```
{
"cells": [
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
   "vscode": {
    "languageId": "html"
   }
  },
  "outputs": [],
  "source": [
   "<!DOCTYPE html>\n",
   "<html lang=\"en\">\n",
   "<head>\n",
   "    <meta charset=\"UTF-8\">\n",
   "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\n",
   "    <title>2048 Game</title>\n",
   "    <style>\n",
   "        body {\n",
   "            font-family: \"Times New Roman\", sans-serif;\n",
   "            text-align: center;\n",
   "            margin: 20px;\n",
   "            color: rgb(255, 255, 255);\n",
   "        }\n",
   "\n",
   "        .grid-container {\n",
   "            display: grid;\n",
   "            grid-template-columns: repeat(4, 100px);\n",
   "            gap: 10px;\n",
   "            justify-items: center;\n",
   "            justify-content: center;\n",
   "            margin: 0 auto; /* Added to center the grid horizontally */\n",
   "            color: gold;\n",
   "        }\n",
   "\n",
   "        .grid-item {\n",
   "            width: 100px;\n",
   "            height: 100px;\n",
   "            border: 5px solid #000000;\n",
   "            display: flex;\n",
   "            align-items: center;\n",
   "            justify-content: center;\n",
   "            font-size: 24px;\n",
   "            font-weight: bold;\n",
   "            color: rgb(0, 0, 0);\n",
   "            background-color: #ffb703; \n",
   "        }\n",
   "\n",
   "        .game-over {\n",
   "            color: red;\n",
   "            font-size: 30px;\n",
   "            font-weight: bold;\n",
   "        }\n",
   "    </style>\n",
   "    <title>2048 Game</title>\n",
   "</head>\n",
   "<body>\n",
   "<body style=\"background-color:black;\">\n",
   "\n",
   "    <div>\n",
   "        <h1>2048 Game</h1>\n",
   "        <p>Use arrow keys to play!</p>\n",
   "        <div class=\"grid-container\" id=\"grid-container\"></div>\n",
   "        <p id=\"game-over\" id=\"gameover\"></p>\n",
   "    </div>\n",
   "\n",
   "    <script>\n",
   "        document.addEventListener('DOMContentLoaded', () => {\n",
   "            const gridSize = 4;\n",
   "            let grid = Array.from({\n",
   "                length: gridSize\n",
   "            }, () => Array(gridSize).fill(0));\n",
   "            let gameOver = false;\n",
   "\n",
   "            function initializeGame() {\n",
   "                addNewTile();\n",
   "                addNewTile();\n",
   "                updateGrid();\n",
   "            }\n",
   "\n",
   "            function addNewTile() {\n",
   "                const emptyCells = [];\n",
   "                grid.forEach((row, r) => {\n",
   "                    row.forEach((cell, c) => {\n",
   "                        if (cell === 0) {\n",
   "                            emptyCells.push([r, c]);\n",
   "                        }\n",
   "                    });\n",
   "                });\n",
   "\n",
   "                if (emptyCells.length > 0) {\n",
   "                    const [r, c] = \n",
   "                        emptyCells[\n",
   "                        ];\n",
   "                    grid[r][c] = Math.random() < 0.9 ? 2 : 4;\n",
   "                }\n",
   "            }\n",
   "\n",
   "            function updateGrid() {\n",
   "                const gridContainer = document.getElementById('grid-container');\n",
   "                gridContainer.innerHTML = '';\n",
   "\n",
   "                grid.forEach((row, r) => {\n",
   "                    row.forEach((cell, c) => {\n",
   "                        const gridItem = document.createElement('div');\n",
   "                        gridItem.classList.add('grid-item');\n",
   "                        gridItem.textContent = cell === 0 ? '' : binaryFormat(cell); // Convert to formatted binary\n",
   "                        gridContainer.appendChild(gridItem);\n",
   "                    });\n",
   "                });\n",
   "            }\n",
   "\n",
   "            function binaryFormat(value) {\n",
   "                // Convert the decimal value to binary and pad with leading zeros\n",
   "                return value.toString(2).padStart(4, '0');\n",
   "            }\n",
   "\n",
   "            function move(direction) {\n",
   "                if (gameOver) return;\n",
   "\n",
   "                let moved = false;\n",
   "\n",
   "                switch (direction) {\n",
   "                    case 'up':\n",
   "                        moved = moveUp();\n",
   "                        break;\n",
   "                    case 'down':\n",
   "                        moved = moveDown();\n",
   "                        break;\n",
   "                    case 'left':\n",
   "                        moved = moveLeft();\n",
   "                        break;\n",
   "                    case 'right':\n",
   "                        moved = moveRight();\n",
   "                        break;\n",
   "                }\n",
   "\n",
   "                if (moved) {\n",
   "                    addNewTile();\n",
   "                    updateGrid();\n",
   "                    checkGameOver();\n",
   "                }\n",
   "            }\n",
   "\n",
   "            function moveUp() {\n",
   "                let moved = false;\n",
   "\n",
   "                for (let j = 0; j < gridSize; j++) {\n",
   "                    for (let i = 1; i < gridSize; i++) {\n",
   "                        if (grid[i][j] !== 0) {\n",
   "                            let row = i-1;\n",
   "                            while (row > 0 && grid[row - 1][j] === 0) {\n",
   "                                grid[row - 1][j] = grid[row][j];\n",
   "                                grid[row][j] = 0;\n",
   "                                row--;\n",
   "                                moved = true;\n",
   "                            }\n",
   "                            if (row > 0 && grid[row - 1][j] === grid[row][j]) {\n",
   "                                grid[row - 1][j] *= 2;\n",
   "                                grid[row][j] = 0;\n",
   "                                moved = true;\n",
   "                            }\n",
   "                        }\n",
   "                    }\n",
   "                }\n",
   "\n",
   "                return moved;\n",
   "            }\n",
   "\n",
   "            function moveDown() {\n",
   "                let moved = false;\n",
   "\n",
   "                for (let j = 0; j < gridSize; j++) {\n",
   "                    for (let i = gridSize - 2; i >= 0; i--) {\n",
   "                        if (grid[i][j] !== 0) {\n",
   "                            let row = i+1;\n",
   "                            while (row < gridSize - 1 && grid[row + 1][j] === 0) {\n",
   "                                grid[row + 1][j] = grid[row][j];\n",
   "                                grid[row][j] = 0;\n",
   "                                row++;\n",
   "                                moved = true;\n",
   "                            }\n",
   "                            if (row < gridSize - 1 && grid[row + 1][j] === grid[row][j]) {\n",
   "                                grid[row + 1][j] *= 2;\n",
   "                                grid[row][j] = 0;\n",
   "                                moved = true;\n",
   "                            }\n",
   "                        }\n",
   "                    }\n",
   "                }\n",
   "\n",
   "                return moved;\n",
   "            }\n",
   "\n",
   "            function moveLeft() {\n",
   "                let moved = false;\n",
   "\n",
   "                for (let i = 0; i < gridSize; i++) {\n",
   "                    for (let j = 1; j < gridSize; j++) {\n",
   "                        if (grid[i][j] !== 0) {\n",
   "                            let col = j-1;\n",
   "                            while (col > 0 && grid[i][col - 1] === 0) {\n",
   "                                grid[i][col - 1] = grid[i][col];\n",
   "                                grid[i][col] = 0;\n",
   "                                col--;\n",
   "                                moved = true;\n",
   "                            }\n",
   "                            if (col > 0 && grid[i][col - 1] === grid[i][col]) {\n",
   "                                grid[i][col - 1] *= 2;\n",
   "                                grid[i][col] = 0;\n",
   "                                moved = true;\n",
   "                            }\n",
   "                        }\n",
   "                    }\n",
   "                }\n",
   "\n",
   "                return moved;\n",
   "            }\n",
   "\n",
   "            function moveRight() {\n",
   "                let moved = false;\n",
   "\n",
   "                for (let i = 0; i < gridSize; i++) {\n",
   "                    for (let j = gridSize - 2; j >= 0; j--) {\n",
   "                        if (grid[i][j] !== 0) {\n",
   "                            let col = j+1;\n",
   "                            while (col < gridSize - 1 && grid[i][col + 1] === 0) {\n",
   "                                grid[i][col + 1] = grid[i][col];\n",
   "                                grid[i][col] = 0;\n",
   "                                col++;\n",
   "                                moved = true;\n",
   "                            }\n",
   "                            if (col < gridSize - 1 && grid[i][col + 1] === grid[i][col]) {\n",
   "                                grid[i][col + 1] *= 2;\n",
   "                                grid[i][col] = 0;\n",
   "                                moved = true;\n",
   "                            }\n",
   "                        }\n",
   "                    }\n",
   "                }\n",
   "\n",
   "                return moved;\n",
   "            }\n",
   "\n",
   "            function checkGameOver() {\n",
   "                // Check for game over conditions\n",
   "                let isGameOver = true;\n",
   "\n",
   "                // Check if there are any adjacent equal numbers or empty cells\n",
   "                for (let i = 0; i < gridSize; i++) {\n",
   "                    for (let j = 0; j < gridSize; j++) {\n",
   "                        if (\n",
   "                            (i < gridSize - 1 && grid[i][j] === grid[i + 1][j]) ||\n",
   "                            (j < gridSize - 1 && grid[i][j] === grid[i][j + 1]) ||\n",
   "                            grid[i][j] === 0\n",
   "                        ) {\n",
   "                            isGameOver = false;\n",
   "                            break;\n",
   "                        }\n",
   "                    }\n",
   "                    if (!isGameOver) {\n",
   "                        break;\n",
   "                    }\n",
   "                }\n",
   "\n",
   "                if (isGameOver) {\n",
   "                    document.getElementById('game-over').innerText = 'Game Over!';\n",
   "                    gameOver = true;\n",
   "                }\n",
   "            }\n",
   "\n",
   "            // Event listener for keyboard input\n",
   "            document.addEventListener('keydown', (event) => {\n",
   "                if (!gameOver) {\n",
   "                    switch (event.key) {\n",
   "                        case 'ArrowUp':\n",
   "                            move('up');\n",
   "                            break;\n",
   "                        case 'ArrowDown':\n",
   "                            move('down');\n",
   "                            break;\n",
   "                        case 'ArrowLeft':\n",
   "                            move('left');\n",
   "                            break;\n",
   "                        case 'ArrowRight':\n",
   "                            move('right');\n",
   "                            break;\n",
   "                    }\n",
   "                }\n",
   "            });\n",
   "\n",
   "            // Initialize the game\n",
   "            initializeGame();\n",
   "        });\n",
   "    </script>\n",
   "</body>\n",
   "</html>"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
   "vscode": {
    "languageId": "html"
   }
  },
  "outputs": [],
  "source": [
   "<!DOCTYPE html>\n",
   "<html lang=\"en\">\n",
   "<head>\n",
   "    <meta charset=\"UTF-8\">\n",
   "    <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\n",
   "    <title>Custom Background</title>\n",
   "</head>\n",
   "<body>\n",
   "    <h1> Color Game Instructions </h1>\n",
   "    <p>\n",
   "        Welcome to the color game! <br>\n",
   "        In this game wegaga you to try to create a color using the RGB values to match each of the color squares you see on top!<br>\n",
   "        Fill in the fields for each RGB to create a new color!<br>\n",
   "    </p>\n",
   "\n",
   "    <!-- Use of a collection type to represent a list of square images that is stored in order to match \n",
   "    the squares with the RGB values of the backgrounde-->\n",
   "    <p>\n",
   "        <li><img src=\"redsquare.jpg\" id=\"redSquare\" width=\"auto\" height=\"100\" length=\"100\"></img></li>\n",
   "        <li><img src=\"yellowsquare.jpg\" id=\"yellowSquare\" width=\"auto\" height=\"100\" length=\"100\"></img></li>\n",
   "        <li><img src=\"bluesquare.jpg\" id=\"blueSquare\" width=\"auto\" height=\"100\" length=\"100\"></img></li>\n",
   "        <li><img src=\"greensquare.jpg\" id=\"greenSquare\" width=\"auto\" height=\"100\" length=\"100\"></img></li>\n",
   "        <li><img src=\"blueorange.jpg\" id=\"blueOrange\" width=\"auto\" height=\"100\" length=\"100\"></img></li>\n",
   "        <li><img src=\"purplesquare.jpg\" id=\"purpleSquare\" width=\"auto\" height=\"100\" length=\"100\"></img></li>\n",
   "    </p>\n",
   "\n",
   "</body>\n",
   "</html>\n",
   "\n",
   "\n",
   "<head>\n",
   "    <label for=\"red\">Red:</label>\n",
   "    <input type=\"number\" id=\"red\" min=\"0\" max=\"255\" value=\"0\"> <!-- Instructions for input-->\n",
   "    <br>\n",
   "    <label for=\"green\">Green:</label>\n",
   "    <input type=\"number\" id=\"green\" min=\"0\" max=\"255\" value=\"0\"> <!-- Instructions for input-->\n",
   "    <br>\n",
   "    <label for=\"blue\">Blue:</label>\n",
   "    <input type=\"number\" id=\"blue\" min=\"0\" max=\"255\" value=\"0\"> <!-- Instructions for input-->\n",
   "    <br>\n",
   "    <button onclick=\"updateBackground()\">Update Background</button>\n",
   "\n",
   "    <div id=\"background\" style=\"width: 100vw; height: 100vh;\"></div>\n",
   "\n",
   "    <script>\n",
   "        function updateBackground() {\n",
   "            var red = document.getElementById('red').value;\n",
   "            var green = document.getElementById('green').value;\n",
   "            var blue = document.getElementById('blue').value;\n",
   "\n",
   "            var binaryRed = decimalToBinary(red);\n",
   "            var binaryGreen = decimalToBinary(green);\n",
   "            var binaryBlue = decimalToBinary(blue);\n",
   "\n",
   "            var backgroundColor = `rgb(${red}, ${green}, ${blue})`;\n",
   "\n",
   "            document.getElementById(\"background\").style.backgroundColor = backgroundColor;\n",
   "        }\n",
   "\n",
   "        function decimalToBinary(decimal) {\n",
   "            return decimal.toString(2).padStart(8, '0');\n",
   "        }\n",
   "    </script>\n",
   "    <script src=\"squares.js\"></script>\n",
   "    <script src=\"rgbslider.js\"></script>\n",
   "</body>\n",
   "</html>"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
   "vscode": {
    "languageId": "javascript"
   }
  },
  "outputs": [],
  "source": [
   "squares.js\n",
   "\n",
   "window.addEventListener('load', function () {\n",
   "    // Create an image element\n",
   "    const img = document.createElement('img');\n",
   "\n",
   "    // Set the source and other attributes\n",
   "    img.src = selectImageBasis();\n",
   "    img.alt = 'Color Square';\n",
   "    img.width = 100;\n",
   "    img.height = 100;\n",
   "\n",
   "    // Append the image to the body\n",
   "    document.body.appendChild(img);\n",
   "});"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
   "vscode": {
    "languageId": "javascript"
   }
  },
  "outputs": [],
  "source": [
   "rgbslider.js\n",
   "\n",
   "function mycolor() {\n",
   "    \n",
   "    var red = decimalToBinary(document.getElementById('red').value);\n",
   "\n",
   "    var green = decimalToBinary(document.getElementById('green').value);\n",
   "\n",
   "    var blue = decimalToBinary(document.getElementById('blue').value);\n",
   "\n",
   "    var color = 'rgb(' + binaryToDecimal(red) + ',' + binaryToDecimal(green) + ',' + binaryToDecimal(blue) + ')';\n",
   "\n",
   "    document.body.style.backgroundColor = color;\n",
   "\n",
   "    document.getElementById('box').value = color;\n",
   "}\n",
   "\n",
   "document.getElementById('red').\n",
   "    addEventListener('input', mycolor);\n",
   "\n",
   "document.getElementById('green').\n",
   "    addEventListener('input', mycolor);\n",
   "\n",
   "document.getElementById('blue').\n",
   "    addEventListener('input', mycolor);\n",
   "\n",
   "function binaryToDecimal(binary) {\n",
   "    return parseInt(binary, 2); // parseInt is a procedure that contributes to the program's intended purpose, where I have defined:\n",
   "    // the procedure's name (binaryToDecimal)\n",
   "    // the return type (integer value)\n",
   "    // one or more parameters (binary, 2)\n",
   "}\n",
   "\n",
   "function decimalToBinary(decimal) {\n",
   "    return decimal.toString(2).padStart(8, '0');\n",
   "}"
  ]
 }
],
"metadata": {
 "kernelspec": {
  "display_name": "Python 3",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
   "version": 3
  },
  "file_extension": ".py",
  "mimetype": "text/x-python",
  "name": "python",
  "nbconvert_exporter": "python",
  "pygments_lexer": "ipython3",
  "version": "3.10.12"
 }
},
"nbformat": 4,
"nbformat_minor": 2
}
```