JavaScript Island Perimeter

Challenge

You are given `row x col` `grid` representing a map where `grid[i][j] = 1` represents land and `grid[i][j] = 0` represents water.

Grid cells are connected horizontally/vertically (not diagonally). The `grid` is completely surrounded by water, and there is exactly one island (ex. one or more connected land cells).

The island doesn't have "lakes", meaning the water inside isn't connected to the water around the island. One cell is a square with side length 1. The grid is rectangular, width and height don't exceed 100. Determine the perimeter of the island.

1st Example

```
Input: grid = [[0,1,0,0],
               [1,1,1,0],
               [0,1,0,0],
               [1,1,0,0]]
Output: 16
Explanation: The perimeter is the 16 yellow stripes in the
             image above.
```

2nd Example

```
Input: grid = [[1]]
Output: 4
```

3rd Example

```
Input: grid = [[1,0]]
Output: 4
```

Constraints

- `row == grid.length`
- `col == grid[i].length`
- `1 <= row, col <= 100`
- `grid[i][j]` is `0` or `1`.
- There is exactly one island in `grid`.

Solution

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

    const lookLeft = (i, j) => {
        if (j === 0) {
            return true;
        }

        return !grid[i][j - 1];
    };

    const lookUp = (i, j) => {
        if (i === 0) {
            return true;
        }
```

Solution continues on next page...

```
        return !grid[i - 1][j];
    };

    const lookRight = (i, j) => {
        if (j === grid[i].length - 1) {
            return true;
        }

        return !grid[i][j + 1];
    };

    const lookBottom = (i, j) => {
        if (i === grid.length - 1) {
            return true;
        }

        return !grid[i + 1][j];
    };

    for (let i = 0; i < grid.length; i++) {
        for (let j = 0; j < grid[i].length; j++) {
            if (grid[i][j]) {
                lookLeft(i, j) && count++;

                lookUp(i, j) && count++;

                lookRight(i, j) && count++;

                lookBottom(i, j) && count++;
            }
        }
    }

    return count;
};
```

Explanation

I've defined a function called `islandPerimeter` that takes a grid as input. It's purpose is to calculate the perimeter of an island represented by the grid.

The function starts by initializing a variable called `count` to `0`, which will be used to keep track of the perimeter.

Next, the function defines four helper functions: `lookLeft`, `lookUp`, `lookRight`, and `lookBottom`. These functions are used to check the neighboring cells of a given cell in the grid.

The `lookLeft` function checks if the current cell is at the leftmost edge of the grid. If it is, it returns true; otherwise, it checks if the cell to the left is water (represented by a false value in the grid). Similarly, the `lookUp`, `lookRight`, and `lookBottom` functions perform similar checks for the cells above, to the right, and below the current cell, respectively.

The function then enters a nested loop to iterate over each cell in the grid. For each cell, it checks if it represents land (a true value in the grid).

If the cell is land, the function calls the four helper functions to check the neighboring cells in each direction. If any of the neighboring cells are water, it increments the count variable by `1` to keep track of the perimeter.

Finally, the function returns the calculated perimeter count.

In summary, the `islandPerimeter` function calculates the perimeter of an island by examining the neighboring cells of each land cell and incrementing the count if any neighboring cell is water.