Print Tree with Stars

how to print tree with star char in js

Here's how to print a tree shape using star characters (`*`) in JavaScript. This example creates a tree with a specified number of levels.

```javascript
function printTree(levels) {
  for (let i = 1; i <= levels; i++) {
    const spaces = ' '.repeat(levels - i);        // Add leading spaces for
alignment
    const stars = '*'.repeat(2 * i - 1);          // Create stars for current level
    console.log(spaces + stars + spaces);         // Print each level with stars
centered
  }
}

printTree(5);  // Change the number to make a taller or shorter tree
```

Explanation:

- `spaces` aligns each row of stars to the center.
- `stars` generates the appropriate number of `*` characters for each row.

For example, if `levels` is 5, the output would be:

```markdown
    *
   ***
  *****
 *******
*********
```

You can adjust `levels` to make the tree taller or shorter.

ere's how to print a tree shape using star characters (*) in JavaScript multi solution

Certainly! Here are multiple ways to print a star tree in JavaScript, each with a unique approach:

1. Basic `for` Loop Solution

This approach uses two loops: one for each level and another to build each row with spaces and stars.

```javascript
function printTree(levels) {
  for (let i = 1; i <= levels; i++) {
    let spaces = ' '.repeat(levels - i);
    let stars = '*'.repeat(2 * i - 1);
    console.log(spaces + stars + spaces);
  }
}

printTree(5);
```

2. Using Recursion

This recursive approach uses a helper function to print each row until it reaches the desired number of levels.

```javascript
function printTreeRecursive(levels, currentLevel = 1) {
  if (currentLevel > levels) return;

  let spaces = ' '.repeat(levels - currentLevel);
  let stars = '*'.repeat(2 * currentLevel - 1);
  console.log(spaces + stars + spaces);

  printTreeRecursive(levels, currentLevel + 1);
}

printTreeRecursive(5);
```

3. Using Array `map` and `fill`

This functional approach uses an array with `map` and `fill` to construct each level in the tree.

```javascript
function printTreeArray(levels) {
  Array(levels).fill('').map((_, i) => {
    const spaces = ' '.repeat(levels - i - 1);
    const stars = '*'.repeat(2 * i + 1);
    console.log(spaces + stars + spaces);
  });
}

printTreeArray(5);
```

4. Building the Tree in a String (for Logging Once)

This version builds the entire tree as a single string and logs it at the end.

```javascript
function printTreeAsString(levels) {
  let tree = '';
  for (let i = 1; i <= levels; i++) {
    let spaces = ' '.repeat(levels - i);
    let stars = '*'.repeat(2 * i - 1);
    tree += spaces + stars + spaces + '\n';
  }
  console.log(tree);
}

printTreeAsString(5);
```

5. Using a While Loop

Here's a version that uses a `while` loop instead of a `for` loop.

```javascript
function printTreeWhile(levels) {
  let i = 1;
  while (i <= levels) {
    let spaces = ' '.repeat(levels - i);
    let stars = '*'.repeat(2 * i - 1);
    console.log(spaces + stars + spaces);
    i++;
  }
}

printTreeWhile(5);
```

Each of these methods achieves the same output but uses a different technique. You can pick one based on your preference or performance needs.