JavaScript Design HashMap

Challenge

Design a HashMap without using any built-in hash table libraries.

Implement the `MyHashMap` class:

- `MyHashMap()` initializes the object with an empty map.
- `void put(int key, int value)` inserts a `(key, value)` pair into the HashMap. If the `key` already exists in the map, update the corresponding `value`.
- `int get(int key)` returns the `value` to which the specified `key` is mapped, or `-1` if this map contains no mapping for the `key`.
- `void remove(key)` removes the `key` and its corresponding `value` if the map contains the mapping for the `key`.

Example

```
Input: [
           'MyHashMap', 'put', 'put', 'get',
           'get', 'put', 'get', 'remove', 'get'
       ]
       [
           [], [1, 1], [2, 2], [1], [3],
           [2, 1], [2], [2], [2]
       ]
```

Example continues on next page...

```
Output: [null, null, null, 1, -1, null, 1, null, -1]
Explanation: MyHashMap myHashMap = new MyHashMap();
myHashMap.put(1, 1); // The map is now [[1,1]]
myHashMap.put(2, 2); // The map is now [[1,1], [2,2]]
myHashMap.get(1);    /* return 1,
                        The map is now [[1,1], [2,2]] */
myHashMap.get(3);    /* return -1 (ex. not found),
                        The map is now [[1,1], [2,2]] */
myHashMap.put(2, 1); /* The map is now [[1,1], [2,1]]
                        (ex. update the existing value) */
myHashMap.get(2);    /* return 1,
                        The map is now [[1,1], [2,1]] */
myHashMap.remove(2); /* remove the mapping for 2,
                        The map is now [[1,1]] */
myHashMap.get(2);    /* return -1 (ex. not found),
                        The map is now [[1,1]] */
```

Constraints

- `0 <= key, value <= 10^6`
- At most `10^4` calls will be made to `put`, `get`, and `remove`.

Solution

```
let MyHashMap = function() {
    this.map = {};
};

MyHashMap.prototype.put = function(key, value) {
    this.map[key] = value;
};
```

Solution continues on next page...

```
MyHashMap.prototype.get = function(key) {
    return this.map[key] != undefined ? this.map[key] : -1;
};

MyHashMap.prototype.remove = function(key) {
    delete this.map[key];
};
```

Explanation

I've built a class called `MyHashMap` which is used to create a hash map data structure. The hash map stores key-value pairs, where keys are unique identifiers and values are associated data.

The `MyHashMap` class is defined using the constructor function. The constructor initializes an empty object called `map` as a property of the class.

The `put` method is defined as a prototype function of the `MyHashMap` class. It takes two parameters, `key` and `value`, and assigns the `value` to the `map` object using the `key` as the identifier.

The `get` method is also defined as a prototype function of the `MyHashMap` class. It takes one parameter, `key`, and checks if the `map` object has a property with that key. If the property exists, it returns the associated value; otherwise, it returns `-`.

The `remove` method is defined as a prototype function of the `MyHashMap` class. It takes one parameter, `key`, and deletes the property with that key from the `map` object.

To summarize, I've designed a hash map data structure with the ability to add, retrieve, and remove key-value pairs.

Author: Trevor Morin