```
1
       /* jshint esversion: 6 */
                                                                                                                    CONFIGURE
2
       class MazeBuilder {
3
 4
            // Original JavaScript code by Chirp Internet: www.chirpinternet.eu
            // Please acknowledge use of this code by including this header.
// taken from https://www.the-art-of-web.com/javascript/playable-maze-generator/ and a There are 16 function
// original base class can be found at https://www.the-art-of-web.com/maze-builder.js Function with the large
5
6
7
            constructor(width, height) {
8
                                                                                                             About (Median) is 1.
9
                 this.width = width;
10
                 this.height = height;
                                                                                                             Documentation (/docs) has
11
                 this.cols = 2 * this.width + 1;
                                                                                                                    3.5.
                 this.rows = 2 * this.height + 1;
12
                                                                                                             Install (/install)
The most complex fu
13
                 this.maze = this.initArray([]);
14
                                                                                                             Contribute 3/contribute media
                 /* place initial walls */
15
                                                                                                             Blog (/blog)
16
                 this.maze.forEach((row, r) => {
17
                     row.forEach((cell, c) => {
18
                          switch (r) {
19
                               case 0:
20
                               case this.rows - 1:
                                                                                                                    One unused varia
21
                                   this.maze[r][c] = ["wall"];
22
                                   break;
                                                                                                                    3 MazeBuilder
23
24
                               default:
                                   if ((r % 2) == 1) {
    if ((c == 0) || (c == this.cols - 1)) {
25
26
27
                                            this.maze[r][c] = ["wall"];
28
                                   } else if (c % 2 == 0) {
29
30
                                        this.maze[r][c] = ["wall"];
31
32
                          }
                     });
33
34
35
                     if (r == 0) {
36
                          /st place exit in top row st/
37
                          let doorPos = this.posToSpace(this.rand(1, this.width));
38
                          this.maze[r][doorPos] = ["door", "exit"];
39
                     } else if (r == this.rows - 1) {
40
                          /* place entrance in bottom row */
41
                          let doorPos = this.posToSpace(this.rand(1, this.width));
42
                          this.maze[r][doorPos] = ["door", "entrance"];
43
                     } else {
44
                          /* place treat in row */
45
                          let treatPos = this.posToSpace(this.rand(1, this.width));
46
                          this.maze[r][treatPos] = ["treat"];
47
                     }
48
                 });
49
50
                 /* start partitioning */
51
                 this.partition(1, this.height - 1, 1, this.width - 1);
52
53
54
            initArray(value) {
55
                 return new Array(this.rows).fill().map(() => new Array(this.cols).fill(value));
56
57
58
            rand(min, max) {
                 return min + Math.floor(Math.random() * (1 + max - min));
59
60
61
62
            posToSpace(x) {
                 return 2 * (x - 1) + 1;
63
64
65
            posToWall(x) {
66
                 return 2 * x;
67
68
69
            inBounds(r, c) {
70
```

https://jshint.com

```
if ((typeof this.maze[r] == "undefined") || (typeof this.maze[r][c] == "undefined"
71
72
                    return false; /* out of bounds */
73
               return true;
74
75
           }
76
                                                                                                             version 2,13.6
There are 16 function
77
           shuffle(array) {
78
               /* source: https://stackoverflow.com/a/12646864 */
                                                                                                     (https://github.com/jshint/j:
Function with the larg
               for (let i = array.length - 1; i > 0; i--) {
79
                    const j = Math.floor(Math.random() * (i + 1));
80
                                                                                                      About (Mandian) is 1.
81
                    [array[i], array[j]] = [array[j], array[i]];
82
                                                                                                      Documentation (/docs) has
83
               return array;
                                                                                                             3.5.
                                                                                                      Install (/install)
The most complex fu
84
           }
85
                                                                                                      Contribute 3/contribute media
86
           partition(r1, r2, c1, c2) {
               /* create partition walls
87
                   ref: https://en.wikipedia.org/wiki/Maze_generation_algorithm#Recursive_divis Bong (/blog)
88
89
               let horiz, vert, x, y, start, end;
90
91
               if ((r2 < r1) || (c2 < c1)) {
92
                    return false;
                                                                                                             One unused varia
93
94
                                                                                                             3 MazeBuilder
95
               if (r1 == r2) {
96
                    horiz = r1;
97
               } else {
98
                    x = r1 + 1;
                    y = r2 - 1;
99
                    start = Math.round(x + (y - x) / 4);
end = Math.round(x + 3 * (y - x) / 4);
100
101
102
                    horiz = this.rand(start, end);
               }
103
104
105
               if (c1 == c2) {
                    vert = c1;
106
107
               } else {
108
                    x = c1 + 1;
109
                    y = c2 - 1;
110
                    start = Math.round(x + (y - x) / 3);
                    end = Math.round(x + 2 * (y - x) / 3);
111
112
                    vert = this.rand(start, end);
               }
113
114
               for (let i = this.posToWall(r1) - 1; i \leftarrow this.posToWall(r2) + 1; i++) {
115
116
                    for (let j = this.posToWall(c1) - 1; j <= this.posToWall(c2) + 1; j++) {
                        if ((i == this.posToWall(horiz)) || (j == this.posToWall(vert))) {
117
118
                            this.maze[i][j] = ["wall"];
119
                    }
120
121
               }
122
123
               let gaps = this.shuffle([true, true, true, false]);
124
               /st create gaps in partition walls st/
125
               if (gaps[0]) {
126
127
                    let gapPosition = this.rand(c1, vert);
                    this.maze[this.posToWall(horiz)][this.posToSpace(gapPosition)] = [];
128
129
130
               if (gaps[1]) {
131
132
                    let gapPosition = this.rand(vert + 1, c2 + 1);
133
                    this.maze[this.posToWall(horiz)][this.posToSpace(gapPosition)] = [];
134
               }
135
136
               if (gaps[2]) {
                    let gapPosition = this.rand(r1, horiz);
137
                    this.maze[this.posToSpace(gapPosition)][this.posToWall(vert)] = [];
138
139
140
141
               if (gaps[3]) {
                    let gapPosition = this.rand(horiz + 1, r2 + 1);
142
```

https://jshint.com

```
this.maze[this.posToSpace(gapPosition)][this.posToWall(vert)] = [];
143
144
               }
145
               /* recursively partition newly created chambers */
146
               this.partition(r1, horiz - 1, c1, vert - 1);
147
148
               this.partition(horiz + 1, r2, c1, vert - 1);
               this.partition(r1, horiz - 1, vert + 1, c2);
149
150
               this.partition(horiz + 1, r2, vert + 1, c2);
151
          }
152
          isGap(...cells) {
153
154
               return cells.every((array) => {
155
                   let row, col;
156
                   [row, col] = array;
157
                   if (this.maze[row][col].length > 0) {
                       if (!this.maze[row][col].includes("door")) {
158
159
                           return false;
160
161
162
                   return true;
163
               });
          }
164
165
166
          display(id) {
167
               this.parentDiv = document.getElementById(id);
168
169
               if (!this.parentDiv) {
170
                   alert("Cannot initialize maze - no element found with id \"" + id + "\"");
171
                   return false;
172
               }
173
               while (this.parentDiv.firstChild) {
174
                   this.parentDiv.removeChild(this.parentDiv.firstChild);
175
176
177
178
               const container = document.createElement("div");
179
               container.id = "maze";
180
181
               this.maze.forEach((row) => {
182
                   let rowDiv = document.createElement("div");
                   row.forEach((cell) => {
183
184
                       let cellDiv = document.createElement("div");
                       if (cell) {
185
186
                           cellDiv.className = cell.join(" ");
187
                       rowDiv.appendChild(cellDiv);
188
189
                   });
190
                   container.appendChild(rowDiv);
191
192
               this.parentDiv.appendChild(container);
193
194
               return true;
195
          }
196
      }
```



version 2,13.6 There are 16 functior (https://github.com/jshint/j: Function with the larg

About (Mandian) is 1.

Documentation (/docs)
3.5.
Install (/install)
The most complex fu

Contribute 3/White the hedia Blog (/blog)

One unused varia

3 MazeBuilder

https://jshint.com 3/3