```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport"
content="width=device-width, initial-
scale=1.0, maximum-scale=1.0, user-
scalable=no">
    <title>Match-3 Mobile Game</title>
    <style>
        body {
            margin: 0;
            overflow: hidden;
            touch-action: none;
        }
        #gameCanvas {
            background: #f0f0f0;
        }
    </style>
</head>
<body>
    <canvas id="gameCanvas"></canvas>
```

```
    <script>
        const canvas =
document.getElementById('gameCanvas');
        const ctx = canvas.getContext('2d');

        // Mobile screen adjustment
        function resizeCanvas() {
            canvas.width = window.innerWidth;
            canvas.height = window.innerHeight;
        }
        window.addEventListener('resize',
resizeCanvas);
        resizeCanvas();

        // Game Constants
        const TILE_SIZE = 60;
        const COLORS = ['#FF3366',
'#33CC99', '#FF9933', '#9966FF',
'#FFCC00'];
        let grid = [];
```

```
        let rows = 8;
        let cols = 6;

        // Initialize Grid
        function initGrid() {
            for(let i=0; i<rows; i++) {
                grid[i] = [];
                for(let j=0; j<cols; j++) {
                    grid[i][j] = {
                        type:
Math.floor(Math.random() *
COLORS.length),
                        x: j * TILE_SIZE,
                        y: i * TILE_SIZE
                    };
                }
            }
        }

        // Draw Game
        function draw() {
```

```
        ctx.clearRect(0, 0, canvas.width,
canvas.height);

        // Draw grid
        for(let i=0; i<rows; i++) {
            for(let j=0; j<cols; j++) {
                ctx.fillStyle = COLORS[grid[i]
[j].type];
                ctx.fillRect(grid[i][j].x, grid[i][j].y,
TILE_SIZE-2, TILE_SIZE-2);
            }
        }
    }

    // Touch Handling
    let touchStartX, touchStartY,
selectedTile;

    canvas.addEventListener('touchstart',
function(e) {
        e.preventDefault();
```

```
        const rect =
canvas.getBoundingClientRect();
        touchStartX = e.touches[0].clientX -
rect.left;
        touchStartY = e.touches[0].clientY -
rect.top;
        selectedTile =
getTileAtPosition(touchStartX,
touchStartY);
    });

    canvas.addEventListener('touchend',
function(e) {
        e.preventDefault();
        const rect =
canvas.getBoundingClientRect();
        const touchEndX =
e.changedTouches[0].clientX - rect.left;
        const touchEndY =
e.changedTouches[0].clientY - rect.top;
```

```
            if(selectedTile) {
                const direction =
getSwipeDirection(touchStartX,
touchStartY, touchEndX, touchEndY);
                handleSwipe(selectedTile,
direction);
            }
        });

        // Game Logic
        function getTileAtPosition(x, y) {
            const col = Math.floor(x /
TILE_SIZE);
            const row = Math.floor(y /
TILE_SIZE);
            if(row >=0 && row < rows && col >=0
&& col < cols) {
                return {row, col};
            }
            return null;
        }
```

```
        function getSwipeDirection(startX,
startY, endX, endY) {
            const dx = endX - startX;
            const dy = endY - startY;

            if(Math.abs(dx) > Math.abs(dy)) {
                return dx > 0 ? 'right' : 'left';
            } else {
                return dy > 0 ? 'down' : 'up';
            }
        }

        function handleSwipe(tile, direction) {
            let targetTile = null;

            switch(direction) {
                case 'left':
                    if(tile.col > 0) targetTile = {row:
tile.row, col: tile.col-1};
                    break;
```

```
            case 'right':
                if(tile.col < cols-1) targetTile =
{row: tile.row, col: tile.col+1};
                break;
            // Add up/down logic if needed
        }

        if(targetTile) {
            // Swap tiles
            const temp = grid[tile.row][tile.col];
            grid[tile.row][tile.col] =
grid[targetTile.row][targetTile.col];
            grid[targetTile.row][targetTile.col]
= temp;

            checkMatches();
        }
    }

    function checkMatches() {
        // Horizontal matches
```

```
        for(let i=0; i<rows; i++) {
            for(let j=0; j<cols-2; j++) {
                if(grid[i][j].type === grid[i]
[j+1].type &&
                    grid[i][j].type === grid[i]
[j+2].type) {
                    // Remove matched tiles
                    grid[i][j].type = -1;
                    grid[i][j+1].type = -1;
                    grid[i][j+2].type = -1;
                }
            }
        }

        // Add vertical match check here
        refillGrid();
    }

    function refillGrid() {
        for(let j=0; j<cols; j++) {
            let emptySpots = 0;
```

```
            for(let i=rows-1; i>=0; i--) {
                if(grid[i][j].type === -1) {
                    emptySpots++;
                } else if(emptySpots > 0) {
                    grid[i+emptySpots][j].type =
grid[i][j].type;
                    grid[i][j].type = -1;
                }
            }
            // Fill new tiles
            for(let i=0; i<emptySpots; i++) {
                grid[i][j].type =
Math.floor(Math.random() *
COLORS.length);
            }
        }
    }

    // Game Loop
    function gameLoop() {
        draw();
```

```
requestAnimationFrame(gameLoop);
        }

        // Start Game
        initGrid();
        gameLoop();
    </script>
</body>
</html>
```