JavaScript Number of Islands

Challenge

Given an `m x n` 2D binary grid `grid` which represents a map of `'1'`s (land) and `'0'`s (water), return the number of islands.

An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

1st Example

```
Input: grid = [
['1','1','1','1','0'],
['1','1','0','1','0'],
['1','1','0','0','0'],
['0','0','0','0','0']
]
Output: 1
```

2nd Example

```
Input: grid = [
['1','1','0','0','0'],
['1','1','0','0','0'],
['0','0','1','0','0'],
['0','0','0','1','1']
]
Output: 3
```

Constraints

- `m == grid.length`
- `n == grid[i].length`
- `1 <= m, n <= 300`
- `grid[i][j]` is `'0'` or `'1'`.

Solution

```
const numIslands = (grid) => {
    let count = 0;

    for (let i = 0; i < grid.length; i++) {
        for (let j = 0; j < grid[0].length; j++) {
            if (grid[i][j] === '1') {
                count++;
                dfs(grid, i,j);
            }
        }
    }

    return count;
};
```

Solution continues on next page...

```
const dfs = (grid, row, col) => {
    if (row < 0 ||
        row >= grid.length ||
        col < 0 ||
        col >= grid[0].length) {
        return;
    }

    const value = grid[row][col];

    if (value === '1') {
        grid[row][col] = '#';
        dfs(grid, row + 1, col);
        dfs(grid, row - 1, col);
        dfs(grid, row, col + 1);
        dfs(grid, row, col - 1);
    }
};
```

Explanation

I've defined two functions: `numIslands` and `dfs`.

The `numIslands` function takes a grid as input and returns the number of islands present in the grid. It initializes a variable `count` to `0`.

Inside the function, there is a nested loop that iterates over each cell in the grid. If the current cell has a value of `'1'`, indicating it is part of an island, the `count` is incremented and the `dfs` function is called with the current grid and the indices of the current cell.

The `dfs` function is a recursive function that performs a depth-first search on the grid. It takes the grid, row index, and column index as input.

Within the `dfs` function, there is an initial check to see if the current row or column indices are out of bounds of the grid. If they are, the function returns.

Next, the value of the current cell is retrieved from the grid.

If the value is `'1'`, indicating an unvisited cell that is part of an island, the value of the cell is changed to `'#'` to mark it as visited. Then, the `dfs` function is called recursively for the adjacent cells: one cell below, one cell above, one cell to the right, and one cell to the left.

This recursive exploration continues until all connected cells of the current island are visited.

After the DFS traversal is complete, the `numIslands` function returns the final count of islands.

In summary, I've utilized a depth-first search algorithm to traverse the grid and count the number of islands. The `numIslands` function iterates through each cell, marking visited cells and exploring adjacent cells using the `dfs` function. The `dfs` function recursively explores connected cells until all islands are visited. The final count of islands is returned as the output of the `numIslands` function.

Author: Trevor Morin