document.addEventListener('DOMContentLoaded', () => {

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

const scoreDisplay = document.getElementById('score');

const newGameBtn = document.getElementById('new-game-btn');

const gameOverOverlay = document.getElementById('game-over-overlay');

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

const gridSize = 4;

let board = [];

let score = 0;

let isGameOver = false;

// 初始化游戏

function initGame() {

board = Array(gridSize).fill(null).map(() => Array(gridSize).fill(0));

score = 0;

isGameOver = false;

updateScore();

gameOverOverlay.classList.add('hidden');

addRandomTile();

addRandomTile();

renderBoard();

}

// 渲染游戏棋盘

function renderBoard() {

gameBoard.innerHTML = '';

for (let r = 0; r < gridSize; r++) {

for (let c = 0; c < gridSize; c++) {

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

tile.className = 'tile';

const value = board[r][c];

if (value !== 0) {

tile.textContent = value;

tile.setAttribute('data-value', value);

}

gameBoard.appendChild(tile);

}

}

}

// 在随机空位添加新方块（90%为2，10%为4）

function addRandomTile() {

const emptyTiles = [];

for (let r = 0; r < gridSize; r++) {

for (let c = 0; c < gridSize; c++) {

if (board[r][c] === 0) {

emptyTiles.push({ r, c });

}

}

}

if (emptyTiles.length > 0) {

const { r, c } = emptyTiles[Math.floor(Math.random() \* emptyTiles.length)];

board[r][c] = Math.random() < 0.9 ? 2 : 4;

}

}

// 更新分数显示

function updateScore() {

scoreDisplay.textContent = score;

}

// 处理键盘输入

function handleInput(e) {

if (isGameOver) return;

let boardChanged = false;

switch (e.key) {

case 'ArrowUp':

boardChanged = moveUp();

break;

case 'ArrowDown':

boardChanged = moveDown();

break;

case 'ArrowLeft':

boardChanged = moveLeft();

break;

case 'ArrowRight':

boardChanged = moveRight();

break;

default:

return;

}

if (boardChanged) {

addRandomTile();

renderBoard();

if (checkGameOver()) {

endGame();

}

}

}

// 核心移动与合并逻辑

// 1. 滑动：将所有数字推到一边

// 2. 合并：合并相邻的相同数字

// 3. 再次滑动：填补合并后产生的空隙

function slide(row) {

// 过滤掉0，得到一个只含数字的数组

let filteredRow = row.filter(val => val);

let missing = gridSize - filteredRow.length;

// 在数组末尾补上0

let zeros = Array(missing).fill(0);

return filteredRow.concat(zeros);

}

function combine(row) {

for (let i = 0; i < gridSize - 1; i++) {

if (row[i] !== 0 && row[i] === row[i + 1]) {

row[i] \*= 2;

row[i + 1] = 0;

score += row[i];

updateScore();

}

}

return row;

}

// 向左移动

function moveLeft() {

let changed = false;

for (let r = 0; r < gridSize; r++) {

const originalRow = [...board[r]];

let row = board[r];

row = slide(row);

row = combine(row);

row = slide(row);

board[r] = row;

if (JSON.stringify(originalRow) !== JSON.stringify(board[r])) {

changed = true;

}

}

return changed;

}

// 向右移动

function moveRight() {

let changed = false;

for (let r = 0; r < gridSize; r++) {

const originalRow = [...board[r]];

let row = board[r];

row.reverse(); // 反转

row = slide(row);

row = combine(row);

row = slide(row);

row.reverse(); // 再次反转回来

board[r] = row;

if (JSON.stringify(originalRow) !== JSON.stringify(board[r])) {

changed = true;

}

}

return changed;

}

// 矩阵转置（行变列，列变行），用于上下移动

function transpose() {

const newBoard = Array(gridSize).fill(null).map(() => Array(gridSize).fill(0));

for (let r = 0; r < gridSize; r++) {

for (let c = 0; c < gridSize; c++) {

newBoard[c][r] = board[r][c];

}

}

board = newBoard;

}

// 向上移动 (转置 -> 左移 -> 转置回来)

function moveUp() {

transpose();

const changed = moveLeft();

transpose();

return changed;

}

// 向下移动 (转置 -> 右移 -> 转置回来)

function moveDown() {

transpose();

const changed = moveRight();

transpose();

return changed;

}

// 检查游戏是否结束

function checkGameOver() {

// 检查是否还有空位

for (let r = 0; r < gridSize; r++) {

for (let c = 0; c < gridSize; c++) {

if (board[r][c] === 0) {

return false; // 游戏未结束

}

}

}

// 检查水平方向是否还有可合并的方块

for (let r = 0; r < gridSize; r++) {

for (let c = 0; c < gridSize - 1; c++) {

if (board[r][c] === board[r][c + 1]) {

return false; // 游戏未结束

}

}

}

// 检查垂直方向是否还有可合并的方块

for (let c = 0; c < gridSize; c++) {

for (let r = 0; r < gridSize - 1; r++) {

if (board[r][c] === board[r + 1][c]) {

return false; // 游戏未结束

}

}

}

return true; // 游戏结束

}

// 游戏结束处理

function endGame() {

isGameOver = true;

gameOverOverlay.classList.remove('hidden');

}

// 事件监听

document.addEventListener('keydown', handleInput);

newGameBtn.addEventListener('click', initGame);

restartBtn.addEventListener('click', initGame);

// 启动游戏

initGame();

});