Double match game:

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

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

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

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

let score = 0;

let board = [];

const createBoard = () => {

for (let i = 0; i < 16; i++) {

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

tile.classList.add('tile');

gameBoard.appendChild(tile);

}

generateRandomTile();

generateRandomTile();

updateBoard();

};

const generateRandomTile = () => {

const emptyTiles = [];

board.forEach((tile, index) => {

if (tile === 0) emptyTiles.push(index);

});

const randomTile = emptyTiles[Math.floor(Math.random() \* emptyTiles.length)];

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

};

const updateBoard = () => {

const tiles = document.querySelectorAll('.tile');

tiles.forEach((tile, index) => {

tile.textContent = board[index] === 0 ? '' : board[index];

tile.className = 'tile';

if (board[index] > 0) tile.classList.add('tile-' + board[index]);

});

scoreDisplay.textContent = score;

};

const moveTiles = (direction) => {

const originalBoard = board.slice();

if (direction === 'up' || direction === 'down') {

for (let i = 0; i < 4; i++) {

const column = [board[i], board[i + 4], board[i + 8], board[i + 12]];

const newColumn = mergeTiles(column, direction === 'down');

for (let j = 0; j < 4; j++) {

board[i + j \* 4] = newColumn[j];

}

}

} else {

for (let i = 0; i < 4; i++) {

const row = board.slice(i \* 4, i \* 4 + 4);

const newRow = mergeTiles(row, direction === 'right');

for (let j = 0; j < 4; j++) {

board[i \* 4 + j] = newRow[j];

}

}

}

if (JSON.stringify(board) !== JSON.stringify(originalBoard)) {

generateRandomTile();

}

updateBoard();

};

const mergeTiles = (tiles, reverse = false) => {

if (reverse) tiles.reverse();

const merged = tiles.filter(tile => tile !== 0);

for (let i = 0; i < merged.length - 1; i++) {

if (merged[i] === merged[i + 1]) {

merged[i] \*= 2;

merged[i + 1] = 0;

score += merged[i];

}

}

const newTiles = merged.filter(tile => tile !== 0);

while (newTiles.length < 4) {

newTiles.push(0);

}

if (reverse) newTiles.reverse();

return newTiles;

};

const handleKeyPress = (event) => {

switch (event.key) {

case 'ArrowUp':

moveTiles('up');

break;

case 'ArrowDown':

moveTiles('down');

break;

case 'ArrowLeft':

moveTiles('left');

break;

case 'ArrowRight':

moveTiles('right');

break;

}

};

const newGame = () => {

board = Array(16).fill(0);

score = 0;

createBoard();

};

document.addEventListener('keydown', handleKeyPress);

newGameButton.addEventListener('click', newGame);

newGame();

});