JavaScript Happy Number

Challenge

Write an algorithm to determine if a number `n` is happy.

A happy number is a number defined by the following process:

- Starting with any positive integer, replace the number by the sum of the squares of its digits.
- Repeat the process until the number equals `1` (where it will stay), or it loops endlessly in a cycle which does not include `1`.
- Those numbers for which this process ends in `1` are happy.

Return `true` if `n` is a happy number, and `false` if not.

1st Example

```
Input: n = 19
Output: true
Explanation: 1² + 9² = 82
             8² + 2² = 68
             6² + 8² = 100
             1² + 0² + 0² = 1
```

2nd Example

```
Input: n = 2
Output: false
```

Constraints

- `1 <= n <= 2³¹ - 1`

Solution

```
const isHappy = (n) => {
    const hashMap = {};

    const recursion = (number) => {
        const array = number.toString().split('');
        let newNumber = 0;

        for (let i = 0; i < array.length; i++) {
            newNumber += Number(array[i])**2;
        }

        if (newNumber === 1) {
            return true;
        }

        if (hashMap[newNumber]) {
            return false;
        }

        hashMap[newNumber] = newNumber;

        return recursion(newNumber);
    };

    return recursion(n);
};
```

Explanation

I've written a function called `isHappy` that takes in a number `n` as a parameter. Its purpose is to determine if the number is a `happy number` using recursion.

Inside the function, an empty object called `hashMap` is created to keep track of numbers encountered during the recursion.

The function defines a nested function called `recursion` that takes in a number as a parameter.

Within the `recursion` function, the number is converted to a string and split into an array of individual digits.

A variable called `newNumber` is initialized to `0`.

A for loop iterates through each digit in the array. For each digit, it is squared using the exponentiation operator (`**`), and the result is added to `newNumber`.

After the loop, there are three conditional statements:

If `newNumber` is equal to `1`, it means that the number is a happy number, so the function returns true.

If `newNumber` already exists as a key in the `hashMap` object, it means that the recursion has entered a cycle, and the number is not a happy number. In this case, the function returns false.

If neither of the above conditions are met, the `newNumber` is added as a key to the `hashMap` object, and the recursion continues by calling the `recursion` function with `newNumber` as the argument.

Finally, the function returns the result of the initial call to the `recursion` function with the input number `n`.

In summary, the `isHappy` function determines if a number is a `happy number` by recursively calculating the sum of the squares of its digits. It uses a hashMap object to keep track of encountered numbers and returns true if the recursion reaches `1`, or false if it enters a cycle.