JavaScript Two Sum

Challenge

Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

1st Example

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9,
             we return [0, 1].
```

2nd Example

```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```

3rd Example

```
Input: nums = [3,3], target = 6
Output: [0,1]
```

Constraints

- `2 <= nums.length <= 10⁴`
- `-10⁹ <= nums[i] <= 10⁹`
- `-10⁹ <= target <= 10⁹`
- Only one valid answer exists.

Solution

```
const twoSum = (nums, target) => {
    let numsObject = {};

    for (let i = 0; i < nums.length; i++) {
        let n          = nums[i],
            compliment = target-n;

        if (numsObject.hasOwnProperty(compliment)) {
            return [i,numsObject[compliment]];
        }

        numsObject[n] = i;
    }

    return [-1, -1];
};
```

Explanation

I've built a function called `twoSum` that takes in an array of numbers (`nums`) and a target number. The purpose of this function is to find a pair of numbers in the array that add up to the target and return their indices in an array. If no such pair is found, it returns `[-1, -1]`.

Inside the function, an empty object called `numsObject` is initialized.

A `for` loop is used to iterate through each element in the `nums` array. Within the loop, the current element is assigned to a variable `n`, and the difference between the target and `n` is calculated and assigned to a variable `compliment`.

An `if` statement checks if the `numsObject` object has a property with the value of `compliment`. If such a property exists, it means that a pair of numbers that add up to the target has been found.

In this case, the function returns an array containing the current index (`i`) and the index of the compliment in the `numsObject` object.

If the compliment property does not exist, it means that the current number has not been encountered before. In that case, the current number (`n`) is added as a property to the `numsObject` object, with the value of the current index (`i`).

After the loop finishes, if no pair of numbers that add up to the target is found, the function returns `[-1, -1]`.

In summary, this function searches for a pair of numbers in the input array that add up to the target. It uses an object to store previously encountered numbers and their indices. The function iterates through the array, checking if the compliment of each number exists in the object. If a pair is found, their indices are returned. If no pair is found, `[-1, -1]` is returned.