Set

Bài gốc: Tất cả những gì bạn cần biết về Map và Set trong Javascript

Set là gì

`Set` là một tập hợp đặc biệt – "tập hợp các giá trị (value)" (không có key), nơi mà mỗi value chỉ có thể xuất hiện được một lần.

Các phương thức chính của nó:

- `new Set(iterable)` – tạo set và nếu một `iterable` object được cung cấp (thường là một array), copy các giá trị từ nó vào set.
- `set.add(value)` – thêm một value, return về chính set đó
- `set.delete(value)` – xóa một value, return `true` nếu value tồn tại tại thời điểm gọi, không thì là `false`.
- `set.has(value)` – return `true` nếu value tồn tại trong set, không thì `false`
- `set.clear()` – xóa mọi thứ từ set
- `set.size` – đếm số lượng phần tử

Điều đặc biệt là khi gọi `set.add(value)` nhiều lần với cùng một value thì cũng không có tác dụng gì cả. Nguyên nhân là mỗi value chỉ được xuất hiện một lần trong Set.

Ví dụ, chúng ta có các vị khách đến thăm, chúng ta muốn ghi nhớ có bao nhiêu người riêng biệt. Nhưng nếu các vị khách đến thăm nhiều lần dễ dẫn đến chúng ta nhớ nhầm. Một vị khách nên được "tính" là một lần mà thôi.

`Set` sinh ra để làm điều như vậy:

```
let set = new Set()
let john = { name: 'John' }
let pete = { name: 'Pete' }
let mary = { name: 'Mary' }
// Một số user đến thăm nhiều lần
set.add(john)
set.add(pete)
set.add(mary)
set.add(john)
set.add(mary)
// set giữ cho value là duy nhất
alert(set.size) // 3
for (let user of set) {
  alert(user.name) // John (sau đó là Pete và Mary)
}
```

Một trường hợp thường dùng `Set` là loại bỏ các phần tử trùng lặp trong một mảng. Nếu theo những cách thông thường thì ta phải lặp qua các phần tử của mảng, nhưng cách này thì hiệu suất khá là tệ. `Set` mang đến

một hiểu suất tốt hơn nhiều.

```
const arr = [1, 2, 3, 1, 3]
const newArr = Array.from(new Set(arr)) // [1, 2, 3]
const newArr2 = [...new Set(arr)] // [1, 2, 3]
```

Lặp qua Set

Chúng ta có thể lặp qua một set với `for..of` hoặc sử dụng `forEach`

```
let set = new Set(['oranges', 'apples', 'bananas'])
for (let value of set) alert(value)
// tương tự với forEach:
set.forEach((value, valueAgain, set) => {
  alert(value)
})
```

Để ý một điều hài hước là callback function được truyền vào `forEach` có 3 đối số: `value`, tiếp theo cũng là value nhưng với tên gọi khác là `valueAgain`, sau cùng là `object đích`. Thật vậy, cùng một giá trị nhưng lại xuất hiện 2 lần đối số.

Đó là để tương thích với `Map` khi mà callback được truyền trong `forEach` cũng có 3 đối số. Trông có vẻ lạ lạ nhưng để cho chắc chắn đó mà 😀 . Nó có thể giúp thay thế `Map` với `Set` lẫn nhau trong một số trường hợp nhất định hoặc ngược lại.

Một số phương thức tương tự như `Map` cũng được hỗ trợ

- `set.keys()` – return một iterable object cho các value
- `set.values()` – giống như `set.keys()`, cho việc tương thích với `Map`
- `set.entries()` – return một iterable object cho các entry `[value, value]`, cũng cho việc tương thích với `Map` luôn

Tóm lại

`Map` là tập hợp các cặp value được định danh bằng key, các key có thể thuộc bất cứ kiểu dữ liệu nào.

`Set` là tập hợp các value riêng biệt.

Việc lặp qua `Map` và `Set` thì luôn luôn được theo thứ tự, vì thế chúng ta không thể nói rằng những tập hợp các tập hợp này không được sắp xếp, nhưng chúng ta không thể sắp xếp lại các phần tử hoặc trực tiếp get một phần từ bằng vị trí thứ tự của nó.

Tham khảo

Bài viết được dịch từ Map and Set tại javascript.info