Introduction

In this blog, you  will learn basic concepts of  puzzle game in JavaScript and purpose of JavaScript used in  Applications. It gives the puzzle game an initial stage to start within JavaScript. JavaScript is a scripting language, it's used for within HTML Web pages. It is a lightweight programming language. JavaScript code is used directly in HTML pages.

**Step 1**

First, open a Sublime Text editor,

* Open start -> Sublime Text Editor.
* Go to file -> File -> New File

**Step 2**

Now, that in the below sample, you can insert CSS styles inside the <style> tag.

1. html {
2. **height**: 100%;
3. }
5. body {
6. **height**: 100%;
7. **background-image**: linear-gradient(#7B9F35, #226666);
8. **display**: flex;
9. justify-**content**: center;
10. align-items: center;
11. flex-**direction**: column;
12. }
14. .game {
15. /\* position: absolute;
16. top: 50%;
17. left: 50%;
18. transform: translate(-50%, -50%); \*/
19. box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5);
20. **padding**: 15px;
21. **background-color**: #AA3939;
22. border-radius: 5px;
23. }
25. .grid {
26. **display**: grid;
27. grid-template-columns: 80px 80px 80px 80px;
28. grid-template-rows: 80px 80px 80px 80px;
29. **border**: 1px solid #550000;
30. }
32. .grid button {
33. **background-color**: #cfcfcf;
34. **color**: #003333;
35. **font-size**: 24px;
36. **font-weight**: bold;
37. **border**: 1px solid #550000;
38. **outline**: none;
39. **cursor**: pointer;
40. }
42. .footer {
43. **margin-top**: 15px;
44. **display**: flex;
45. justify-**content**: space-between;
46. }
48. .footer button {
49. **border**: none;
50. **font-size**: 20px;
51. **font-weight**: bold;
52. border-radius: 5px;
53. box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5);
54. **padding**: 5px;
55. **width**: 80px;
56. **background-color**: #D4EE9F;
57. **color**: #003333;
58. **outline**: none;
59. **cursor**: pointer;
60. }
62. .footer button:hover {
63. **color**: #D4EE9F;
64. **background-color**: #003333;
65. }
67. .footer span {
68. flex: 1;
69. **text-align**: center;
70. **font-size**: 20px;
71. **color**: #D4EE9F;
72. **font-weight**: bold;
73. **margin**: auto 0;
74. }
76. .message {
77. **color**:#AA3939;
78. **height**: 80px;
79. }

**Step 3**

Now let's write a function to an object in JavaScript. In the following example, you will see how to do that with the following code.

1. **class** Box {
2. constructor(x, y) {
3. **this**.x = x;
4. **this**.y = y;
5. }
7. getTopBox() {
8. **if** (**this**.y === 0) **return** **null**;
9. **return** **new** Box(**this**.x, **this**.y - 1);
10. }
12. getRightBox() {
13. **if** (**this**.x === 3) **return** **null**;
14. **return** **new** Box(**this**.x + 1, **this**.y);
15. }
17. getBottomBox() {
18. **if** (**this**.y === 3) **return** **null**;
19. **return** **new** Box(**this**.x, **this**.y + 1);
20. }
22. getLeftBox() {
23. **if** (**this**.x === 0) **return** **null**;
24. **return** **new** Box(**this**.x - 1, **this**.y);
25. }

**Step 4**

In the  following code, you declared a function called “getNextdoorBoxes()” and takes four parameters. In the next line, we are creating one object of the “getRandomNextdoorBox()” function and at the time of creation we pass two parameters into it.

1. getNextdoorBoxes() {
2. **return** [
3. **this**.getTopBox(),
4. **this**.getRightBox(),
5. **this**.getBottomBox(),
6. **this**.getLeftBox()
7. ].filter(box => box !== **null**);
8. }
10. getRandomNextdoorBox() {
11. **const** nextdoorBoxes = **this**.getNextdoorBoxes();
12. **return** nextdoorBoxes[Math.floor(Math.random() \* nextdoorBoxes.length)];
13. }
14. }
16. **const** swapBoxes = (grid, box1, box2) => {
17. **const** temp = grid[box1.y][box1.x];
18. grid[box1.y][box1.x] = grid[box2.y][box2.x];
19. grid[box2.y][box2.x] = temp;
20. };
22. **const** isSolved = grid => {
23. **return** (
24. grid[0][0] === 1 &&
25. grid[0][1] === 2 &&
26. grid[0][2] === 3 &&
27. grid[0][3] === 4 &&
28. grid[1][0] === 5 &&
29. grid[1][1] === 6 &&
30. grid[1][2] === 7 &&
31. grid[1][3] === 8 &&
32. grid[2][0] === 9 &&
33. grid[2][1] === 10 &&
34. grid[2][2] === 11 &&
35. grid[2][3] === 12 &&
36. grid[3][0] === 13 &&
37. grid[3][1] === 14 &&
38. grid[3][2] === 15 &&
39. grid[3][3] === 0
40. );
41. };
43. **const** getRandomGrid = () => {
44. let grid = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 0]];
46. // Shuffle
47. let blankBox = **new** Box(3, 3);
48. **for** (let i = 0; i < 1000; i++) {
49. **const** randomNextdoorBox = blankBox.getRandomNextdoorBox();
50. swapBoxes(grid, blankBox, randomNextdoorBox);
51. blankBox = randomNextdoorBox;
52. }
54. **if** (isSolved(grid)) **return** getRandomGrid();
55. **return** grid;
56. };
58. **class** State {
59. constructor(grid, move, time, status) {
60. **this**.grid = grid;
61. **this**.move = move;
62. **this**.time = time;
63. **this**.status = status;
64. }
66. **static** ready() {
67. **return** **new** State(
68. [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]],
69. 0,
70. 0,
71. "ready"
72. );
73. }
75. **static** start() {
76. **return** **new** State(getRandomGrid(), 0, 0, "playing");
77. }
78. }
80. **class** Game {
81. constructor(state) {
82. **this**.state = state;
83. **this**.tickId = **null**;
84. **this**.tick = **this**.tick.bind(**this**);
85. **this**.render();
86. **this**.handleClickBox = **this**.handleClickBox.bind(**this**);
87. }
89. **static** ready() {
90. **return** **new** Game(State.ready());
91. }
93. tick() {
94. **this**.setState({ time: **this**.state.time + 1 });
95. }
97. setState(newState) {
98. **this**.state = { ...**this**.state, ...newState };
99. **this**.render();
100. }
102. handleClickBox(box) {
103. **return** **function**() {
104. **const** nextdoorBoxes = box.getNextdoorBoxes();
105. **const** blankBox = nextdoorBoxes.find(
106. nextdoorBox => **this**.state.grid[nextdoorBox.y][nextdoorBox.x] === 0
107. );
108. **if** (blankBox) {
109. **const** newGrid = [...**this**.state.grid];
110. swapBoxes(newGrid, box, blankBox);
111. **if** (isSolved(newGrid)) {
112. clearInterval(**this**.tickId);
113. **this**.setState({
114. status: "won",
115. grid: newGrid,
116. move: **this**.state.move + 1
117. });
118. } **else** {
119. **this**.setState({
120. grid: newGrid,
121. move: **this**.state.move + 1
122. });
123. }
124. }
125. }.bind(**this**);
126. }
128. render() {
129. **const** { grid, move, time, status } = **this**.state;
131. // Render grid
132. **const** newGrid = document.createElement("div");
133. newGrid.className = "grid";
134. **for** (let i = 0; i < 4; i++) {
135. **for** (let j = 0; j < 4; j++) {
136. **const** button = document.createElement("button");
138. **if** (status === "playing") {
139. button.addEventListener("click", **this**.handleClickBox(**new** Box(j, i)));
140. }
142. button.textContent = grid[i][j] === 0 ? "" : grid[i][j].toString();
143. newGrid.appendChild(button);
144. }
145. }
146. document.querySelector(".grid").replaceWith(newGrid);
148. // Render button
149. **const** newButton = document.createElement("button");
150. **if** (status === "ready") newButton.textContent = "Play";
151. **if** (status === "playing") newButton.textContent = "Reset";
152. **if** (status === "won") newButton.textContent = "Play";
153. newButton.addEventListener("click", () => {
154. clearInterval(**this**.tickId);
155. **this**.tickId = setInterval(**this**.tick, 1000);
156. **this**.setState(State.start());
157. });
158. document.querySelector(".footer button").replaceWith(newButton);
160. // Render move
161. document.getElementById("move").textContent = `Move: ${move}`;
163. // Render time
164. document.getElementById("time").textContent = `Time: ${time}`;
166. // Render message
167. **if** (status === "won") {
168. document.querySelector(".message").textContent = "You win!";
169. } **else** {
170. document.querySelector(".message").textContent = "";
171. }
172. }
173. }
175. **const** GAME = Game.ready();

The HTML 5 code below shows how to complete the code for this awesome puzzle game!

1. <!doctype html>
2. <html>
3. <head>
4. <title>Puzzle game</title>
5. <link rel="stylesheet" type="text/css" href="style.css">
6. </head>
8. <body>
10. <h1>PRABAKARAN ACT</h1>
11. <div class="game">
12. <div class="grid">
13. <button>1</button>
14. <button>2</button>
15. <button>3</button>
16. <button>4</button>
17. <button>5</button>
18. <button>6</button>
19. <button>7</button>
20. <button>8</button>
21. <button>9</button>
22. <button>10</button>
23. <button>11</button>
24. <button>12</button>
25. <button>13</button>
26. <button>14</button>
27. <button>15</button>
28. <button></button>
29. </div>
31. <div class="footer">
32. <button>Play</button>
33. <span id="move">Move: 100</span>
34. <span id="time">Time: 100</span>
35. </div>
36. </div>
37. <h1 class="message">You win!</h1>

40. <script src="script.js"></script>
41. </body>
42. </html>