Flatten Array

This problem will allow us to perform a task that has practical uses in a developer's career. We'll discuss how to flatten deeply nested arrays and why a recursive solution is a good way to attack this problem.

Flatten Array

Instructions

Write a function that will take an array of deeply nested arrays and extract every item, flattening the array. It should return a new array that contains the items of each internal array, preserving order.

Input: Array

Output: Array

Examples:

```
flatten([ [ [ [1], 2], 3], [4], [], [[5]]]);
// -> [1, 2, 3, 4, 5]

flatten(['abc', ['def', ['ghi', ['jkl']]]]);
// -> ['abc', 'def', 'ghi', 'jkl']
```

Hints

- As in the last problem, we have to process every item we receive. There's no way to get around that so the best time complexity we can hope for is `O(n)`.

```
function flatten(nestedArray) {
    // Your code here
}
```

Solution

This problem is easier to discuss step-by-step. First, we'll figure out how to flatten arrays only a single level deep.

This means nesting is only 1 level deep. We're aiming to flatten arrays like this:

```
[[1, 2, 3], 4, 5, [6], [7], [8, 9], 10]
```

To do this, we'll create a new, empty array. We'll then process our input array and check each item in a loop.

In the loop, we can check the type of each item. If it's not an array, we can push it into our new array.

If it's an array, we need to then run another loop on that array.

Partial Solution

```
function flatten(nestedArray) {
    const newArray = [];

    for(let i = 0; i < nestedArray.length; i++) {
        const thisItem = nestedArray[i];

         if(Array.isArray(thisItem)) {
             for(let j = 0; j < thisItem.length; j++) {
                 newArray.push(thisItem[j]);
             }
11            } else {
12                newArray.push(thisItem);
13            }
14        }
15
16        return newArray;
17    }
```

This function flattens arrays only one level deep. We go through our input array one by one and perform a check on each item.

If the item we're currently processing is not an array (line 11), we push it into our new array.

If it *is* an array (lines 7 - 10), we push each individual item into our new array.

Repeating the Strategy

Lines 8 - 10 are the key. To be able to flatten arrays that are indefinitely nested, we'll need to repeat this process. We'll need a recursive function.

When we encounter an array, we need to run our function on that array.

Full Solution

```
1   function flatten(nestedArray) {
2       const newArray = [];
3
4       for(let i = 0; i < nestedArray.length; i++) {
5           const thisItem = nestedArray[i];
6
7            if(Array.isArray(thisItem)) {
8                const flatItem = flatten(thisItem);
9
10               for(let j = 0; j < flatItem.length; j++) {
```

```
10            for(let j = 0; j < flatItem.length; j++) {
11                newArray.push(flatItem[j]);
12            }
13        } else {
14            newArray.push(thisItem);
15        }
16    }
17
18    return newArray;
19 }
```

Notice that the only difference between the two code blocks is line 8. Rather than passing each value inside `thisItem` to `newArray`, we first run `thisItem` through the `flatten` function.

Time Complexity

Line 10 may make this function seem like it's `O(n^2)` - we have a for-loop inside a for-loop, which usually results in `O(n^2)`. However, the inner loop only processes what the outer loop skips over.

(/learn)

For example, take the array `[1, 2, [3, 4, 5], 6, 7]`. The outer array only has 5 elements in it:

- `1`
- `2`
- `[3, 4, 5]`
- `6`
- `7`

This means the outer `forEach` loop will essentially skip over `3`, `4`, and `5`, leaving the inner for-loop to deal with those.

The final time complexity is:

O(n)

because the inner and outer loops operate on different items. No item is processed twice.

Space Complexity

Space complexity is also

O(n).

Every item is stored in the brand new array.

Conclusion

Recursion is a powerful tool that some problems require. If a process needs to be repeated over and over, sometimes recursion is the only tool that can solve it. We'll use it again in future problems.

← Back
Is Unique

Next →
Remove Dupes

✓ Mark as Completed

Report an Issue