Minsweeper

Contains HTML and Javascript file

```
<!-- HTML part of the code -->
<!DOCTYPE html>
<html lang="en">
    <head>
        <!-- Required meta tags -->
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1" />

        <!-- This CSS code is made freely available the Bootstrap team. The file link directly links to
        <link
            href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
            rel="stylesheet"
            integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
            crossorigin="anonymous"
        />
        <link
            rel="stylesheet"
            href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.7.2/font/bootstrap-icons.css"
        />

        <title>Minesweeper</title>
    </head>
    <body class="bg-dark text-white">
        <div class="container">
            <div class="row my-3">
                <div class="col">
                    <table
                        class="mx-auto"
                        id="buttonGrid"
                        style="border-collapse: collapse; line-height: 0"
                    ></table>
                </div>
            </div>
            <div class="row">
                <div class="col text-center">
                    <ul class="list-group list-group-flush">
                        <li class="list-group-item bg-dark text-white">
                            <strong>Controls</strong>
                        </li>
                        <li class="list-group-item bg-dark text-white">
                            Left click to reveal a box.
                        </li>
                        <li class="list-group-item bg-dark text-white">
                            Right click to flag.
                        </li>
                        <li class="list-group-item bg-dark text-white">
                            Double click to reveal all surronding squares.
                        </li>
                        <li class="list-group-item bg-dark text-white">
                            <input
                                type="checkbox"
                                class="btn-check"
```

```
                                    id="speedMode"
                                    autocomplete="off"
                                />
                                <label class="btn btn-outline-info" for="speedMode"
                                    >Speed Mode</label
                                >
                                Left click to reveal all surronding square (Off by default)
                            </li>
                        </ul>
                    </div>
                </div>
            </div>

            <script src="main.js"></script>

            <!-- The following Javascript code is made freely available the Bootstrap team. The file link di
            <script
                src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
                integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
                crossorigin="anonymous"
            ></script>

        </body>
    </html>

-------------------------------------------------------------------
// Javascript section of Code
// Settings
const ROWS = 8;
const COLS = 12;
const MINECOUNT = 10;
const TILESIZE = 40;
// The images are made freely available by UchiMama at uchimama.itch.io
// They can be found and downloaded for free at https://uchimama.itch.io/minesweeper-tileset
const IMAGES = {
    0: 'Images/empty.png',
    1: 'Images/1.png',
    2: 'Images/2.png',
    3: 'Images/3.png',
    4: 'Images/4.png',
    5: 'Images/5.png',
    6: 'Images/6.png',
    7: 'Images/7.png',
    8: 'Images/8.png',
    unknown: 'Images/unknown.png',
    flag: 'Images/flag.png',
    mine: 'Images/mine.png',
};
const NUMBER_IMAGES = [
    '0.png',
    '1.png',
    '2.png',
    '3.png',
    '4.png',
    '5.png',
    '6.png',
    '7.png',
    '8.png',
];

// Makes loops simpler
const RELATIVE_NEIGHBORS = [
    [-1, -1],
    [-1, 0],
    [-1, 1],
    [0, -1],
    [0, 1],
    [1, -1],
    [1, 0],
    [1, 1],
];

var flagged = 0;
var mineList = [];

const genMines = () => {
    // Generate empty board
    let board = [];
    for (let i = 0; i < ROWS; i++) {
        board.push([]);
        for (let j = 0; j < COLS; j++) {
            board[i].push(0);
        }
    }

    // Generate mines
    while (mineList.length < MINECOUNT) {
        let row = Math.floor(Math.random() * ROWS);
        let col = Math.floor(Math.random() * COLS);

        if (JSON.stringify(mineList).indexOf(JSON.stringify([row, col])) == -1) {
            mineList.push([row, col]);
            board[row][col] = 'mine';
        }
    }
    mineList.sort();

    // Define how many mines around a square
    for (let row = 0; row < ROWS; row++) {
        for (let col = 0; col < COLS; col++) {
            if (board[row][col] === 0) {
                let mineCount = 0;
                for (let i = -1; i < 2; i++) {
                    for (let j = -1; j < 2; j++) {
                        if (
                            (-1 < row + i) &
                            (-1 < col + j) &
                            (row + i < ROWS) &
                            (col + j < COLS)
                        ) {
```

```
                        if (board[row + i][col + j] === 'mine') {
                            mineCount++;
                        }
                    }
                }
            }
            board[row][col] = mineCount;
        }
    }
}

return board;
};
const board = genMines();
console.log(board);

const checkGameOver = (btn) => {
    // Check for loss
    let boardState = Array.from(document.getElementsByClassName('button'));
    if (btn.src.includes(IMAGES['mine'])) {
        setTimeout(() => {
            if (confirm('YOU LOSE! Would you like to play again?')) {
                location.reload();
            }
        }, 100);
        for (button of boardState) {
            let pos = JSON.parse(button.id);
            if (board[pos[0]][pos[1]] === 'mine') {
                button.src = IMAGES['mine'];
            }
        }
    }

    // Check for win
    boardState = Array.from(document.getElementsByClassName('button'));
    let hidden = boardState.filter(
        (button) =>
            button.src.includes(IMAGES['unknown']) ||
            button.src.includes(IMAGES['flag'])
    );
    for (let button = 0; button < hidden.length; button++) {
        hidden[button] = JSON.parse(hidden[button].id);
    }
    hidden.sort();
    console.log(hidden);
    console.log(mineList);
    if (JSON.stringify(hidden) === JSON.stringify(mineList)) {
        setTimeout(() => {
            if (confirm('YOU WIN! Would you like to play again?')) {
                location.reload();
            }
        }, 10);
    }
};
```

```javascript
const clear = (pos) => {
    let row = pos[0];
    let col = pos[1];

    for (neighbor of RELATIVE_NEIGHBORS) {
        let i = neighbor[0];
        let j = neighbor[1];

        if (-1 < row + i && row + i < ROWS && -1 < col + j && col + j < COLS) {
            let pos = [row + i, col + j];
            let btn = document.getElementById(JSON.stringify(pos));

            if (board[pos[0]][pos[1]] === 0 && btn.src.includes(IMAGES['unknown'])) {
                btn.src = IMAGES[board[pos[0]][pos[1]]];
                clear(pos);
            }

            btn.src = IMAGES[board[pos[0]][pos[1]]];
        }
    }
    return;
};

const clearNeighbors = (btn) => {
    let pos = JSON.parse(btn.id);

    if (NUMBER_IMAGES.includes(btn.src.split('/').slice(-1)[0])) {
        let row = pos[0];
        let col = pos[1];

        for (neighbor of RELATIVE_NEIGHBORS) {
            let i = neighbor[0];
            let j = neighbor[1];

            if (-1 < row + i && row + i < ROWS && -1 < col + j && col + j < COLS) {
                let pos = [row + i, col + j];
                let btn = document.getElementById(JSON.stringify(pos));
                let boardState = Array.from(document.getElementsByClassName('button'));

                if (
                    board[pos[0]][pos[1]] === 'mine' &&
                    !btn.src.includes(IMAGES['flag'])
                ) {
                    for (button of boardState) {
                        let pos = JSON.parse(button.id);
                        if (board[pos[0]][pos[1]] === 'mine') {
                            button.src = IMAGES['mine'];
                        }
                    }
                } else if (!btn.src.includes(IMAGES['flag'])) {
                    btn.src = IMAGES[board[pos[0]][pos[1]]];
                    if (btn.src.includes(IMAGES[0])) {
                        clear(pos);
                    }
                }
                checkGameOver(btn);
            }
        }
    }
};

// Create visuals and event detectors
const buttonGrid = document.getElementById('buttonGrid');
for (let row = 0; row < ROWS; row++) {
    let tr = document.createElement('tr');

    for (let col = 0; col < COLS; col++) {
        let button = document.createElement('img');
        button.src = IMAGES['unknown'];
        button.height = TILESIZE;
        button.width = TILESIZE;
        button.id = JSON.stringify([row, col]);
        button.classList.add('button');

        // Basic click
        button.addEventListener('click', (e) => {
            let btn = e.target;
            let pos = JSON.parse(btn.id);

            if (btn.src.includes(IMAGES['unknown'])) {
                btn.src = IMAGES[board[pos[0]][pos[1]]];
            } else if (document.getElementById('speedMode').checked) {
                clearNeighbors(btn);
            }

            if (btn.src.includes(IMAGES[0])) {
                clear(pos);
            }

            checkGameOver(btn);
        });

        // Flag button
        button.addEventListener(
            'contextmenu',
            (e) => {
                e.preventDefault();
                let btn = e.target;

                if (btn.src.includes(IMAGES['unknown']) && flagged <= MINECOUNT) {
                    btn.src = IMAGES['flag'];
                    flagged++;
                } else if (btn.src.includes(IMAGES['flag'])) {
                    btn.src = IMAGES['unknown'];
                    flagged--;
                }
            },
            false
        );
        // Clear by double click
        button.addEventListener(
            'dblclick',
            (e) => {
                if (document.getElementById('speedMode').checked) {
                    console.log('do not clear neighbors')
                    return;
                }
                console.log('clear neighbors')
                let btn = e.target;
                clearNeighbors(btn);
            },
            false
        );

        let td = document.createElement('td');
        td.classList.add('p-0');
        td.appendChild(button);

        tr.appendChild(td);
    }
    buttonGrid.appendChild(tr);
}
```