JavaScript Sum of Left Leaves

Challenge

Given the `root` of a binary tree, return the sum of all left leaves.

A leaf is a node with no children. A left leaf is a leaf that is the left child of another node.

1st Example

```
Input: root = [3,9,20,null,null,15,7]
Output: 24
Explanation: There are two left leaves in the binary tree,
             with values 9 and 15 respectively.
```

2nd Example

```
Input: root = [1]
Output: 0
```

Constraints

- `-1000 <= Node.val <= 1000`
- The number of nodes in the tree is in the range `[1, 1000]`.

Solution

```
const sumOfLeftLeaves = (root) => {
    if (!root) return 0;

    const {left, right} = root;

    let [sumLeft, sumRight] = [sumOfLeftLeaves(left),
                               sumOfLeftLeaves(right)];

    if (!sumLeft   &&
        left       &&
        !left.left &&
        !left.right) sumLeft = left.val;

    return Number(sumLeft) + Number(sumRight);
};
```

Explanation

I've written a function called `sumOfLeftLeaves` that calculates the sum of the values of the left leaves in a binary tree.

If the root node is null, indicating an empty tree, the function immediately returns `0`.

If the root node is not null, the function uses destructuring assignment to extract the `left` and `right` children of the root node.

The function then recursively calls itself on the `left` and `right`

children, storing the returned values in the variables `sumLeft` and `sumRight` respectively.

Next, it checks if `sumLeft` is `0`, which means there are no left leaves encountered yet. It also checks if the `left` child exists and is a leaf node, meaning it has no left or right child. If these conditions are met, the value of the leaf node is assigned to `sumLeft`.

Finally, the function returns the sum of `sumLeft` and `sumRight`, after converting them to numbers using the `Number` function.

In summary, this function recursively calculates the sum of the values of the left leaves in a binary tree. It traverses the tree, keeping track of the sum of the left leaves encountered so far, and returns the final sum.

Author: Trevor Morin