Tic-Tac-Toe Game Practice

HTML

Source code:

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
minimum-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>Tic Tac Toe</title>
   <link rel="stylesheet" href="tic.css">
</head>
<body>
   <section>
      <h1 class="game--title">Tic Tac Toe</h1>
      <div class="game--container">
         <div data-cell-index="0" class="cell"></div>
         <div data-cell-index="1" class="cell"></div>
         <div data-cell-index="2" class="cell"></div>
         <div data-cell-index="3" class="cell"></div>
         <div data-cell-index="4" class="cell"></div>
         <div data-cell-index="5" class="cell"></div>
         <div data-cell-index="6" class="cell"></div>
         <div data-cell-index="7" class="cell"></div>
         <div data-cell-index="8" class="cell"></div>
      </div>
      <h3 class="game--status"></h3>
      <button class="game--restart">Restart Game</button>
   </section>
<script src="tic.js"></script>
</body>
</html
```

CSS

Source code:

```
body {
    font-family: "Poppins", sans-serif;
    background-color: #8c52ff;
}
section {
    text-align: center;
    border-radius: 10px;
    width: 400px;
    margin: auto;
    background-color: #fff;
    margin-top: 75px;
    box-shadow: 0px 10px 36px 16px rgba(0,0,0,0.1);
}
.game--title {
    padding-top: 20px;
}
.game--container {
    display: grid;
    grid-template-columns: repeat(3, auto);
    width: 150px;
    margin: 120px;
    margin-bottom: 10px;
    margin-top: 10px;
}
.cell {
    font-family: "Poppins", sans-serif;
    width: 50px;
    height: 50px;
    box-shadow: 0 0 0 1px #333333;
    border: 1px solid #333333;
    cursor: pointer;
    line-height: 50px;
    font-size: 20px;
}
button {
    margin-bottom: 20px;
    height: 30px;
    border: none;
```

```
    border-radius: 5px;
    color: #fff;
    background-color: #8c52ff;
    cursor: pointer;
}
button:hover {
    transition: 0.25s;
    background-color: #c1a2fe;
}
```

JAVASCRIPT

Source code:

```
const statusDisplay = document.querySelector('.game--status');

let gameActive = true;

let currentPlayer = "X";

let gameState = ["", "", "", "", "", "", "", "", ""];

const winningMessage = () => `Player ${currentPlayer} has won!`;
const drawMessage = () => `Game ended in a draw!`;
const currentPlayerTurn = () => `It's ${currentPlayer}'s turn`;

statusDisplay.innerHTML = currentPlayerTurn();

document.querySelectorAll('.cell').forEach(cell => cell.addEventListener('click',
handleCellClick));
document.querySelector('.game--restart').addEventListener('click', handleRestartGame);

function handleCellClick(clickedCellEvent) {
        const clickedCell = clickedCellEvent.target;
        const clickedCellIndex = parseInt(
          clickedCell.getAttribute('data-cell-index')
        );

        if (gameState[clickedCellIndex] !== "" || !gameActive) {
            return;
        }
```

```
        handleCellPlayed(clickedCell, clickedCellIndex);
        handleResultValidation();
}

function handleCellPlayed(clickedCell, clickedCellIndex) {

        gameState[clickedCellIndex] = currentPlayer;
        clickedCell.innerHTML = currentPlayer;
    }

    const winningConditions = [
        [0, 1, 2],
        [3, 4, 5],
        [6, 7, 8],
        [0, 3, 6],
        [1, 4, 7],
        [2, 5, 8],
        [0, 4, 8],
        [2, 4, 6]
    ];
    function handleResultValidation() {
        let roundWon = false;
        for (let i = 0; i <= 7; i++) {
            const winCondition = winningConditions[i];
            let a = gameState[winCondition[0]];
            let b = gameState[winCondition[1]];
            let c = gameState[winCondition[2]];
            if (a === '' || b === '' || c === '') {
                continue;
            }
            if (a === b && b === c) {
                roundWon = true;
                break
            }
        }
    if (roundWon) {
        statusDisplay.innerHTML = winningMessage();
        gameActive = false;
        return;
    }

    let roundDraw = !gameState.includes("");
    if (roundDraw) {
```

```
        statusDisplay.innerHTML = drawMessage();
        gameActive = false;
        return;
    }

    handlePlayerChange();
}

function handlePlayerChange() {
    currentPlayer = currentPlayer === "X" ? "O" : "X";
    statusDisplay.innerHTML = currentPlayerTurn();
}

function handleRestartGame() {
    gameActive = true;
    currentPlayer = "X";
    gameState = ["", "", "", "", "", "", "", "", ""];
    statusDisplay.innerHTML = currentPlayerTurn();
    document.querySelectorAll('.cell')
            .forEach(cell => cell.innerHTML = "");
}
```