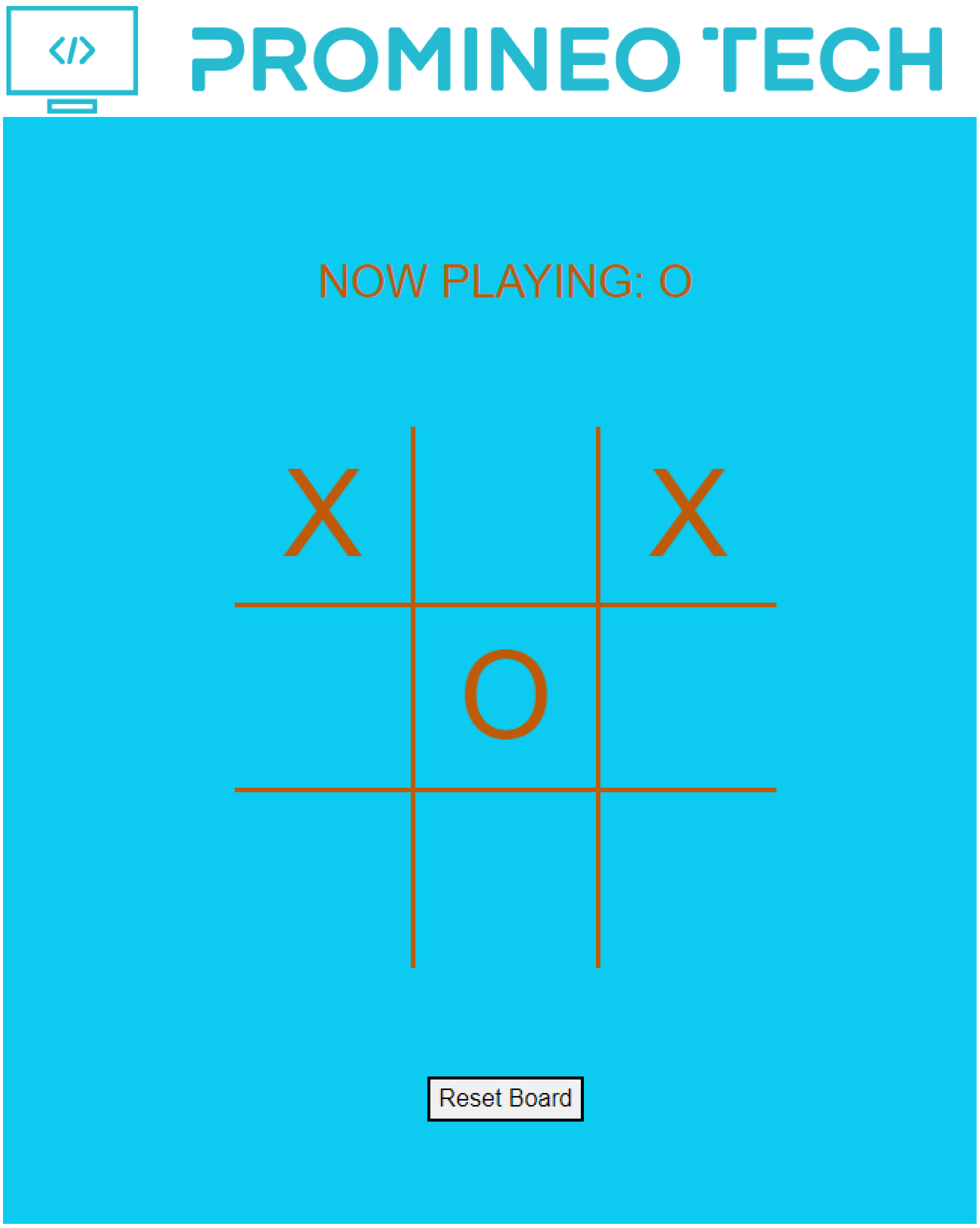


TicTacToe.html U | TicTacToe.js U | TicTacToe.css

TicTacToe.css > .cell

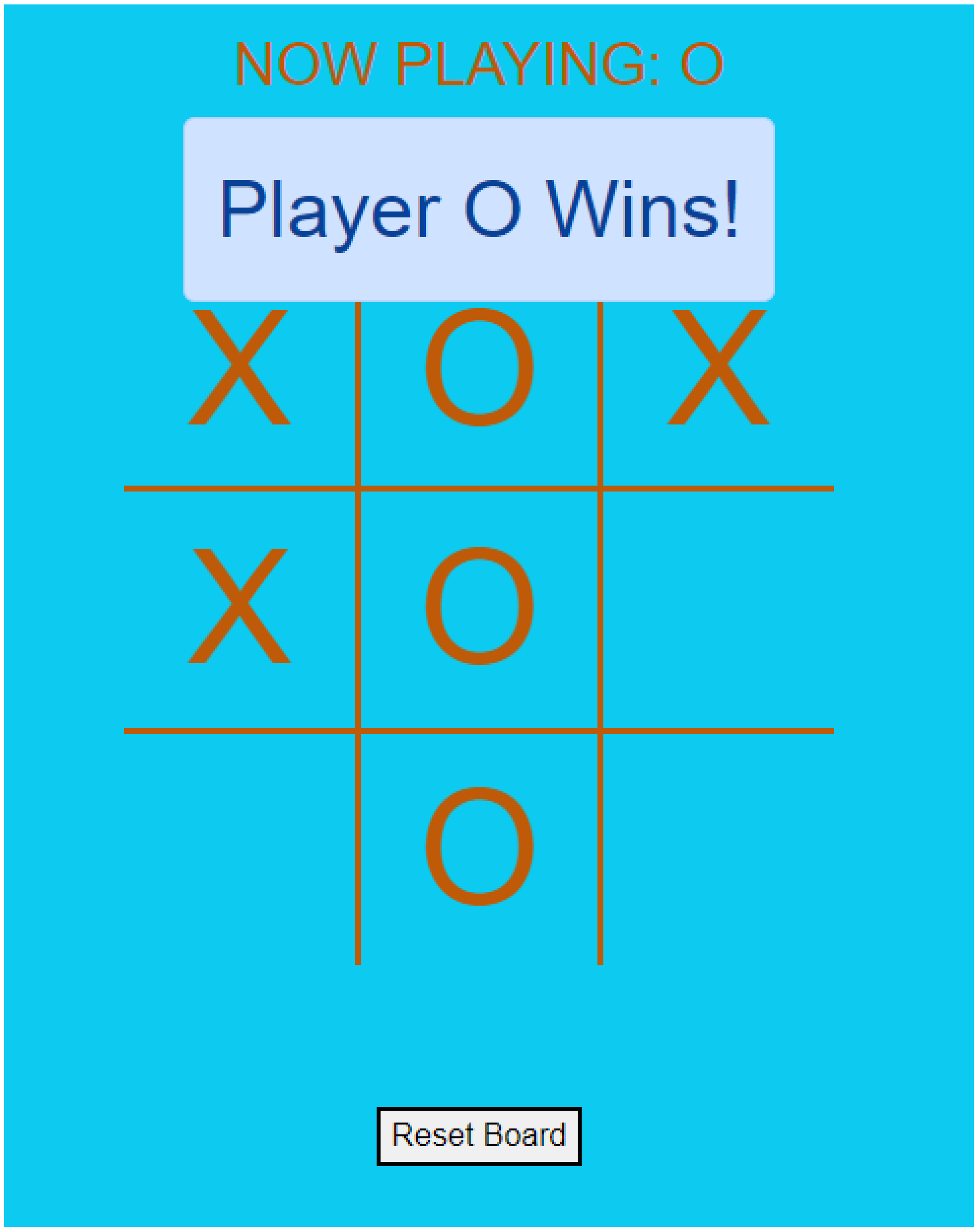
```
    justify-content: center;
    align-items: center;
    color: var(--color);
    font-size: 80px;

}

#gameBoard {
    width: 360px;
    display: flex;
    flex-wrap: wrap;

}

.container {
    justify-content: center;
    align-items: center;
    flex-direction: column;
    display: flex;
    height: 100vh;
    padding: 30px;

}

button:hover {
    cursor: pointer;
}
```

Screenshots of Running Application:

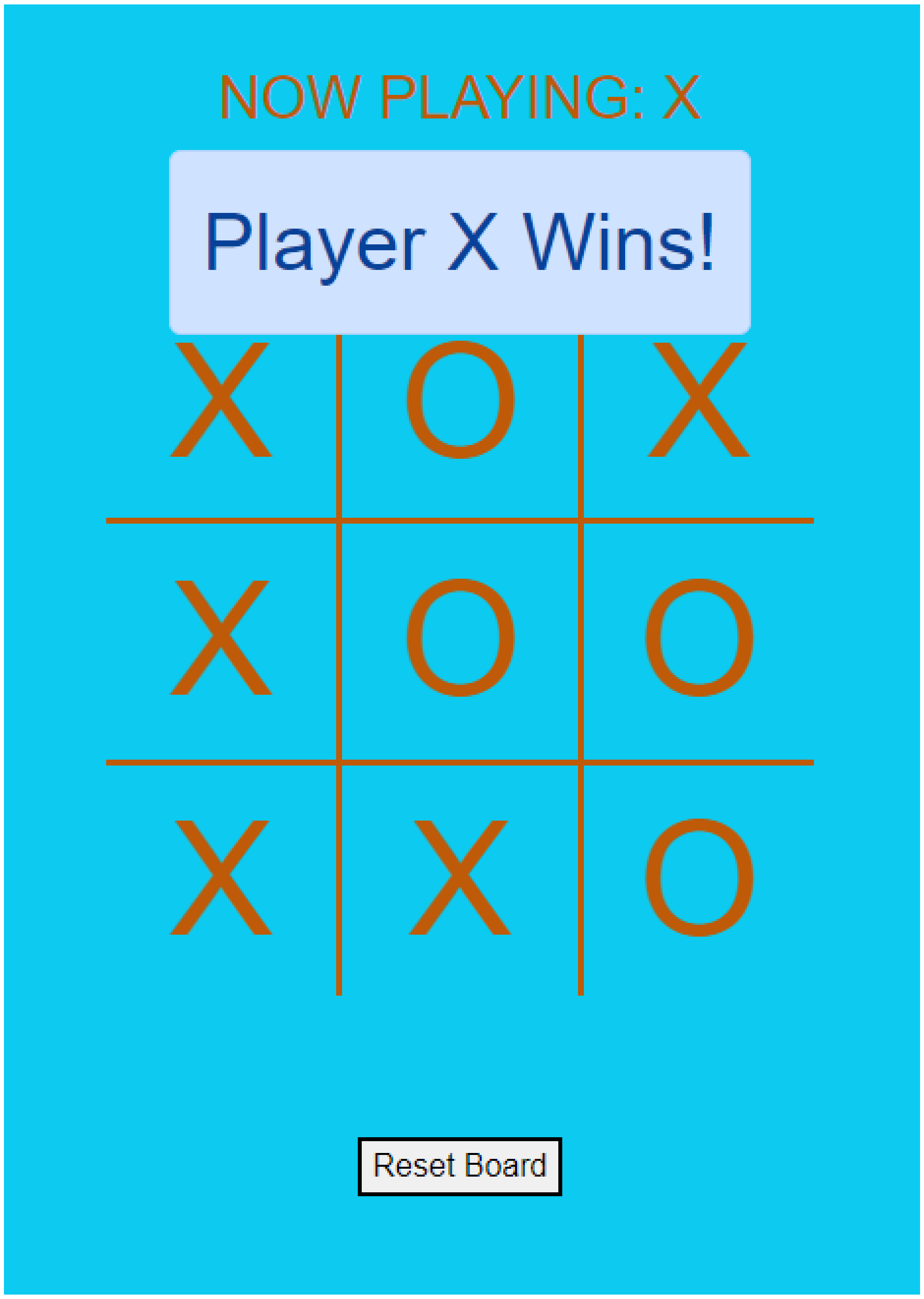


NOW PLAYING: O

PROMINEO TECH



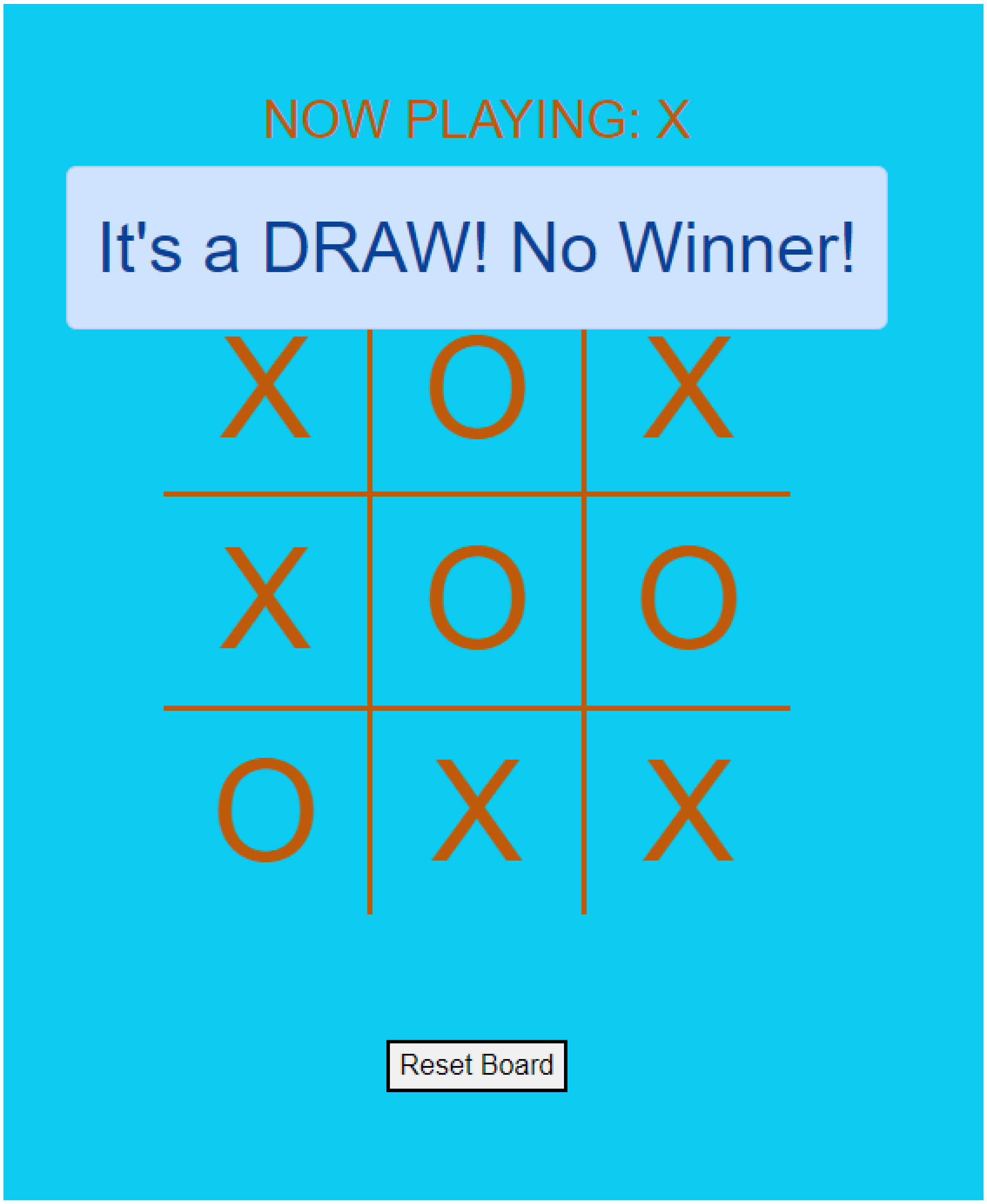
PROMINEO TECH


NOW PLAYING: X

Player X Wins!

X	O	X
X	O	O
X	X	O

PROMINEO TECH


NOW PLAYING: X

It's a DRAW! No Winner!

URL to GitHub Repository:

https://github.com/KalonOhm/TicTacToe