JavaScript Word Pattern

Challenge

Given a `pattern` and a string `s`, find if `s` follows the same pattern.

Here follow means a full match, such that there is a bijection between a letter in `pattern` and a non-empty word in `s`.

1st Example

```
Input: pattern = 'abba', s = 'dog cat cat dog'
Output: true
```

2nd Example

```
Input: pattern = 'abba', s = 'dog cat cat fish'
Output: false
```

3rd Example

```
Input: pattern = 'aaaa', s = 'dog cat cat dog'
Output: false
```

Constraints

- `1 <= pattern.length <= 300`
- `1 <= s.length <= 3000`
- `pattern` contains only lower-case English letters.
- `s` contains only lowercase English letters and spaces `' '`.
- `s` does not contain any leading or trailing spaces.
- All the words in `s` are separated by a single space.

Solution

```
const wordPattern = (pattern, s) => {
    const arr = s.split(' '),
          map = {};

    if (arr.length !== pattern.length ||
        new Set([...pattern]).size !== new Set(arr).size) {
        return false;
    }

    for (let i = 0; i < pattern.length; i++) {
        if (!map[pattern[i]]) {
            map[pattern[i]] = arr[i];
        } else if (map[pattern[i]] !== arr[i]) {
            return false;
        }
    }

    return true;
};
```

Explanation

I've built a function called `wordPattern` that takes two parameters: `pattern` and `s` (string). Its purpose is to check if the pattern matches the string by mapping each character in the pattern to a word in the string.

Inside the function, the string `s` is split into an array of words using the `split()` method and stored in the variable `arr`. An empty object called `map` is also created.

The function then checks if the length of the `arr` array is not equal to the length of the `pattern` or if the number of unique characters in the `pattern` is not equal to the number of unique words in `arr`. If either of these conditions is true, the function immediately returns `false`.

Next, a `for` loop is used to iterate through each character in the `pattern`. Inside the loop, it checks if the current character is already a key in the `map` object. If it is not, the character is added as a key in the `map` object, and the corresponding word in `arr` is added as the value.

If the current character is already a key in the `map` object, it checks if the value of the key matches the corresponding word in `arr`. If the values do not match, the function immediately returns `false`.

If the `for` loop completes without returning `false`, it means that the pattern matches the string, and the function returns `true`.

In summary, the `wordPattern` function checks if a given pattern matches a given string by mapping each character in the pattern to a word in the string. It returns `true` if the mapping is successful, and `false` otherwise.