```
window.addEventListener('DOMContentLoaded', () => {
    const tiles = Array.from(document.querySelectorAll('.tile'));
    const playerDisplay = document.querySelector('.display-player');
    const resetButton = document.querySelector('#reset');
    const announcer = document.querySelector('.announcer');

    let board = ['', '', '', '', '', '', '', '', ''];
    let currentPlayer = 'X';
    let isGameActive = true;

    const PLAYERX_WON = 'PLAYERX_WON';
    const PLAYERO_WON = 'PLAYERO_WON';
    const TIE = 'TIE';


    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];
            const a = board[winCondition[0]];
            const b = board[winCondition[1]];
            const c = board[winCondition[2]];
            if (a === '' || b === '' || c === '') {
                continue;
            }
            if (a === b && b === c) {
                roundWon = true;
                break;
            }
        }

        if (roundWon) {
            announce(currentPlayer === 'X' ? PLAYERX_WON : PLAYERO_WON);
            isGameActive = false;
            return;
        }
    if (!board.includes(''))
        announce(TIE);
}

const announce = (type) => {
    switch (type) {
        case PLAYERO_WON:
            announcer.innerHTML = 'Player <span class="playerO">O</span> Won';
            break;
        case PLAYERX_WON:
            announcer.innerHTML = 'Player <span class="playerX">X</span> Won';
            break;
        case TIE:
            announcer.innerText = 'Game-Tie';
    }
    announcer.classList.remove('hide');
};

const isValidAction = (tile) => {
    if (tile.innerText === 'X' || tile.innerText === 'O') {
        return false;
    }

    return true;
};

const updateBoard = (index) => {
    board[index] = currentPlayer;
}

const changePlayer = () => {
    playerDisplay.classList.remove(`player${currentPlayer}`);
    currentPlayer = currentPlayer === 'X' ? 'O' : 'X';
    playerDisplay.innerText = currentPlayer;
    playerDisplay.classList.add(`player${currentPlayer}`);
}

const userAction = (tile, index) => {
    if (isValidAction(tile) && isGameActive) {
        tile.innerText = currentPlayer;
        tile.classList.add(`player${currentPlayer}`);
        updateBoard(index);
        handleResultValidation();
        changePlayer();
    }
}

const resetBoard = () => {
        board = ['', '', '', '', '', '', '', '', ''];
        isGameActive = true;
        announcer.classList.add('hide');

        if (currentPlayer === 'O') {
            changePlayer();
        }

        tiles.forEach(tile => {
            tile.innerText = '';
            tile.classList.remove('playerX');
            tile.classList.remove('playerO');
        });
    }

    tiles.forEach((tile, index) => {
        tile.addEventListener('click', () => userAction(tile, index));
    });

    resetButton.addEventListener('click', resetBoard);
})
```