CSS:

Body {

Font-family: “Arial”, sans-serif;

}

Section {

Text-align: center;

}

.game—title {

Font-size: 100px;

Color: #d7a62f;

Margin: 10px auto;

}

.game—container {

Display: grid;

Grid-template-columns: repeat(3, auto);

Width: 306px;

Margin: 10px auto;

Background-color: #11213a;

Color: #04c0b2;

}

.cell {

Font-family: “Permanent Marker”, cursive;

Width: 100px;

Height: 100px;

Box-shadow: 2px 2px 2px 2px #ecd7ba;

Border: 2px solid #ecd7ba;

Cursor: pointer;

Line-height: 100px;

Font-size: 60px;

}

.game—status {

Font-size: 50px;

Color: #d7a62f;

Margin: 20px auto;

}

.game—restart {

Background-color: #f7e4ac;

Width: 200px;

Height: 50px;

Font-size: 25px;

Color: #5586e2;

Box-shadow: 2px 2px 2px 2px #d86c23;

Border: 2px solid #d86c23;

}

Js:

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();

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 handleCellPlayed(clickedCell, clickedCellIndex) {

gameState[clickedCellIndex] = currentPlayer;

clickedCell.innerHTML = currentPlayer;

}

Function handlePlayerChange() {

currentPlayer = currentPlayer === “X” ? “O” : “X”;

statusDisplay.innerHTML = currentPlayerTurn();

}

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 handleCellClick(clickedCellEvent) {

Const clickedCell = clickedCellEvent.target;

Const clickedCellIndex = parseInt(clickedCell.getAttribute(‘data-cell-index’));

If (gameState[clickedCellIndex] !== “” || !gameActive) {

Return;

}

handleCellPlayed(clickedCell, clickedCellIndex);

handleResultValidation();

}

Function handleRestartGame() {

gameActive = true;

currentPlayer = “X”;

gameState = [“”, “”, “”, “”, “”, “”, “”, “”, “”];

statusDisplay.innerHTML = currentPlayerTurn();

document.querySelectorAll(‘.cell’).forEach(cell => cell.innerHTML = “”);

}

Document.querySelectorAll(‘.cell’).forEach(cell => cell.addEventListener(‘click’, handleCellClick));

Document.querySelector(‘.game—restart’).addEventListener(‘click’, handleRestartGame);