Map

Map是一组键值对的结构，具有极快的查找速度。

举个例子，假设要根据同学的名字查找对应的成绩，如果用Array实现，需要两个Array：

**var** names = ['Michael', 'Bob', 'Tracy'];

**var** scores = [95, 75, 85];

给定一个名字，要查找对应的成绩，就先要在names中找到对应的位置，再从scores取出对应的成绩，Array越长，耗时越长。

如果用Map实现，只需要一个“名字”-“成绩”的对照表，直接根据名字查找成绩，无论这个表有多大，查找速度都不会变慢。用JavaScript写一个Map如下：

**var** m = **new** Map([['Michael', 95], ['Bob', 75], ['Tracy', 85]]);

m.get('Michael'); *// 95*

初始化Map需要一个二维数组，或者直接初始化一个空Map。Map具有以下方法：

**var** m = **new** Map(); *// 空Map*

m.set('Adam', 67); *// 添加新的key-value*

m.set('Bob', 59);

m.has('Adam'); *// 是否存在key 'Adam': true*

m.get('Adam'); *// 67*

m.**delete**('Adam'); *// 删除key 'Adam'*

m.get('Adam'); *// undefined*

由于一个key只能对应一个value，所以，多次对一个key放入value，后面的值会把前面的值冲掉：

var m = new Map();

m.**set**('Adam', 67);

m.**set**('Adam', 88);

m.get('Adam'); // 88

Set

Set和Map类似，也是一组key的集合，但不存储value。由于key不能重复，所以，在Set中，没有重复的key。

要创建一个Set，需要提供一个Array作为输入，或者直接创建一个空Set：

**var** s1 = **new** Set(); *// 空Set*

**var** s2 = **new** Set([1, 2, 3]); *// 含1, 2, 3*

重复元素在Set中自动被过滤：

var s = new **Set**([1, 2, 3, 3, '3']);

s; // **Set** {1, 2, 3, "3"}

注意数字3和字符串'3'是不同的元素。

通过add(key)方法可以添加元素到Set中，可以重复添加，但不会有效果：

s.add(4);

s; // **Set** {1, 2, 3, 4}

s.**add**(4);

s; // 仍然是 **Set** {1, 2, 3, 4}

通过delete(key)方法可以删除元素：

var s = new **Set**([1, 2, 3]);

s; // **Set** {1, 2, 3}

s.**delete**(3);

s; // **Set** {1, 2}

小结

Map和Set是ES6标准新增的数据类型，请根据浏览器的支持情况决定是否要使用。

#### Iterable

遍历Array可以采用下标循环，遍历Map和Set就无法使用下标。为了统一集合类型，ES6标准引入了新的iterable类型，Array、Map和Set都属于iterable类型。

用for ... of循环遍历集合，用法如下：

**var** a = ['A', 'B', 'C'];

**var** s = **new** Set(['A', 'B', 'C']);

**var** m = **new** Map([[1, 'x'], [2, 'y'], [3, 'z']]);

**for** (**var** x of a) { *// 遍历Array*

console.log(x);

}

**for** (**var** x of s) { *// 遍历Set*

console.log(x);

}

**for** (**var** x of m) { *// 遍历Map*

console.log(x[0] + '=' + x[1]);

}

你可能会有疑问，for ... of循环和for ... in循环有何区别？

for ... in循环由于历史遗留问题，它遍历的实际上是对象的属性名称。一个Array数组实际上也是一个对象，它的每个元素的索引被视为一个属性。

当我们手动给Array对象添加了额外的属性后，for ... in循环将带来意想不到的意外效果：

**var** a = ['A', 'B', 'C'];

a.name = 'Hello';

**for** (**var** x **in** a) {

console.log(x); *// '0', '1', '2', 'name'*

}

for ... in循环将把name包括在内，但Array的length属性却不包括在内。

for ... of循环则完全修复了这些问题，它只循环集合本身的元素：

**var** a = ['A', 'B', 'C'];

a.name = 'Hello';

**for** (**var** x of a) {

console.log(x); *// 'A', 'B', 'C'*

}

这就是为什么要引入新的for ... of循环。

forEach

Set与Array类似，但Set没有索引，因此回调函数的前两个参数都是元素本身：

**var** s = **new** Set(['A', 'B', 'C']);

s.**forEach**(**function** (element, sameElement, set) {

console.log(element);

});

Map的回调函数参数依次为value、key和map本身：

**var** m = **new** Map([[1, 'x'], [2, 'y'], [3, 'z']]);

m.**forEach**(**function** (value, key, map) {

console.log(value);

});

如果对某些参数不感兴趣，由于JavaScript的函数调用不要求参数必须一致，因此可以忽略它们。例如，只需要获得Array的element：

**var** a = ['A', 'B', 'C'];

a.**forEach**(**function** (element) {

console.log(element);

});