```
/* jshint esversion: 6 */

class MazeBuilder {
    // 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
    // original base class can be found at https://www.the-art-of-web.com/maze-builder.js
    constructor(width, height) {
        this.width = width;
        this.height = height;
        this.cols = 2 * this.width + 1;
        this.rows = 2 * this.height + 1;
        this.maze = this.initArray([]);

        /* place initial walls */
        this.maze.forEach((row, r) => {
            row.forEach((cell, c) => {
                switch (r) {
                    case 0:
                    case this.rows - 1:
                        this.maze[r][c] = ["wall"];
                        break;

                    default:
                        if ((r % 2) == 1) {
                            if ((c == 0) || (c == this.cols - 1)) {
                                this.maze[r][c] = ["wall"];
                            }
                        } else if (c % 2 == 0) {
                            this.maze[r][c] = ["wall"];
                        }
                }
            });

            if (r == 0) {
                /* place exit in top row */
                let doorPos = this.posToSpace(this.rand(1, this.width));
                this.maze[r][doorPos] = ["door", "exit"];
            } else if (r == this.rows - 1) {
                /* place entrance in bottom row */
                let doorPos = this.posToSpace(this.rand(1, this.width));
                this.maze[r][doorPos] = ["door", "entrance"];
            } else {
                /* place treat in row */
                let treatPos = this.posToSpace(this.rand(1, this.width));
                this.maze[r][treatPos] = ["treat"];
            }
        });

        /* start partitioning */
        this.partition(1, this.height - 1, 1, this.width - 1);
    }

    initArray(value) {
        return new Array(this.rows).fill().map(() => new Array(this.cols).fill(value));
    }

    rand(min, max) {
        return min + Math.floor(Math.random() * (1 + max - min));
    }

    posToSpace(x) {
        return 2 * (x - 1) + 1;
    }

    posToWall(x) {
        return 2 * x;
    }

    inBounds(r, c) {
```

CONFIGURE

version 2.13.6
(https://github.com/jshint/js

About (/about)
Documentation (/docs)
Install (/install)
Contribute (/contribute)
Blog (/blog)

Metrics

There are 16 function
Function with the larg
median is 1.

Largest function has
3.5.

The most complex fu
of 13 while the media

One unused varia

3 MazeBuilder

```
            if ((typeof this.maze[r] == "undefined") || (typeof this.maze[r][c] == "undefined"
                return false; /* out of bounds */
            }
            return true;
        }

        shuffle(array) {
            /* source: https://stackoverflow.com/a/12646864 */
            for (let i = array.length - 1; i > 0; i--) {
                const j = Math.floor(Math.random() * (i + 1));
                [array[i], array[j]] = [array[j], array[i]];
            }
            return array;
        }

        partition(r1, r2, c1, c2) {
            /* create partition walls
               ref: https://en.wikipedia.org/wiki/Maze_generation_algorithm#Recursive_division
            let horiz, vert, x, y, start, end;

            if ((r2 < r1) || (c2 < c1)) {
                return false;
            }

            if (r1 == r2) {
                horiz = r1;
            } else {
                x = r1 + 1;
                y = r2 - 1;
                start = Math.round(x + (y - x) / 4);
                end = Math.round(x + 3 * (y - x) / 4);
                horiz = this.rand(start, end);
            }

            if (c1 == c2) {
                vert = c1;
            } else {
                x = c1 + 1;
                y = c2 - 1;
                start = Math.round(x + (y - x) / 3);
                end = Math.round(x + 2 * (y - x) / 3);
                vert = this.rand(start, end);
            }

            for (let i = this.posToWall(r1) - 1; i <= this.posToWall(r2) + 1; i++) {
                for (let j = this.posToWall(c1) - 1; j <= this.posToWall(c2) + 1; j++) {
                    if ((i == this.posToWall(horiz)) || (j == this.posToWall(vert))) {
                        this.maze[i][j] = ["wall"];
                    }
                }
            }

            let gaps = this.shuffle([true, true, true, false]);

            /* create gaps in partition walls */
            if (gaps[0]) {
                let gapPosition = this.rand(c1, vert);
                this.maze[this.posToWall(horiz)][this.posToSpace(gapPosition)] = [];
            }

            if (gaps[1]) {
                let gapPosition = this.rand(vert + 1, c2 + 1);
                this.maze[this.posToWall(horiz)][this.posToSpace(gapPosition)] = [];
            }

            if (gaps[2]) {
                let gapPosition = this.rand(r1, horiz);
                this.maze[this.posToSpace(gapPosition)][this.posToWall(vert)] = [];
            }

            if (gaps[3]) {
                let gapPosition = this.rand(horiz + 1, r2 + 1);
```

JSHint
version 2.13.6
(https://github.com/jshint/js

About (/about)
Documentation (/docs)
Install (/install)
Contribute (/contribute)
Blog (/blog)

Metrics

There are 16 function
Function with the larg
median is 1.

Largest function has
3.5.

The most complex fu
of 13 while the media

One unused varia

3 MazeBuilder

```
            this.maze[this.posToSpace(gapPosition)][this.posToWall(vert)] = [];
        }

        /* recursively partition newly created chambers */
        this.partition(r1, horiz - 1, c1, vert - 1);
        this.partition(horiz + 1, r2, c1, vert - 1);
        this.partition(r1, horiz - 1, vert + 1, c2);
        this.partition(horiz + 1, r2, vert + 1, c2);
    }

    isGap(...cells) {
        return cells.every((array) => {
            let row, col;
            [row, col] = array;
            if (this.maze[row][col].length > 0) {
                if (!this.maze[row][col].includes("door")) {
                    return false;
                }
            }
            return true;
        });
    }

    display(id) {
        this.parentDiv = document.getElementById(id);

        if (!this.parentDiv) {
            alert("Cannot initialize maze - no element found with id \"" + id + "\"");
            return false;
        }

        while (this.parentDiv.firstChild) {
            this.parentDiv.removeChild(this.parentDiv.firstChild);
        }

        const container = document.createElement("div");
        container.id = "maze";

        this.maze.forEach((row) => {
            let rowDiv = document.createElement("div");
            row.forEach((cell) => {
                let cellDiv = document.createElement("div");
                if (cell) {
                    cellDiv.className = cell.join(" ");
                }
                rowDiv.appendChild(cellDiv);
            });
            container.appendChild(rowDiv);
        });

        this.parentDiv.appendChild(container);
        return true;
    }
}
```

JSHint version 2.13.6
(https://github.com/jshint/js

Metrics

There are 16 function
Function with the larg
median is 1.

Largest function has
3.5.

The most complex fu
of 13 while the media

One unused varia

3 MazeBuilder