TP Illustratif projet DAW : HTML, JS, CSS

/mon-projet-puzzle

│— index.html

│— style.css

│— script.js

│— puzzle.jpg <-- Ton image à découper

Le fichier **`puzzle.jpg`** doit être une image de **300×300 pixels** (ou toute autre taille carrée).

1. Code `index.html` (Interface du puzzle)

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Jeu de Puzzle</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>🧩 Jeu de Puzzle</h1>
  <div id="puzzle-container"></div>
  <button id="shuffleButton">Mélanger</button>
  <script src="script.js"></script>
</body>
</html>
```

2. Code `style.css` (Mise en page)

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
}

#puzzle-container {
    display: grid;
    grid-template-columns: repeat(3, 100px);
    grid-template-rows: repeat(3, 100px);
    gap: 2px;
    margin: 20px auto;
    width: 306px;
    height: 306px;
}

.piece {
    width: 100px;
    height: 100px;
    background-image: url("puzzle.jpg");
    background-size: 300px 300px;
    border: 1px solid #000;
    cursor: pointer;
}

.empty {
    background: none;
    border: none;
}
```

3. Code `script.js` (Logique du puzzle)

```
document.addEventListener("DOMContentLoaded", function () {
    const puzzleContainer = document.getElementById("puzzle-container");
    const shuffleButton = document.getElementById("shuffleButton");

    const imageUrl = "puzzle.jpg";
    let emptyIndex = 8;
    let pieces = [];

    function createPuzzle() {
        pieces = [];
        puzzleContainer.innerHTML = "";

        let positions = [...Array(9).keys()];
        positions = shuffleArray(positions);

        for (let i = 0; i < 9; i++) {
            const piece = document.createElement("div");
            piece.classList.add("piece");

            if (positions[i] !== 8) {
                let x = (positions[i] % 3) * -100;
                let y = Math.floor(positions[i] / 3) * -100;
                piece.style.backgroundPosition = `${x}px ${y}px`;
            } else {
                piece.classList.add("empty");
                emptyIndex = i;
            }

            piece.dataset.index = i;
            piece.dataset.correctIndex = positions[i];
```

```
        piece.addEventListener("click", movePiece);
        pieces.push(piece);
        puzzleContainer.appendChild(piece);
    }
}

function movePiece() {
    let clickedIndex = pieces.indexOf(this);

    if (isAdjacent(clickedIndex, emptyIndex)) {
        swap(clickedIndex, emptyIndex);
        emptyIndex = clickedIndex;
        renderPuzzle();

        if (isPuzzleSolved()) {
            setTimeout(() => alert("  Bravo, puzzle réussi !"), 300);
        }
    }
}

function swap(i, j) {
    [pieces[i].dataset.correctIndex, pieces[j].dataset.correctIndex] =
        [pieces[j].dataset.correctIndex, pieces[i].dataset.correctIndex];

    if (pieces[j].classList.contains("empty")) {
        pieces[j].classList.remove("empty");
        pieces[i].classList.add("empty");
    }
}

function renderPuzzle() {
```

```
    puzzleContainer.innerHTML = "";
    pieces.forEach(piece => puzzleContainer.appendChild(piece));
  }

  function isAdjacent(index1, index2) {
    const row1 = Math.floor(index1 / 3);
    const col1 = index1 % 3;
    const row2 = Math.floor(index2 / 3);
    const col2 = index2 % 3;
    return (Math.abs(row1 - row2) === 1 && col1 === col2) || (Math.abs(col1 - col2) === 1 && row1
=== row2);
  }

  function isPuzzleSolved() {
    return pieces.every((piece, index) => piece.dataset.correctIndex == index);
  }

  function shuffleArray(array) {
    do {
      for (let i = array.length - 1; i > 0; i--) {
        const j = Math.floor(Math.random() * (i + 1));
        [array[i], array[j]] = [array[j], array[i]];
      }
    } while (!isSolvable(array));
    return array;
  }

  function isSolvable(arr) {
    let inversions = 0;
    for (let i = 0; i < arr.length; i++) {
      for (let j = i + 1; j < arr.length; j++) {
```

```
            if (arr[i] > arr[j] && arr[j] !== 8) {
                inversions++;
            }
        }
    }
    return inversions % 2 === 0;
}

shuffleButton.addEventListener("click", createPuzzle);
createPuzzle();
});
```