

LLM Interaction Log — Lab 1 Part 1

Student Name: Hassan Sohail

UWO email address: hsohail7@uwo.ca

GitHub Classroom Repository URL: <https://github.com/AlexandraLHeureuxECE/lab-1-hsohail7.git>

LLM Tool Disclosure

List the LLM tool(s) used for this lab.

Tool name(s): Chatgpt 5.2

How each tool was used (briefly): Prompting by text

Full LLM Interaction Record

Below is the complete record of all meaningful LLM interactions used during the development of this lab.

For EACH interaction, include:

- the prompt (verbatim)
- a brief summary of the LLM's response (1–2 sentences)

Do NOT include full LLM output in this section. Add more rows as needed.

	Prompt (verbatim)	LLM Response Summary (1–2 sentences)
1	hi, I have to make a tik tac toe game "A simple, web-based Tic-Tac-Toe game for two people playing on the same device. The game should behave in a way that would feel familiar to someone who has played Tic-Tac-Toe before. Specific design and implementation decisions are intentionally left to you." "Game Expectations (High-Level) Your application should allow players to: interact with a visible 3x3 grid, take turns, reach a clear game outcome, and start a new game without refreshing the page. give the option to restart the game at any time. The interface should communicate what is happening as the game progresses." use "HTML + CSS + JavaScript" to make the game "Required Extra Feature (Choose ONE) In addition to basic gameplay, implement one of the following: Keyboard support The game can	The ChatGPT 5.2 model provided the three-file structure I needed, explained which folders to place them in, and then supplied the code for each respective file. It also suggested appearance customizations and included brief instructions for running the project locally and deploying it with GitHub Pages.

	<p>be played without using a mouse. OR</p> <p>Appearance customization A simple option that changes how the board or marks appear (e.g., theme toggle, mark style). OR</p> <p>Enhanced game feedback "use the appererance constimization feature so please go ahead and do it and let me see thef iles and interconnect them so i can run them loccally and run on github pages as per the requiemtns</p>	
2	<p>ok the work works, but I feel like we can communicate better as said in the requiemtns "Game Expectations (High-Level) Your application should allow players to: interact with a visible 3x3 grid, take turns, reach a clear game outcome, and start a new game without refreshing the page. give the option to restart the game at any time. The interface should communicate what is happening as the game progresses" it does that but its not apparnet for the user when we plays so i want o make sure its that also to add on, "When theme is light, button text becomes "Light"; when dark, "Dark"" these too so like make sure the UI doest break and then make it again also outline a rules for game if you can to make it smoothe play and makirn sure our UI/UX doesnt break and also communutes because my main emphasis is for "Game Expectations (High-Level) Your application should allow players to: interact with a visible 3x3 grid, take turns, reach a clear game outcome, and start a new game without refreshing the page. give the option to restart the game at any time. The interface should communicate what is happening as the game progresses" these are important so you cn break it down slwoly so you make sure u achvie it in correct manner</p>	<p>The model reviewed the existing Tic-Tac-Toe UI and pointed out a clear set of "communicate what's happening" UX rules, like turn guidance, end-of-game messages, restart behavior, and theme label meaning. It then provided updated index.html, style.css, and script.js files that make the status and hint messages clearer, explain the theme button as showing the current theme, and improve visual feedback (including more noticeable win highlights), all without breaking GitHub Pages deployment.</p>

3	<p>its better but the main thing i dont liek rn the grid boxes changes its shapes i feel like if we can keep it 1 size and then add the emoji/x or o or even the a/b but making sure the style is constant thruout?</p>	<p>The model noticed that inconsistent glyph sizes (especially with emojis) were making the Tic-Tac-Toe cells feel uneven. To fix this, it proposed locking each cell into a square shape and rendering the marks inside a fixed-size inner container. It then provided clear CSS and JavaScript changes that keep sizing consistent, center the marks properly, and make X/O, emoji, and letter styles behave uniformly without changing the core gameplay.</p>
4	<p>ok can u give me enntier codes dont give me snipets and also mkae sure the emojis we add are big so we can see them thanks boss</p>	<p>The model provided full replacement files (index.html, style.css, and script.js) that keep the Tic-Tac-Toe gameplay intact while fixing uneven cell sizing by locking each grid cell into a square and rendering marks inside a fixed-size .mark container. It also preserved clear status and hint messaging, kept the Light/Dark button labeled as the “current theme,” and ensured emoji marks stay large, readable, and stable for GitHub Pages deployment.</p>
5	<p>no again the grids shift alot i want a static size for the grid pls can u make it 1 size regardlss if its empty or that? so thnk what casuses that probem and fix it thank u</p>	<p>The model realized that the “grid shifting” was mostly a visual issue caused by hover and active transform animations. To stabilize the layout, it suggested making the board a fixed square with explicit width and height and fixed grid rows and columns. It then provided full replacement index.html, style.css, and script.js files that remove transform-based motion, keep cells and marks consistently sized, render large emoji inside a fixed .mark container, and preserve the game’s status and hint messaging while remaining compatible with GitHub Pages.</p>
6	<p>its not working it still tranforms it moves the box does gets bigger when a user clicks, i want to make sure its a hard grid size and thats it, plesse first check whats the underlining issue, and then figure out if my option is pasuable if its not then tell me if it is then give mt he corrected code</p>	<p>The model explained that the “cell gets bigger when clicked” effect usually comes from things like focus rings, box shadows (including the win highlight), and default browser button styling not from actual layout changes. It then provided fully updated files that lock the board size in place, reset the button appearance, switch outer glows to inset highlights, keep emoji large inside a fixed mark container, and even included a quick DevTools tip to confirm that the cell width and height never change.</p>

7	<p>it didnt work at all, and i am using chrome btw, so what is the issue make a limited size of text u made it work once earlier but idk waht u are doing that it aintw working</p>	<p>The model traced the remaining “tile grows on click” effect in Chrome to built-in browser button focus and active styles, which can still change how a button looks even when its size is fixed. To fix it for good, it suggested fully resetting the cell button styles using all: unset and then reapplying only the styles that matter. It also limited mark size with a CSS variable and provided fully updated index.html, style.css, and script.js files to make sure the grid looks completely static in Chrome.</p>
8	<p>ok finally its better but one issue right now is thatht the mark styles the emoji arent centered is there a way to fix that? thank check and then lmk if not then its ok</p>	<p>The model explained that emoji glyphs can look slightly off-center because of font baseline differences, even when the container itself is centered. It fixed this neatly by adding a data-mark-style="emoji" attribute to the board in JavaScript and applying a small translateY(-2px) tweak in CSS only when emoji mode is active without affecting the locked grid or any of the other mark styles.</p>
9		
10		

Selected Interaction Analysis (3 Required)

From the full interaction record above, select THREE (3) interactions that were most meaningful.

For each selected interaction, include the FULL LLM output (verbatim or lightly trimmed) and complete the analysis below.

You must include:

- one interaction that led to the initial version
- one interaction that resulted in a visible UI change

Selected Interaction 1

Prompt (verbatim):

<verbatim prompt text>

hi, I have to make a tik tac toe game "A simple, web-based Tic-Tac-Toe game for two people playing on the same device. The game should behave in a way that would feel familiar to someone who has played Tic-Tac-Toe before. Specific design and implementation decisions are intentionally left to you." "Game Expectations (High-Level) Your application should allow players to: interact with a visible 3×3 grid, take turns, reach a clear game outcome, and start a new game without refreshing the page. give the option to restart the game at any time. The interface should communicate what is happening as the game progresses." use "HTML + CSS + JavaScript" to make the game "Required Extra Feature (Choose ONE) In addition to basic gameplay, implement one of the following: Keyboard support The game can be played without using a mouse. OR Appearance customization A simple option that changes how the board or marks appear (e.g., theme toggle, mark style). OR Enhanced game feedback" use the appearance customization feature so please go ahead and do it and let me see the files and interconnect them so i can run them locally and run on github pages as per the requirements

LLM Output (verbatim or lightly trimmed):

<LLM response excerpt>

File structure

Create a folder like tic-tac-toe/:

tic-tac-toe/

index.html

style.css

script.js

index.html

```
<!doctype html>

<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Tic-Tac-Toe</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <main class="app" id="app">
      <header class="header">
        <div>
          <h1>Tic-Tac-Toe</h1>
          <p class="sub">Two players • same device</p>
        </div>
        <div class="controls">
          <div class="control">
            <label class="label" for="themeToggle">Theme</label>
            <button id="themeToggle" class="btn btn-secondary" type="button" aria-pressed="false">
              Light
            </button>
          </div>
          <div class="control">
            <label class="label" for="markStyle">Mark style</label>
            <select id="markStyle" class="select" aria-label="Choose mark style">
              <option value="classic" selected>Classic (X / O)</option>
            </select>
          </div>
        </div>
      </header>
      <div class="grid">
        <div class="row">
          <div class="cell"></div>
          <div class="cell"></div>
          <div class="cell"></div>
        </div>
        <div class="row">
          <div class="cell"></div>
          <div class="cell"></div>
          <div class="cell"></div>
        </div>
        <div class="row">
          <div class="cell"></div>
          <div class="cell"></div>
          <div class="cell"></div>
        </div>
      </div>
    </main>
  </body>

```

```
<option value="emoji">Emoji (X / O)</option>
<option value="letters">Letters (A / B)</option>
</select>
</div>

<div class="control">
  <label class="label" for="restartBtn">Game</label>
  <button id="restartBtn" class="btn" type="button">
    Restart
  </button>
</div>
</div>
</header>

<section class="panel">
  <div id="status" class="status" role="status" aria-live="polite">
    Player X's turn
  </div>
  <div class="scoreboard" aria-label="Scoreboard">
    <div class="pill"><span id="p1Label">Player X</span>: <strong id="scoreX">0</strong></div>
    <div class="pill">Draws: <strong id="scoreD">0</strong></div>
    <div class="pill"><span id="p2Label">Player O</span>: <strong id="scoreO">0</strong></div>
  </div>
</section>
<section class="board-wrap">
  <div
    id="board">
```

```
class="board"
role="grid"
aria-label="Tic-Tac-Toe board"
>
<!-- 9 buttons injected by JS -->
</div>
</section>
<footer class="footer">
<span>Tip: Use the appearance options to change the look.</span>
</footer>
</main>
<script src="script.js"></script>
</body>
</html>
style.css
:root {
--bg: #0b1020;
--panel: rgba(255, 255, 255, 0.06);
--panel-2: rgba(255, 255, 255, 0.10);
--text: #e9ecf1;
--muted: rgba(233, 236, 241, 0.70);
--border: rgba(255, 255, 255, 0.14);
--shadow: 0 18px 50px rgba(0, 0, 0, 0.35);
--accent: #7c5cff;
--accent-2: #2ee59d;
--cell: rgba(255, 255, 255, 0.07);
```

```
--cell-hover: rgba(255, 255, 255, 0.12);
--focus: rgba(124, 92, 255, 0.55);
--win: rgba(46, 229, 157, 0.20);

}

[data-theme="light"] {

--bg: #f6f7fb;
--panel: rgba(0, 0, 0, 0.04);
--panel-2: rgba(0, 0, 0, 0.07);
--text: #141824;
--muted: rgba(20, 24, 36, 0.65);
--border: rgba(0, 0, 0, 0.10);
--shadow: 0 18px 50px rgba(18, 22, 33, 0.12);
--accent: #5b49ff;
--accent-2: #0aa66b;
--cell: rgba(0, 0, 0, 0.04);
--cell-hover: rgba(0, 0, 0, 0.07);
--focus: rgba(91, 73, 255, 0.30);
--win: rgba(10, 166, 107, 0.16);

}

* { box-sizing: border-box; }

body {

margin: 0;
min-height: 100vh;
display: grid;
place-items: center;
font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, Helvetica, Arial;
```

```
background: radial-gradient(1200px 700px at 30% 15%, rgba(124, 92, 255, 0.25),  
transparent 55%),  
radial-gradient(1000px 600px at 75% 80%, rgba(46, 229, 157, 0.16), transparent 55%),  
var(--bg);  
color: var(--text);  
padding: 20px;  
}  
.app {  
width: min(860px, 100%);  
background: var(--panel);  
border: 1px solid var(--border);  
border-radius: 18px;  
box-shadow: var(--shadow);  
overflow: hidden;  
}  
.header {  
display: flex;  
align-items: flex-start;  
justify-content: space-between;  
gap: 16px;  
padding: 18px 18px 12px;  
border-bottom: 1px solid var(--border);  
}  
h1 {  
margin: 0;  
font-size: 22px;
```

```
letter-spacing: 0.2px;  
}  
  
.sub {  
margin: 6px 0 0;  
color: var(--muted);  
font-size: 13px;  
}  
  
.controls {  
display: flex;  
flex-wrap: wrap;  
gap: 10px;  
justify-content: flex-end;  
}  
  
.control {  
display: grid;  
gap: 6px;  
}  
  
.label {  
font-size: 12px;  
color: var(--muted);  
}  
  
.btn, .select {  
height: 36px;  
padding: 0 12px;  
border-radius: 12px;  
border: 1px solid var(--border);
```

```
background: var(--panel-2);
color: var(--text);
outline: none;
}

.btn {
cursor: pointer;
font-weight: 600;
}

.btn:focus-visible, .select:focus-visible {
box-shadow: 0 0 0 4px var(--focus);
}

.btn-secondary {
position: relative;
}

.panel {
padding: 14px 18px;
display: grid;
gap: 10px;
border-bottom: 1px solid var(--border);
}

.status {
font-size: 14px;
color: var(--text);
padding: 10px 12px;
border-radius: 14px;
background: var(--panel-2);
```

```
border: 1px solid var(--border);  
}  
  
.scoreboard {  
display: flex;  
gap: 10px;  
flex-wrap: wrap;  
}  
  
.pill {  
padding: 8px 10px;  
border-radius: 999px;  
background: var(--panel-2);  
border: 1px solid var(--border);  
font-size: 13px;  
color: var(--muted);  
}  
  
.pill strong { color: var(--text); }  
  
.board-wrap {  
padding: 18px;  
display: grid;  
place-items: center;  
}  
  
.board {  
width: min(420px, 92vw);  
aspect-ratio: 1 / 1;  
display: grid;  
grid-template-columns: repeat(3, 1fr);
```

```
gap: 10px;  
}  
  
.cell {  
border: 1px solid var(--border);  
background: var(--cell);  
border-radius: 18px;  
cursor: pointer;  
display: grid;  
place-items: center;  
font-size: clamp(36px, 8vw, 64px);  
font-weight: 800;  
user-select: none;  
transition: transform 120ms ease, background 120ms ease;  
}  
  
.cell:hover { background: var(--cell-hover); transform: translateY(-1px); }  
.cell:active { transform: translateY(0px) scale(0.99); }  
  
.cell:focus-visible {  
outline: none;  
box-shadow: 0 0 0 4px var(--focus);  
}  
  
.cell[disabled] {  
cursor: not-allowed;  
opacity: 0.95;  
}  
  
.cell.win {  
background: var(--win);
```

```
border-color: rgba(46, 229, 157, 0.35);  
}  
.footer {  
padding: 12px 18px 16px;  
color: var(--muted);  
font-size: 12px;  
border-top: 1px solid var(--border);  
}  
  
script.js  
  
// Tic-Tac-Toe (2 players, same device) + Appearance Customization (theme + mark style)  
const boardEl = document.getElementById("board");  
const statusEl = document.getElementById("status");  
const restartBtn = document.getElementById("restartBtn");  
const themeToggle = document.getElementById("themeToggle");  
const markStyleSelect = document.getElementById("markStyle");  
const scoreXEl = document.getElementById("scoreX");  
const scoreOEl = document.getElementById("scoreO");  
const scoreDEl = document.getElementById("scoreD");  
const p1LabelEl = document.getElementById("p1Label");  
const p2LabelEl = document.getElementById("p2Label");  
  
let board = Array(9).fill(null); // stores "X" / "O"  
  
let currentPlayer = "X";  
  
let gameOver = false;  
  
let scores = { X: 0, O: 0, D: 0 };  
  
const WIN_LINES = [  
[0, 1, 2],
```

```
[3, 4, 5],  
[6, 7, 8],  
[0, 3, 6],  
[1, 4, 7],  
[2, 5, 8],  
[0, 4, 8],  
[2, 4, 6],  
];  
  
// Appearance options  
  
const MARK_STYLES = {  
  
    classic: { X: "X", O: "O", p1: "Player X", p2: "Player O" },  
  
    emoji: { X: "✖️", O: "⭕", p1: "Player ✖️", p2: "Player ⭕" },  
  
    letters: { X: "A", O: "B", p1: "Player A", p2: "Player B" },  
  
};  
  
function getMarkStyle() {  
  
    return MARK_STYLES[markStyleSelect.value] || MARK_STYLES.classic;  
}  
  
function setStatus(text) {  
  
    statusEl.textContent = text;  
}  
  
function renderBoard() {  
  
    boardEl.innerHTML = "";  
  
    const style = getMarkStyle();  
  
    for (let i = 0; i < 9; i++) {  
  
        const btn = document.createElement("button");  
  
        btn.className = "cell";
```

```
btn.type = "button";
btn.setAttribute("role", "gridcell");
btn.setAttribute("aria-label", `Cell ${i + 1}`);
btn.dataset.index = String(i);
const value = board[i];
btn.textContent = value ? style[value] : "";
if (value || gameOver) btn.disabled = true;
btn.addEventListener("click", onCellClick);
boardEl.appendChild(btn);
}

// Update player labels for scoreboard
p1LabelEl.textContent = style.p1;
p2LabelEl.textContent = style.p2;
// Update scores UI
scoreXEl.textContent = String(scores.X);
scoreOEl.textContent = String(scores.O);
scoreDEl.textContent = String(scores.D);
}

function onCellClick(e) {
const idx = Number(e.currentTarget.dataset.index);
if (gameOver) return;
if (board[idx]) return;
board[idx] = currentPlayer;
const winnerInfo = getWinner(board);
if (winnerInfo) {
gameOver = true;
```

```
const style = getMarkStyle();

const winChar = style[winnerInfo.winner];

setStatus(`🎉 ${winChar} wins!`);

scores[winnerInfo.winner] += 1;

renderBoard();

highlightWinningCells(winnerInfo.line);

return;

}

if (isDraw(board)) {

gameOver = true;

setStatus("🤝 It's a draw!");

scores.D += 1;

renderBoard();

return;

}

currentPlayer = currentPlayer === "X" ? "O" : "X";

const style = getMarkStyle();

setStatus(`Player ${style[currentPlayer]}'s turn`);

renderBoard();

}

function getWinner(b) {

for (const line of WIN_LINES) {

const [a, c, d] = line;

if (b[a] && b[a] === b[c] && b[a] === b[d]) {

return { winner: b[a], line };

}

}
```

```
}

return null;
}

function isDraw(b) {
    return b.every((cell) => cell !== null);
}

function highlightWinningCells(line) {
    const cells = boardEl.querySelectorAll(".cell");
    line.forEach((i) => cells[i].classList.add("win"));
}

function restartGame({ keepScore = true } = {}) {
    board = Array(9).fill(null);
    currentPlayer = "X";
    gameOver = false;
    if (!keepScore) scores = { X: 0, O: 0, D: 0 };
    const style = getMarkStyle();
    setStatus(`Player ${style[currentPlayer]}'s turn`);
    renderBoard();
}

// Theme toggle (appearance customization)

function setTheme(theme) {
    document.documentElement.setAttribute("data-theme", theme);
    const isLight = theme === "light";
    themeToggle.textContent = isLight ? "Light" : "Dark";
    themeToggle.setAttribute("aria-pressed", String(isLight));
}
```

```
themeToggle.addEventListener("click", () => {
  const current = document.documentElement.getAttribute("data-theme") || "dark";
  setTheme(current === "light" ? "dark" : "light");
});

// Mark style change (appearance customization)
markStyleSelect.addEventListener("change", () => {
  // Keep the game state, just re-render with new appearance
  const style = getMarkStyle();
  if (!gameOver) setStatus(`Player ${style[currentPlayer]}'s turn`);
  renderBoard();
});

// Restart button
restartBtn.addEventListener("click", () => restartGame({ keepScore: true }));

// Initial setup
(function init() {
  // Default theme: dark (no attribute means dark variables apply)
  setTheme("dark");

  // Create board UI
  const style = getMarkStyle();
  setStatus(`Player ${style[currentPlayer]}'s turn`);
  renderBoard();
})();

Run locally
```

Option A (simplest)

Just double-click index.html (works fine for this project).

Option B (recommended)

Use a local server (prevents any future file/CORS issues):

VS Code → install Live Server → “Open with Live Server”

Deploy to GitHub Pages

Create a repo and upload these 3 files to the repo root.

On GitHub: Settings → Pages

Source: “Deploy from a branch”

Branch: main and folder /root

Save → open the Pages URL GitHub gives you.

Intent:

What were you trying to achieve at this point?

I was trying to generate the initial working version of the Tic-Tac-Toe web app that satisfies the lab requirements: a 3×3 grid, alternating turns, win/draw detection, restart without refreshing, and one extra feature (appearance customization) while remaining deployable on GitHub Pages.

Impact on the Application:

What changed in the application as a result?

This interaction produced the complete starter project structure (index.html, style.css, script.js) and a playable Tic-Tac-Toe game with a theme toggle and mark-style selector, plus basic status text and scoring.

Evaluation:

- What worked as expected?

The game ran locally, supported turn taking, win/draw logic, restart, and included appearance customization (theme + mark style). The folder structure also matched what is required for GitHub Pages.

- What did not work or required adjustment?

The UI messaging and “communicate what’s happening” requirement wasn’t strong enough yet (status text was minimal). Also, later testing showed the grid/cell visuals could feel inconsistent (emoji sizing and hover effects).

Decision:

What did you choose to do next because of this outcome?

I tested the game and then prompted the LLM again to improve UX communication

Selected Interaction 2

Prompt (verbatim):

<verbatim prompt text>

it didnt work at all, and i am using chrome btw, so what is the issue make a limited size of text u made it work once earlier but idk waht u are doing that it aintw working

LLM Output (verbatim or lightly trimmed):

<LLM response excerpt>

You're on **Chrome** — the reason it *still* looks like the **box “gets bigger” when clicked** is almost always this:

The real underlying issue (Chrome + <button>)

Chrome applies **built-in “active/focus” button styling** (UA styles). Even if you set width/height, Chrome can still apply subtle effects (like inner shadows, outlines, border rendering differences) that make the tile **look like it grows/shifts** when clicked.

The only reliable fix is to **completely reset the button styles** using:

- all: unset; (this wipes all browser default button styling)
- then re-apply ONLY the styles we want (width/height/border/etc.)

Also, you asked to **limit the size of the mark** (so the symbol doesn't dominate and visually “change” the tile). We'll do that too.

Below is the **full corrected code** (3 files). Copy/paste exactly.

index.html

```
<!doctype html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <title>Tic-Tac-Toe</title>
```

```
<link rel="stylesheet" href="style.css" />
</head>

<body>
<main class="app" id="app">
  <header class="header">
    <div>
      <h1>Tic-Tac-Toe</h1>
      <p class="sub">Two players • same device</p>
    </div>
  <div class="controls">
    <div class="control">
      <label class="label" for="themeToggle">Theme (current)</label>
      <button id="themeToggle" class="btn" type="button" aria-pressed="false">
        Dark
      </button>
    </div>
    <div class="control">
      <label class="label" for="markStyle">Mark style</label>
      <select id="markStyle" class="select">
        <option value="classic" selected>Classic (X / O)</option>
        <option value="emoji">Emoji (X / O)</option>
        <option value="letters">Letters (A / B)</option>
      </select>
    </div>
  </div>
</main>
```

```
</div>

<div class="control">
  <label class="label" for="restartBtn">Game</label>
  <button id="restartBtn" class="btn" type="button">
    Restart
  </button>
</div>
</div>
</header>

<section class="panel">
  <div id="status" class="status" role="status" aria-live="polite">
    Player X's turn — click an empty square.
  </div>

  <div id="hint" class="hint" aria-live="polite">
    Goal: get 3 in a row (horizontal, vertical, or diagonal).
  </div>

  <div class="scoreboard" aria-label="Scoreboard">
    <div class="pill"><span id="p1Label">Player X</span>: <strong id="scoreX">0</strong></div>
    <div class="pill">Draws: <strong id="scoreD">0</strong></div>
    <div class="pill"><span id="p2Label">Player O</span>: <strong id="scoreO">0</strong></div>
  </div>
```

```

</section>

<section class="board-wrap">
  <div id="board" class="board" role="grid" aria-label="Tic-Tac-Toe board"></div>
  <div class="micro">Tip: Restart anytime. Winning line highlights.</div>
</section>

<footer class="footer">
  <span>Made with HTML + CSS + JavaScript</span>
</footer>
</main>

<script src="script.js"></script>
</body>
</html>

```

style.css (THIS is the key fix: all: unset on .cell)

```

:root {
  --bg: #0b1020;
  --panel: rgba(255, 255, 255, 0.06);
  --panel-2: rgba(255, 255, 255, 0.10);
  --text: #e9ecf1;
  --muted: rgba(233, 236, 241, 0.70);
  --border: rgba(255, 255, 255, 0.14);
  --shadow: 0 18px 50px rgba(0, 0, 0, 0.35);
  --cell: rgba(255, 255, 255, 0.07);
}

```

```
--cell-hover: rgba(255, 255, 255, 0.12);  
--focus: rgba(124, 92, 255, 0.65);  
--win: rgba(46, 229, 157, 0.20);  
  
/* Hard-fixed board square size (stable) */  
--board-size: clamp(300px, 70vmin, 420px);  
--gap: 10px;  
--radius: 18px;  
  
/* Limited mark size (no huge symbols) */  
--mark-size: clamp(44px, 7.5vmin, 64px);  
}
```

```
[data-theme="light"] {  
    --bg: #f6f7fb;  
    --panel: rgba(0, 0, 0, 0.04);  
    --panel-2: rgba(0, 0, 0, 0.07);  
    --text: #141824;  
    --muted: rgba(20, 24, 36, 0.65);  
    --border: rgba(0, 0, 0, 0.10);  
    --shadow: 0 18px 50px rgba(18, 22, 33, 0.12);  
    --cell: rgba(0, 0, 0, 0.04);  
    --cell-hover: rgba(0, 0, 0, 0.07);  
    --focus: rgba(91, 73, 255, 0.45);  
    --win: rgba(10, 166, 107, 0.16);  
}
```

```
* { box-sizing: border-box; }

body {
  margin: 0;
  min-height: 100vh;
  display: grid;
  place-items: center;
  font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, Helvetica, Arial;
  background: radial-gradient(1200px 700px at 30% 15%, rgba(124, 92, 255, 0.25),
  transparent 55%),
    radial-gradient(1000px 600px at 75% 80%, rgba(46, 229, 157, 0.16), transparent
  55%),
  var(--bg);
  color: var(--text);
  padding: 20px;
}

.app {
  width: min(860px, 100%);
  background: var(--panel);
  border: 1px solid var(--border);
  border-radius: 18px;
  box-shadow: var(--shadow);
  overflow: hidden;
}
```

```
.header {  
  display: flex;  
  align-items: flex-start;  
  justify-content: space-between;  
  gap: 16px;  
  padding: 18px 18px 12px;  
  border-bottom: 1px solid var(--border);  
}  
  
h1 { margin: 0; font-size: 22px; letter-spacing: 0.2px; }  
.sub { margin: 6px 0 0; color: var(--muted); font-size: 13px; }
```

```
.controls {  
  display: flex;  
  flex-wrap: wrap;  
  gap: 10px;  
  justify-content: flex-end;  
}
```

```
.control { display: grid; gap: 6px; }  
.label { font-size: 12px; color: var(--muted); }
```

```
.btn, .select {  
  height: 36px;  
  padding: 0 12px;  
  border-radius: 12px;
```

```
border: 1px solid var(--border);  
background: var(--panel-2);  
color: var(--text);  
outline: none;  
}
```

```
.btn { cursor: pointer; font-weight: 700; }
```

```
.btn:focus-visible, .select:focus-visible {  
  box-shadow: 0 0 0 3px var(--focus);  
}
```

```
.panel {  
  padding: 14px 18px;  
  display: grid;  
  gap: 10px;  
  border-bottom: 1px solid var(--border);  
}
```

```
.status {  
  font-size: 14px;  
  padding: 10px 12px;  
  border-radius: 14px;  
  background: var(--panel-2);  
  border: 1px solid var(--border);  
}
```

```
.hint {  
  font-size: 12.5px;  
  color: var(--muted);  
  padding: 10px 12px;  
  border-radius: 14px;  
  background: transparent;  
  border: 1px dashed var(--border);  
}  
  
 .scoreboard { display: flex; gap: 10px; flex-wrap: wrap; }
```

```
.pill {  
  padding: 8px 10px;  
  border-radius: 999px;  
  background: var(--panel-2);  
  border: 1px solid var(--border);  
  font-size: 13px;  
  color: var(--muted);  
}  
 .pill strong { color: var(--text); }
```

```
.board-wrap {  
  padding: 18px;  
  display: grid;  
  place-items: center;
```

```
gap: 10px;  
}  
  
/* ✅ HARD fixed square board */  
.board {  
width: var(--board-size);  
height: var(--board-size);  
display: grid;  
grid-template-columns: repeat(3, 1fr);  
grid-template-rows: repeat(3, 1fr);  
gap: var(--gap);  
}  
  
/* ✅ FULL RESET of button default styles to stop Chrome “active” effects */  
.cell {  
all: unset; /* <- this is the real fix */  
box-sizing: border-box; /* re-enable sizing model */  
  
width: 100%;  
height: 100%;  
border: 1px solid var(--border);  
background: var(--cell);  
border-radius: var(--radius);  
  
display: grid;  
place-items: center;
```

```
cursor: pointer;  
user-select: none;  
  
/* Stop any visual growth */  
outline: none;  
box-shadow: none;  
  
/* Minor hover only (no transform) */  
transition: background 120ms ease;  
  
/* remove click highlight artifacts on Chrome */  
-webkit-tap-highlight-color: transparent;  
}
```

```
.cell:hover { background: var(--cell-hover); }
```

```
/* Focus ring INSIDE only, not changing perceived size */  
.cell:focus-visible {  
box-shadow: inset 0 0 0 4px var(--focus);  
}
```

```
.cell[disabled] {  
cursor: not-allowed;  
opacity: 0.95;  
}
```

```
/* ✅ Mark is limited size & cannot affect layout */

.mark {
    width: 72%;
    height: 72%;
    display: grid;
    place-items: center;

    font-size: var(--mark-size);
    font-weight: 900;
    line-height: 1;

    overflow: hidden;

    font-family: ui-sans-serif, system-ui, -apple-system, "Segoe UI",
        "Apple Color Emoji", "Segoe UI Emoji", "Noto Color Emoji", sans-serif;
}

/* Winning highlight stays INSIDE tile */

.cell.win {
    background: var(--win);
    box-shadow: inset 0 0 0 4px rgba(46, 229, 157, 0.55);
}

.micro { font-size: 12px; color: var(--muted); text-align: center; }
```

```
.footer {  
  padding: 12px 18px 16px;  
  color: var(--muted);  
  font-size: 12px;  
  border-top: 1px solid var(--border);  
}
```

script.js

```
const boardEl = document.getElementById("board");  
  
const statusEl = document.getElementById("status");  
  
const hintEl = document.getElementById("hint");  
  
const restartBtn = document.getElementById("restartBtn");  
  
  
const themeToggle = document.getElementById("themeToggle");  
  
const markStyleSelect = document.getElementById("markStyle");  
  
  
const scoreXEl = document.getElementById("scoreX");  
  
const scoreOEl = document.getElementById("scoreO");  
  
const scoreDEl = document.getElementById("scoreD");  
  
const p1LabelEl = document.getElementById("p1Label");  
  
const p2LabelEl = document.getElementById("p2Label");  
  
  
let board = Array(9).fill(null);  
  
let currentPlayer = "X";  
  
let gameOver = false;
```

```
let scores = { X: 0, O: 0, D: 0 };

const WIN_LINES = [
  [0, 1, 2],
  [3, 4, 5],
  [6, 7, 8],
  [0, 3, 6],
  [1, 4, 7],
  [2, 5, 8],
  [0, 4, 8],
  [2, 4, 6],
];

const MARK_STYLES = {
  classic: { X: "X", O: "O", p1: "Player X", p2: "Player O" },
  emoji: { X: "✖️", O: "⭕", p1: "Player ✖️", p2: "Player ⭕" },
  letters: { X: "A", O: "B", p1: "Player A", p2: "Player B" },
};

function getMarkStyle() {
  return MARK_STYLES[markStyleSelect.value] || MARK_STYLES.classic;
}

function setStatus(text) {
  statusEl.textContent = text;
}
```

```
function setHint(text) {  
    hintEl.textContent = text;  
}  
  
function updateTurnMessaging() {  
    const style = getMarkStyle();  
    setStatus(`Player ${style[currentPlayer]}'s turn — click an empty square.);  
    setHint("Goal: get 3 in a row (horizontal, vertical, or diagonal).");  
}  
  
function renderBoard() {  
    boardEl.innerHTML = "";  
    const style = getMarkStyle();  
  
    for (let i = 0; i < 9; i++) {  
        const btn = document.createElement("button");  
        btn.className = "cell";  
        btn.type = "button";  
        btn.setAttribute("role", "gridcell");  
        btn.setAttribute("aria-label", `Cell ${i + 1}`);  
        btn.dataset.index = String(i);  
  
        const span = document.createElement("span");  
        span.className = "mark";  
        span.textContent = board[i] ? style[board[i]] : "";  
        btn.appendChild(span);  
        boardEl.appendChild(btn);  
    }  
}
```

```
btn.appendChild(span);

if (board[i] || gameOver) btn.disabled = true;

btn.addEventListener("click", onCellClick);
boardEl.appendChild(btn);
}

p1LabelEl.textContent = style.p1;
p2LabelEl.textContent = style.p2;

scoreXEl.textContent = String(scores.X);
scoreOEl.textContent = String(scores.O);
scoreDEl.textContent = String(scores.D);
}

function onCellClick(e) {
const idx = Number(e.currentTarget.dataset.index);
if (gameOver || board[idx]) return;

board[idx] = currentPlayer;

const winnerInfo = getWinner(board);
if (winnerInfo) {
gameOver = true;
const style = getMarkStyle();
```

```
const winChar = style[winnerInfo.winner];

setStatus(`🎉 ${winChar} wins!`);

setHint("Press Restart to play a new round (no page refresh needed).");

scores[winnerInfo.winner] += 1;

renderBoard();

highlightWinningCells(winnerInfo.line);

return;

}

if (isDraw(board)) {

gameOver = true;

setStatus("🤝 It's a draw!");

setHint("No moves left. Press Restart to play again.");

scores.D += 1;

renderBoard();

return;

}

currentPlayer = currentPlayer === "X" ? "O" : "X";

updateTurnMessaging();

renderBoard();
```

```
}
```

```
function getWinner(b) {  
    for (const line of WIN_LINES) {  
        const [a, c, d] = line;  
        if (b[a] && b[a] === b[c] && b[a] === b[d]) {  
            return { winner: b[a], line };  
        }  
    }  
    return null;  
}
```

```
function isDraw(b) {  
    return b.every((cell) => cell !== null);  
}
```

```
function highlightWinningCells(line) {  
    const cells = boardEl.querySelectorAll(".cell");  
    line.forEach((i) => cells[i].classList.add("win"));  
}
```

```
function restartGame() {  
    board = Array(9).fill(null);  
    currentPlayer = "X";  
    gameOver = false;  
    updateTurnMessaging();
```

```
renderBoard();

}

function setTheme(theme) {
    document.documentElement.setAttribute("data-theme", theme);
    const isLight = theme === "light";
    themeToggle.textContent = isLight ? "Light" : "Dark";
    themeToggle.setAttribute("aria-pressed", String(isLight));
}

themeToggle.addEventListener("click", () => {
    const current = document.documentElement.getAttribute("data-theme") || "dark";
    setTheme(current === "light" ? "dark" : "light");
});

markStyleSelect.addEventListener("change", () => {
    if (!gameOver) updateTurnMessaging();
    renderBoard();
});

restartBtn.addEventListener("click", restartGame);

(function init() {
    setTheme("dark");
    updateTurnMessaging();
    renderBoard();
})
```

})();

Quick sanity check (to prove it's fixed)

After this version:

- the board has fixed width/height
- each cell is fixed
- we removed **all** browser button styles using all: unset
- clicking won't "grow" tiles because there's no UA :active styling left

If you still see movement after this, it's likely **your eye reading the focus ring as growth**. In that case, tell me if you want:

- **no focus ring at all** (completely static), or
- **a tiny dot indicator** instead of a ring.

But the version above is the cleanest "hard grid size" solution on Chrome.

Intent:

What were you trying to achieve at this point?

I was trying to eliminate the visual issue in Chrome where the Tic-Tac-Toe tiles appeared to "grow" or shift when clicked. My goal was to keep a perfectly static, hard-sized grid while still keeping focus accessibility and clear game UI.

Impact on the Application:

What changed in the application as a result?

The LLM introduced a full reset of the .cell button styling using all: unset to remove Chrome's default button active/focus effects. It also enforced a fixed square board size using explicit width and height, removed transform-based animations, and constrained mark size with a --mark-size CSS variable so symbols could not distort the look of the tiles.

Evaluation:

- What worked as expected?

The board became visually stable in Chrome: clicking no longer introduced the browser's default active/focus styling that created the illusion of resizing. The grid stayed the same size, and marks remained consistent due to the fixed mark container and size limits.

- What did not work or required adjustment?

After the grid stability improvements, emojis still looked slightly off-center vertically because of emoji font baseline differences (a rendering issue rather than a layout issue).

Decision:

What did you choose to do next because of this outcome?

I moved on to a targeted UI polish step: fixing emoji alignment so the emoji marks appear visually centered without affecting the hard grid or other mark styles.

Selected Interaction 3

Prompt (verbatim):

<verbatim prompt text>

ok finally its better but one issue right now is that the mark styles the emoji aren't centered is there a way to fix that? thank check and then lmk if not then its ok

LLM Output (verbatim or lightly trimmed):

<LLM response excerpt>

Yep — that's a known thing with emoji: different emoji fonts have slightly "off" baselines, so even if the container is perfectly centered, the glyph can *look* a bit low/high.

It is fixable without breaking your hard grid. The cleanest way is to center the emoji visually by applying a tiny vertical offset **only when the mark style is emoji**.

Here are the **full files** again with that fix baked in (we add a data-mark-style attribute on the board and a CSS rule to nudge emoji up a touch).

index.html

```
<!doctype html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <title>Tic-Tac-Toe</title>

    <link rel="stylesheet" href="style.css" />

  </head>
```

```
<body>

<main class="app" id="app">

<header class="header">

<div>

    <h1>Tic-Tac-Toe</h1>

    <p class="sub">Two players • same device</p>

</div>

<div class="controls">

    <div class="control">

        <label class="label" for="themeToggle">Theme (current)</label>

        <button id="themeToggle" class="btn" type="button" aria-pressed="false">

            Dark

        </button>

    </div>

    <div class="control">

        <label class="label" for="markStyle">Mark style</label>

        <select id="markStyle" class="select">

            <option value="classic" selected>Classic (X / O)</option>

            <option value="emoji">Emoji (X / O)</option>

            <option value="letters">Letters (A / B)</option>

        </select>

    </div>

</div>
```

```
<div class="control">  
  <label class="label" for="restartBtn">Game</label>  
  <button id="restartBtn" class="btn" type="button">  
    Restart  
  </button>  
</div>  
</div>  
</header>
```

```
<section class="panel">  
  <div id="status" class="status" role="status" aria-live="polite">  
    Player X's turn — click an empty square.  
  </div>
```

```
<div id="hint" class="hint" aria-live="polite">  
  Goal: get 3 in a row (horizontal, vertical, or diagonal).  
</div>
```

```
<div class="scoreboard" aria-label="Scoreboard">  
  <div class="pill"><span id="p1Label">Player X</span>: <strong  
    id="scoreX">0</strong></div>  
  <div class="pill">Draws: <strong id="scoreD">0</strong></div>  
  <div class="pill"><span id="p2Label">Player O</span>: <strong  
    id="scoreO">0</strong></div>  
</div>  
</section>
```

```

<section class="board-wrap">
  <div id="board" class="board" role="grid" aria-label="Tic-Tac-Toe board"></div>
  <div class="micro">Tip: Restart anytime. Winning line highlights.</div>
</section>

<footer class="footer">
  <span>Made with HTML + CSS + JavaScript</span>
</footer>
</main>

<script src="script.js"></script>
</body>
</html>

```

style.css (emoji centering fix)

Key addition is this block:

```
.board[data-mark-style="emoji"] .mark {
  transform: translateY(-2px);
}
```

Full file:

```
:root {
  --bg: #0b1020;
  --panel: rgba(255, 255, 255, 0.06);
  --panel-2: rgba(255, 255, 255, 0.10);
  --text: #e9ecf1;
  --muted: rgba(233, 236, 241, 0.70);
```

```
--border: rgba(255, 255, 255, 0.14);  
--shadow: 0 18px 50px rgba(0, 0, 0, 0.35);  
--cell: rgba(255, 255, 255, 0.07);  
--cell-hover: rgba(255, 255, 255, 0.12);  
--focus: rgba(124, 92, 255, 0.65);  
--win: rgba(46, 229, 157, 0.20);  
  
--board-size: clamp(300px, 70vmin, 420px);  
--gap: 10px;  
--radius: 18px;  
  
--mark-size: clamp(44px, 7.5vmin, 64px);  
}
```

```
[data-theme="light"] {  
--bg: #f6f7fb;  
--panel: rgba(0, 0, 0, 0.04);  
--panel-2: rgba(0, 0, 0, 0.07);  
--text: #141824;  
--muted: rgba(20, 24, 36, 0.65);  
--border: rgba(0, 0, 0, 0.10);  
--shadow: 0 18px 50px rgba(18, 22, 33, 0.12);  
--cell: rgba(0, 0, 0, 0.04);  
--cell-hover: rgba(0, 0, 0, 0.07);  
--focus: rgba(91, 73, 255, 0.45);  
--win: rgba(10, 166, 107, 0.16);
```

```
}
```

```
* { box-sizing: border-box; }
```

```
body {  
  margin: 0;  
  min-height: 100vh;  
  display: grid;  
  place-items: center;  
  font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, Helvetica, Arial;  
  background: radial-gradient(1200px 700px at 30% 15%, rgba(124, 92, 255, 0.25),  
    transparent 55%),  
    radial-gradient(1000px 600px at 75% 80%, rgba(46, 229, 157, 0.16), transparent  
    55%),  
    var(--bg);  
  color: var(--text);  
  padding: 20px;  
}
```

```
.app {  
  width: min(860px, 100%);  
  background: var(--panel);  
  border: 1px solid var(--border);  
  border-radius: 18px;  
  box-shadow: var(--shadow);  
  overflow: hidden;  
}
```

```
.header {  
  display: flex;  
  align-items: flex-start;  
  justify-content: space-between;  
  gap: 16px;  
  padding: 18px 18px 12px;  
  border-bottom: 1px solid var(--border);  
}  
  
h1 { margin: 0; font-size: 22px; letter-spacing: 0.2px; }  
.sub { margin: 6px 0 0; color: var(--muted); font-size: 13px; }
```

```
.controls {  
  display: flex;  
  flex-wrap: wrap;  
  gap: 10px;  
  justify-content: flex-end;  
}
```

```
.control { display: grid; gap: 6px; }  
.label { font-size: 12px; color: var(--muted); }
```

```
.btn, .select {  
  height: 36px;  
  padding: 0 12px;
```

```
border-radius: 12px;  
border: 1px solid var(--border);  
background: var(--panel-2);  
color: var(--text);  
outline: none;  
}  
  
.btn { cursor: pointer; font-weight: 700; }
```

```
.btn:focus-visible, .select:focus-visible {  
  box-shadow: 0 0 0 3px var(--focus);  
}
```

```
.panel {  
  padding: 14px 18px;  
  display: grid;  
  gap: 10px;  
  border-bottom: 1px solid var(--border);  
}
```

```
.status {  
  font-size: 14px;  
  padding: 10px 12px;  
  border-radius: 14px;  
  background: var(--panel-2);  
  border: 1px solid var(--border);
```

```
}
```

```
.hint {  
  font-size: 12.5px;  
  color: var(--muted);  
  padding: 10px 12px;  
  border-radius: 14px;  
  background: transparent;  
  border: 1px dashed var(--border);  
}
```

```
.scoreboard { display: flex; gap: 10px; flex-wrap: wrap; }
```

```
.pill {  
  padding: 8px 10px;  
  border-radius: 999px;  
  background: var(--panel-2);  
  border: 1px solid var(--border);  
  font-size: 13px;  
  color: var(--muted);  
}
```

```
.pill strong { color: var(--text); }
```

```
.board-wrap {  
  padding: 18px;  
  display: grid;
```

```
    place-items: center;  
    gap: 10px;  
}
```

```
.board {  
  width: var(--board-size);  
  height: var(--board-size);  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-template-rows: repeat(3, 1fr);  
  gap: var(--gap);  
}
```

```
.cell {  
  all: unset;  
  box-sizing: border-box;  
  
  width: 100%;  
  height: 100%;  
  border: 1px solid var(--border);  
  background: var(--cell);  
  border-radius: var(--radius);  
  
  display: grid;  
  place-items: center;
```

```
cursor: pointer;  
user-select: none;  
  
outline: none;  
box-shadow: none;  
  
transition: background 120ms ease;  
  
-webkit-tap-highlight-color: transparent;  
}
```

```
.cell:hover { background: var(--cell-hover); }
```

```
.cell:focus-visible {  
box-shadow: inset 0 0 0 4px var(--focus);  
}
```

```
.cell[disabled] {  
cursor: not-allowed;  
opacity: 0.95;  
}
```

```
.mark {  
width: 72%;  
height: 72%;  
display: grid;
```

```
place-items: center;

font-size: var(--mark-size);
font-weight: 900;
line-height: 1;

/* emoji/letters rendering stack */
font-family: ui-sans-serif, system-ui, -apple-system, "Segoe UI",
  "Apple Color Emoji", "Segoe UI Emoji", "Noto Color Emoji", sans-serif;
}
```

```
/* ✅ Fix: emoji glyphs often sit low — nudge them up slightly */

.board[data-mark-style="emoji"] .mark {
  transform: translateY(-2px);
}
```

```
/* Winning highlight stays inside */

.cell.win {
  background: var(--win);
  box-shadow: inset 0 0 0 4px rgba(46, 229, 157, 0.55);
}
```

```
.micro { font-size: 12px; color: var(--muted); text-align: center; }
```

```
.footer {
  padding: 12px 18px 16px;
```

```
color: var(--muted);  
font-size: 12px;  
border-top: 1px solid var(--border);  
}
```

 **script.js (adds data-mark-style so CSS can target emoji mode)**

```
const boardEl = document.getElementById("board");  
  
const statusEl = document.getElementById("status");  
  
const hintEl = document.getElementById("hint");  
  
const restartBtn = document.getElementById("restartBtn");  
  
  
const themeToggle = document.getElementById("themeToggle");  
const markStyleSelect = document.getElementById("markStyle");  
  
  
const scoreXEl = document.getElementById("scoreX");  
const scoreOEl = document.getElementById("scoreO");  
const scoreDEl = document.getElementById("scoreD");  
const p1LabelEl = document.getElementById("p1Label");  
const p2LabelEl = document.getElementById("p2Label");  
  
  
let board = Array(9).fill(null);  
let currentPlayer = "X";  
let gameOver = false;  
  
let scores = { X: 0, O: 0, D: 0 };
```

```
const WIN_LINES = [
  [0, 1, 2],
  [3, 4, 5],
  [6, 7, 8],
  [0, 3, 6],
  [1, 4, 7],
  [2, 5, 8],
  [0, 4, 8],
  [2, 4, 6],
];

const MARK_STYLES = {
  classic: { X: "X", O: "O", p1: "Player X", p2: "Player O" },
  emoji: { X: "X", O: "O", p1: "Player X", p2: "Player O" },
  letters: { X: "A", O: "B", p1: "Player A", p2: "Player B" },
};

function getMarkStyle() {
  return MARK_STYLES[markStyleSelect.value] || MARK_STYLES.classic;
}

function setStatus(text) {
  statusEl.textContent = text;
}

function setHint(text) {
```

```
hintEl.textContent = text;
}

function updateTurnMessaging() {
  const style = getMarkStyle();
  setStatus(`Player ${style[currentPlayer]}'s turn — click an empty square.`);
  setHint("Goal: get 3 in a row (horizontal, vertical, or diagonal).");
}

function renderBoard() {
  boardEl.innerHTML = "";
  // ✅ Let CSS know the current mark style (for emoji centering tweak)
  boardEl.dataset.markStyle = markStyleSelect.value;

  const style = getMarkStyle();

  for (let i = 0; i < 9; i++) {
    const btn = document.createElement("button");
    btn.className = "cell";
    btn.type = "button";
    btn.setAttribute("role", "gridcell");
    btn.setAttribute("aria-label", `Cell ${i + 1}`);
    btn.dataset.index = String(i);

    const span = document.createElement("span");

```

```
span.className = "mark";  
span.textContent = board[i] ? style[board[i]] : "";  
btn.appendChild(span);
```

```
if (board[i] || gameOver) btn.disabled = true;
```

```
btn.addEventListener("click", onCellClick);  
boardEl.appendChild(btn);  
}
```

```
p1LabelEl.textContent = style.p1;  
p2LabelEl.textContent = style.p2;
```

```
scoreXEl.textContent = String(scores.X);  
scoreOEl.textContent = String(scores.O);  
scoreDEl.textContent = String(scores.D);  
}
```

```
function onCellClick(e) {  
const idx = Number(e.currentTarget.dataset.index);  
if (gameOver || board[idx]) return;
```

```
board[idx] = currentPlayer;
```

```
const winnerInfo = getWinner(board);  
if (winnerInfo) {
```

```
gameOver = true;

const style = getMarkStyle();

const winChar = style[winnerInfo.winner];

setStatus(`🎉 ${winChar} wins!`);

setHint("Press Restart to play a new round (no page refresh needed).");

scores[winnerInfo.winner] += 1;

renderBoard();

highlightWinningCells(winnerInfo.line);

return;

}

if (isDraw(board)) {

    gameOver = true;

    setStatus("🤝 It's a draw!");

    setHint("No moves left. Press Restart to play again.");

    scores.D += 1;

    renderBoard();

    return;

}

currentPlayer = currentPlayer === "X" ? "O" : "X";
```

```
updateTurnMessaging();
renderBoard();
}

function getWinner(b) {
  for (const line of WIN_LINES) {
    const [a, c, d] = line;
    if (b[a] && b[a] === b[c] && b[a] === b[d]) {
      return { winner: b[a], line };
    }
  }
  return null;
}

function isDraw(b) {
  return b.every((cell) => cell !== null);
}

function highlightWinningCells(line) {
  const cells = boardEl.querySelectorAll(".cell");
  line.forEach((i) => cells[i].classList.add("win"));
}

function restartGame() {
  board = Array(9).fill(null);
  currentPlayer = "X";
}
```

```
gameOver = false;

updateTurnMessaging();

renderBoard();

}

function setTheme(theme) {

document.documentElement.setAttribute("data-theme", theme);

const isLight = theme === "light";

themeToggle.textContent = isLight ? "Light" : "Dark";

themeToggle.setAttribute("aria-pressed", String(isLight));

}

themeToggle.addEventListener("click", () => {

const current = document.documentElement.getAttribute("data-theme") || "dark";

setTheme(current === "light" ? "dark" : "light");

});

markStyleSelect.addEventListener("change", () => {

if (!gameOver) updateTurnMessaging();

renderBoard();

});

restartBtn.addEventListener("click", restartGame);

}

function init() {

setTheme("dark");
```

```
updateTurnMessaging();  
renderBoard();  
});
```

If it's *still* slightly off...

Emoji rendering differs across Windows installs/fonts. If you still see it, tweak just one number:

```
.board[data-mark-style="emoji"] .mark { transform: translateY(-2px); }
```

Try -1px or -3px until it looks perfect on your machine.

That's the standard fix — and it won't affect X/O or A/B at all.

Intent:

What were you trying to achieve at this point?

I wanted to correct a remaining UI issue where emoji marks looked slightly misaligned within cells, even though the mark container was centered. The goal was a minimal fix that only applies to emoji mode and does not break the stable grid layout.

Impact on the Application:

What changed in the application as a result?

The LLM added a data-mark-style attribute to the board element in JavaScript and introduced a CSS rule that applies a small vertical translation (e.g., translateY(-2px)) only when the mark style is set to emoji. This visually re-centers emoji marks while leaving classic and letters modes unchanged.

Evaluation:

- What worked as expected?

Emoji marks appeared properly centered to the eye, improving overall UI consistency. The grid remained locked and stable, and the fix did not affect X/O or A/B rendering.

- What did not work or required adjustment?

No additional functional issues were identified.

Decision:

What did you choose to do next because of this outcome?

Since the issue was fully resolved and no further UI defects were observed, no additional LLM interactions were required.

Optional Notes

Use this space to note any manual edits, unexpected behavior, or observations about your process.