🎮 Project: Build Your Own Minesweeper Game (HTML, CSS, JavaScript)

# 🧠 What Is Minesweeper?

Minesweeper is a classic logic-based game where the player uncovers cells on a grid without triggering hidden mines. The objective is to reveal all safe cells while marking all the mines with flags.

- The game board is a grid of hidden cells.

- Some of these cells contain mines (💣).

- Clicking a safe cell reveals a number indicating how many neighboring cells contain mines (0–8).

- If you reveal a cell with no neighboring mines, it will auto-reveal adjacent cells.

- Right-clicking on a cell lets you flag it (🚩) if you think it has a mine.

- You lose if you click on a mine.

- You win if you reveal all non-mine cells and reveal all other cells.

# 🛠️ Your Task

You will build a Minesweeper using HTML, CSS, and JavaScript. You’ll create a 10x10 grid with 10 randomly placed mines and full game logic.

# 🗂️ Project Setup

1. Create these files:

- index.html  
 - style.css  
 - script.js

2. Basic HTML Layout:

<div id="status">Mines Left: <span id="mines-left">10</span></div>  
<div id="game-board"></div>  
<button id="restart">Restart</button>

3. Use CSS Grid or flex for the game board layout. Each cell should be a square <div> with:

<div class="cell" data-row="0" data-col="0"></div>

# 🔧 Game Variables (Global State)

let board = [];  
let boardSize = 10;  
let totalMines = 10;  
let minesLeft = totalMines;  
let revealedCells = 0;  
let gameStarted = false;  
let gameOver = false;  
let timerInterval;  
let startTime;

This is how cell should look like (board[i][j]):

{

isMine: false,

revealed: false,

flagged: false,

neighborMines: 0,

}

# 🧩 Key Functions to Implement

|  |  |
| --- | --- |
| Function Name | Purpose |
| initGame() | Sets up or resets the game. Builds the board and UI. |
| placeMines(initialRow, initialCol) | Randomly places mines on the board after the first click. |
| countNeighborMines(row, col) | Counts how many mines are around the given cell. |
| handleCellClick(row, col) | Main click handler: reveals cell, starts game, ends game if needed. |
| revealCell(row, col) | Reveals a cell. recursively reveals nearby cells if it’s a 0 (implement later, first build working functionality). |
| toggleFlag(row, col) | Handles right-click to add/remove a flag on a cell. (see hints and tips) |
| checkWin() | Checks if all safe cells are revealed to determine if the player won. |
| endGame(won) | Stops the game and reveals all mines or shows a win message. |
| startTimer() | Starts counting time after the first click. |

# 💡 Tips and Hints

## 🔢 Use data-row and data-col on Each Cell

This will help you locate and identify cells when handling clicks:

<div class="cell" data-row="3" data-col="5"></div>

## 🎨 Use CSS Pseudo-Elements for Icons

.cell.flagged::after {  
 content: "🚩";  
 font-size: 16px;  
}  
  
.cell.mine.revealed::after {  
 content: "💣";  
 font-size: 16px;  
}

## ⚠️ Block Default Context Menu

cell.addEventListener("contextmenu", (event) => {  
 event.preventDefault();  
 toggleFlag(row, col);  
});

# ✅ Minimum Features to Complete

[ ] 10x10 board

[ ] 10 mines placed randomly

[ ] Reveal cells and show mine counts

[ ] Auto-reveal nearby 0s

[ ] Right-click to flag cells

[ ] Timer and mine counter

[ ] Restart button

[ ] Win and lose detection

# 🌟 Bonus Features (Optional)

- Difficulty levels (easy/medium/hard)

- Animation when revealing cells

- Responsive layout for mobile

- Sound effects for clicks or game end

- High score tracking (via localStorage)