Riswan Effendi

1. Cards game

HTML Section

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

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

  <title>Card Game</title>

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

</head>

<body>

  <div class="game">

    <h1>Higher Card Wins</h1>

    <label for="mode-select">Select Mode: </label>

    <select id="mode-select">

      <option value="random">Random Draw</option>

      <option value="sequential">Sequential Run</option>

      <option value="choose">Choose Cards</option>

    </select>

    <label for="player-card-select">Pilih Kartu Player:</label>

    <select id="player-card-select"></select>

    <label for="computer-card-select">Pilih Kartu Komputer:</label>

    <select id="computer-card-select"></select>

    <div class="cards">

      <div>

        <h2>You</h2>

        <div class="card" id="player-card">🂠</div>

      </div>

      <div>

        <h2>Computer</h2>

        <div class="card" id="computer-card">🂠</div>

      </div>

    </div>

    <button id="draw-btn">Draw Card</button>

    <p id="result"></p>

  </div>

  <script src="script.js"></script>

</body>

</html>

CSS Section

body {

  background-color: #0e0e0e;

  color: white;

  font-family: Arial, sans-serif;

  text-align: center;

  margin: 0;

  padding: 20px;

}

.game {

  max-width: 500px;

  margin: auto;

}

.cards {

  display: flex;

  justify-content: space-around;

  margin: 30px 0;

}

.card {

  font-size: 80px;

  background-color: white;

  color: black;

  width: 100px;

  height: 140px;

  display: flex;

  align-items: center;

  justify-content: center;

  border-radius: 10px;

  box-shadow: 0 0 10px #fff;

}

button {

  padding: 10px 30px;

  font-size: 16px;

  background-color: #4caf50;

  color: white;

  border: none;

  border-radius: 5px;

  cursor: pointer;

  margin-top: 10px;

}

#result {

  font-size: 20px;

  margin-top: 20px;

}

label, select {

  font-size: 16px;

  margin: 10px 5px 5px 5px;

  display: block;

}

/\* Hide card selects by default \*/

#player-card-select,

#computer-card-select,

label[for="player-card-select"],

label[for="computer-card-select"] {

  display: none;

}

/\* Show selects when mode is choose \*/

.mode-choose #player-card-select,

.mode-choose #computer-card-select,

.mode-choose label[for="player-card-select"],

.mode-choose label[for="computer-card-select"] {

  display: inline-block;

  margin-right: 10px;

}

JS Section

const suits = ['♠', '♥', '♦', '♣'];

const values = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A'];

function createAllCards() {

  const allCards = [];

  for (const value of values) {

    for (const suit of suits) {

      allCards.push({ value, suit });

    }

  }

  return allCards;

}

const allCards = createAllCards();

const playerCardSelect = document.getElementById('player-card-select');

const computerCardSelect = document.getElementById('computer-card-select');

function fillCardSelect(selectElem) {

  allCards.forEach(card => {

    const option = document.createElement('option');

    option.value = card.value + card.suit;

    option.textContent = card.value + card.suit;

    selectElem.appendChild(option);

  });

}

fillCardSelect(playerCardSelect);

fillCardSelect(computerCardSelect);

function parseCard(str) {

  // For "10♥" length=3, for "Q♠" length=2

  if (str.length === 3) {

    return { value: str.slice(0, 2), suit: str[2] };

  } else {

    return { value: str[0], suit: str[1] };

  }

}

function getCardStrength(value) {

  return values.indexOf(value);

}

// For sequential mode: keep index counters

let playerIndex = 0;

let computerIndex = 0;

const modeSelect = document.getElementById('mode-select');

const gameDiv = document.querySelector('.game');

modeSelect.addEventListener('change', () => {

  if (modeSelect.value === 'choose') {

    gameDiv.classList.add('mode-choose');

  } else {

    gameDiv.classList.remove('mode-choose');

  }

});

document.getElementById('draw-btn').addEventListener('click', () => {

  const mode = modeSelect.value;

  let playerCard, computerCard;

  if (mode === 'random') {

    playerCard = allCards[Math.floor(Math.random() \* allCards.length)];

    computerCard = allCards[Math.floor(Math.random() \* allCards.length)];

  } else if (mode === 'sequential') {

    playerCard = allCards[playerIndex % allCards.length];

    computerCard = allCards[computerIndex % allCards.length];

    playerIndex++;

    computerIndex++;

  } else if (mode === 'choose') {

    playerCard = parseCard(playerCardSelect.value);

    computerCard = parseCard(computerCardSelect.value);

  }

  document.getElementById('player-card').textContent = playerCard.value + playerCard.suit;

  document.getElementById('computer-card').textContent = computerCard.value + computerCard.suit;

  const playerStrength = getCardStrength(playerCard.value);

  const computerStrength = getCardStrength(computerCard.value);

  const result = document.getElementById('result');

  if (playerStrength > computerStrength) {

    result.textContent = 'You win!';

  } else if (playerStrength < computerStrength) {

    result.textContent = 'Computer wins!';

  } else {

    result.textContent = "It's a tie!";

  }

});

// Initialize mode UI state on page load

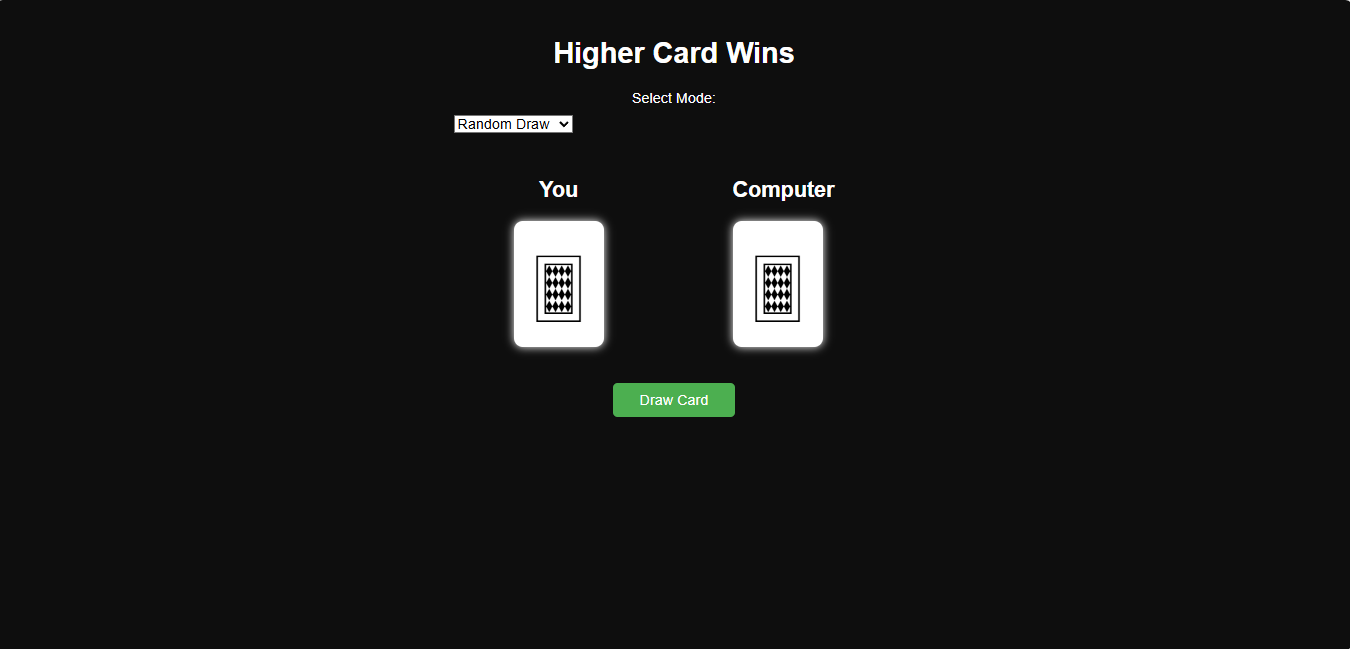
if (modeSelect.value === 'choose') {

  gameDiv.classList.add('mode-choose');

} else {

  gameDiv.classList.remove('mode-choose');

}



1. Memory Games

HTML Section

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Memory Matching Game</title>

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

</head>

<body>

<h1>Memory Matching Game</h1>

<div class="game-board" id="game-board"></div>

<p id="message"></p>

<button id="restart-btn">Restart Game</button>

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

</body>

</html>

CSS Section

body {

  background-color: #222;

  color: white;

  font-family: Arial, sans-serif;

  text-align: center;

  margin: 0;

  padding: 20px;

}

h1 {

  margin-bottom: 20px;

}

.game-board {

  width: 320px;

  margin: 0 auto;

  display: grid;

  grid-template-columns: repeat(4, 70px);

  grid-gap: 15px;

}

.card {

  width: 70px;

  height: 70px;

  background-color: #444;

  border-radius: 8px;

  cursor: pointer;

  display: flex;

  justify-content: center;

  align-items: center;

  font-size: 40px;

  user-select: none;

  color: #222;

  transition: background-color 0.3s, color 0.3s;

}

.card.flipped, .card.matched {

  background-color: white;

  color: black;

  cursor: default;

}

#message {

  margin-top: 20px;

  font-size: 20px;

  min-height: 24px;

}

#restart-btn {

  margin-top: 20px;

  padding: 10px 25px;

  font-size: 16px;

  border: none;

  border-radius: 5px;

  background-color: #4caf50;

  color: white;

  cursor: pointer;

}

JS Section

const emojis = ['🐶', '🐱', '🐭', '🐹', '🐰', '🦊', '🐻', '🐼'];

const gameBoard = document.getElementById('game-board');

const message = document.getElementById('message');

const restartBtn = document.getElementById('restart-btn');

let cards = [];

let flippedCards = [];

let matchedCount = 0;

function shuffle(array) {

  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]];

  }

  return array;

}

function createCards() {

  cards = [];

  // duplicate emojis to make pairs

  const pairedEmojis = emojis.concat(emojis);

  // shuffle pairs

  shuffle(pairedEmojis);

  pairedEmojis.forEach((emoji, index) => {

    const card = document.createElement('div');

    card.classList.add('card');

    card.dataset.emoji = emoji;

    card.dataset.index = index;

    card.textContent = ''; // hidden initially

    card.addEventListener('click', onCardClicked);

    cards.push(card);

    gameBoard.appendChild(card);

  });

}

function onCardClicked(e) {

  const card = e.currentTarget;

  if (

    card.classList.contains('flipped') ||

    card.classList.contains('matched') ||

    flippedCards.length === 2

  ) {

    return;

  }

  flipCard(card);

  flippedCards.push(card);

  if (flippedCards.length === 2) {

    checkForMatch();

  }

}

function flipCard(card) {

  card.classList.add('flipped');

  card.textContent = card.dataset.emoji;

}

function unflipCards() {

  flippedCards.forEach(card => {

    card.classList.remove('flipped');

    card.textContent = '';

  });

  flippedCards = [];

}

function checkForMatch() {

  const [card1, card2] = flippedCards;

  if (card1.dataset.emoji === card2.dataset.emoji) {

    card1.classList.add('matched');

    card2.classList.add('matched');

    matchedCount += 2;

    flippedCards = [];

    if (matchedCount === cards.length) {

      message.textContent = "Congrats! You've matched all pairs!";

    } else {

      message.textContent = "You found a match!";

    }

  } else {

    message.textContent = "No match. Try again!";

    setTimeout(() => {

      unflipCards();

      message.textContent = '';

    }, 1000);

  }

}

function restartGame() {

  matchedCount = 0;

  flippedCards = [];

  message.textContent = '';

  gameBoard.innerHTML = '';

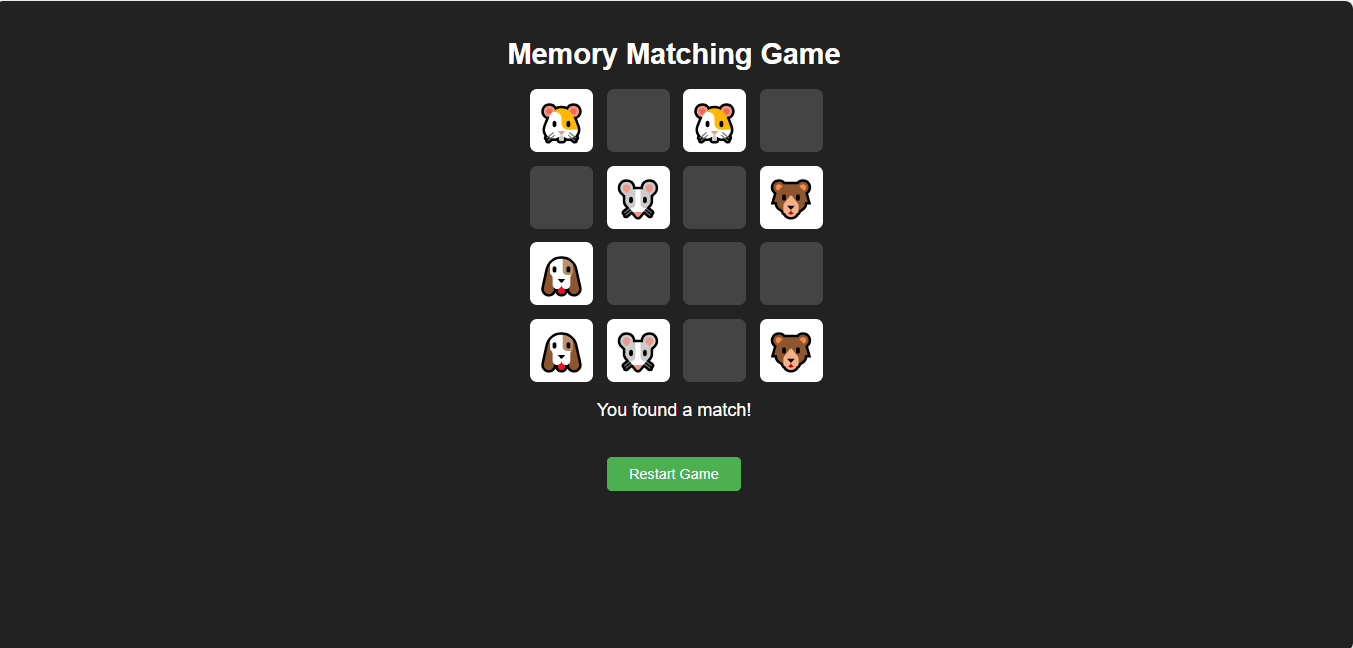
  createCards();

}

restartBtn.addEventListener('click', restartGame);

// Initialize the game on page load

restartGame();



1. Puzzle Number

HTML Section

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

  <title>4x4 Sliding Puzzle</title>

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

</head>

<body>

  <h1>4x4 Sliding Puzzle</h1>

  <div id="puzzle-container"></div>

  <button id="shuffle-button">Shuffle</button>

  <p id="message"></p>

  <script src="script.js"></script>

</body>

</html>

CSS Section

body {

  font-family: Arial, sans-serif;

  background: #2c3e50;

  color: white;

  text-align: center;

  padding: 20px;

}

#puzzle-container {

  display: grid;

  grid-template-columns: repeat(4, 80px);

  grid-gap: 6px;

  justify-content: center;

  margin: 20px auto;

}

.tile {

  width: 80px;

  height: 80px;

  background-color: #3498db;

  font-size: 24px;

  font-weight: bold;

  color: white;

  display: flex;

  justify-content: center;

  align-items: center;

  cursor: pointer;

  border-radius: 4px;

  user-select: none;

}

.empty {

  background-color: #34495e;

  cursor: default;

}

#shuffle-button {

  margin-top: 15px;

  padding: 10px 20px;

  font-size: 16px;

  background-color: #f39c12;

  color: white;

  border: none;

  border-radius: 5px;

  cursor: pointer;

}

#message {

  margin-top: 20px;

  font-size: 18px;

}

JS Section

const container = document.getElementById('puzzle-container');

const shuffleButton = document.getElementById('shuffle-button');

const message = document.getElementById('message');

const GRID\_SIZE = 4;

const TILE\_COUNT = GRID\_SIZE \* GRID\_SIZE;

let tiles = [];

function createTiles() {

  tiles = [...Array(TILE\_COUNT - 1).keys()].map(x => x + 1);

  tiles.push(null); // empty space

  renderTiles();

}

function renderTiles() {

  container.innerHTML = '';

  tiles.forEach((val, index) => {

    const tile = document.createElement('div');

    tile.classList.add('tile');

    if (val === null) {

      tile.classList.add('empty');

    } else {

      tile.textContent = val;

      tile.addEventListener('click', () => handleTileClick(index));

    }

    container.appendChild(tile);

  });

}

function handleTileClick(index) {

  const emptyIndex = tiles.indexOf(null);

  if (isAdjacent(index, emptyIndex)) {

    [tiles[index], tiles[emptyIndex]] = [tiles[emptyIndex], tiles[index]];

    renderTiles();

    checkWin();

  }

}

function isAdjacent(i1, i2) {

  const row1 = Math.floor(i1 / GRID\_SIZE);

  const row2 = Math.floor(i2 / GRID\_SIZE);

  const col1 = i1 % GRID\_SIZE;

  const col2 = i2 % GRID\_SIZE;

  return (Math.abs(row1 - row2) + Math.abs(col1 - col2)) === 1;

}

function shuffleTiles() {

  do {

    tiles = [...Array(TILE\_COUNT - 1).keys()].map(x => x + 1);

    tiles.push(null);

    tiles.sort(() => Math.random() - 0.5);

  } while (!isSolvable(tiles));

  renderTiles();

  message.textContent = '';

}

function isSolvable(array) {

  const invCount = array

    .filter(n => n !== null)

    .reduce((inv, val, i) => {

      for (let j = i + 1; j < array.length; j++) {

        if (array[j] !== null && array[j] < val) inv++;

      }

      return inv;

    }, 0);

  const emptyRowFromBottom = GRID\_SIZE - Math.floor(array.indexOf(null) / GRID\_SIZE);

  if (GRID\_SIZE % 2 === 0) {

    return (invCount + emptyRowFromBottom) % 2 === 0;

  } else {

    return invCount % 2 === 0;

  }

}

function checkWin() {

  const winState = [...Array(TILE\_COUNT - 1).keys()].map(x => x + 1).concat([null]);

  if (tiles.every((val, i) => val === winState[i])) {

    message.textContent = '🎉 You solved it!';

  }

}

shuffleButton.addEventListener('click', shuffleTiles);

createTiles();

