**ES5**

**一：数组包含一些可以直接比较的值：**

clipboard.png

**forf方法**



function unique(a) {

var res = a.filter(function(item, index, array) {

return array.indexOf(item) === index; });

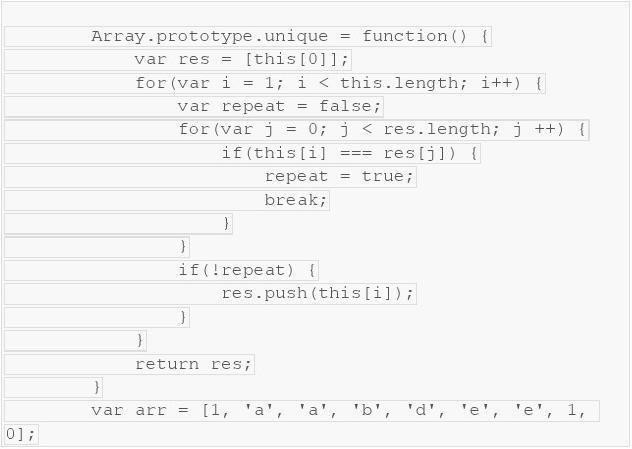
return res;

}

var a = [1, 1, '1', '2', 1];

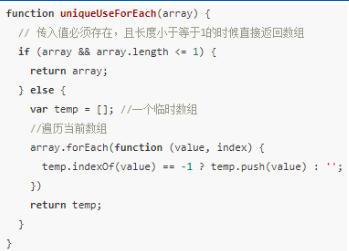
var ans = unique(a);

console.log(ans); // => [1, "1", "2"]



缺点：当数组数量较大时效率低，性能低。

**forEach()方法**



缺点：foreach方法不能退出循环

**对象键值法（hash)**

function unique(a) {

var ret = [];

var hash = {};

for (var i = 0, len = a.length; i < len; i++) {

var item = a[i];

var key = typeof(item) + item;

if (hash[key] !== 1) {

ret.push(item); hash[key] = 1;

} } return ret;

}

var a = [1, 1, 3, 2, '4', 1, 2, 4, '1']; var ans = unique(a); console.log(ans); // => [1, 3, 2, "4", 4, "1"]

注：该方法执行的速度比其他任何方法都快， 就是占用的内存大一些。

**排序后相邻去除法**

function uniq(array){

array.sort();

var temp=[array[0]];

for(var i = 1; i < array.length; i++){

if( array[i] !== temp[temp.length-1]){

temp.push(array[i]); } }

return temp;

}

var aa = [1,2,"2",4,9,"a","a",2,3,5,6,5]; console.log(uniq(aa));

但如果数组中出现1和‘1’时 sort()方法中的tostring方法会将1和‘1’看成一样的

该方法比for方法高效，但是最后获得的结果总是排序之后的，不适用于不改变数组顺序去重

**对于有NAN类型值的数组去重：**

clipboard.png

NAN不全等于自身 判断NANa的方法：

isNAN(NAN);//true

ES5中NAN===NAN;//false

NAN==NAN;//true



**ES6**

***SET***

**ES6 提供了新的数据结构 Set。它类似于数组，但是成员的值都是唯一的，没有重复的值。**

Set 本身是一个构造函数，用来生成 Set 数据结构。Set 结构不会添加重复的值。

遍历器（Iterator）就是这样一种机制。它是一种接口，为各种不同的数据结构提供统一的访问机制。任何数据结构只要部署 Iterator 接口，就可以完成遍历操作（即依次处理该数据结构的所有成员）。

Iterator 接口主要供for...of消费。

扩展运算符（...）内部使用for...of循环，所以也可以用于 Set 结构。

clipboard.png

**Set 对象允许你存储任何类型的 唯一值 ，无论是原始值或者是对象引用。**

**Set对象是值的集合，你可以按照插入的顺序迭代它的元素。**

**Set中的元素只会出现一次，即 Set 中的元素是唯一的。**

**Array.from() 方法从一个类似数组或可迭代的对象(包括 Array，Map，Set，String，TypedArray，arguments 对象等等) 中创建一个新的数组实例**

**set内部会自动===比较 nan等于自身**

function unique(arr) {
//通过Set对象，对数组去重，结果又返回一个Set对象
//通过from方法，将Set对象转为数组
return Array.from(new Set(arr))
}

Array.from(arrayLike[, mapFn[, thisArg]])

|  |  |
| --- | --- |
| arrayLike | 必需，想要转换成真实数组的类数组对象或可迭代的对象。 |
| mapFn | 可选，如果指定了该参数，则最后生成的数组会经过该函数的加工处理后再返回。 |
| thisArg | 可选，执行 mapFn 函数时 this 的值。 |

**2.MAP**

JavaScript 的对象（Object），本质上是键值对的集合（Hash 结构），但是传统上只能用字符串当作键。键会自动转换成字符串

为了解决这个问题，ES6 提供了 Map 数据结构。它类似于对象，也是键值对的集合，但是“键”的范围不限于字符串，各种类型的值（包括对象）都可以当作键。也就是说，Object 结构提供了“字符串—值”的对应，Map 结构提供了“值—值”的对应，是一种更完善的 Hash 结构实现。如果你需要“键值对”的数据结构，Map 比 Object 更合适。

**Map是ES6 提供的新的数据结构。**

**Map 对象保存键值对。任何值(对象或者原始值) 都可以作为一个键或一个值。**

**const map = new Map([
['name', '张三'],
['title', 'Author']
]);**

**map.get('name') // "张三"**

**Map构造函数接受数组作为参数，实际上执行的是下面的算法。**

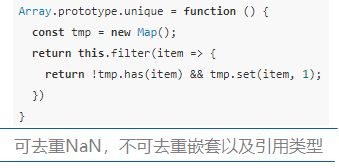
**const items = [
['name', '张三'],
['title', 'Author']
];
const map = new Map();**

**items.forEach(
([key, value]) => map.set(key, value)
);**

**注：Map中不会出现相同的key值**

**function** **arrayNonRepeatfy**(arr) {
**let** hashMap = **new** Map();
**let** result = **new** Array(); // 数组用于返回结果
**for** (**let** i = 0; i < arr.length; i++) {
**if**(hashMap.has(arr[i])) { // 判断 hashMap 中是否已有该 key 值
hashMap.set(arr[i], true); // 后面的true 代表该 key 值在原始数组中重复了，false反之
} **else** { // 如果 hashMap 中没有该 key 值，添加
hashMap.set(arr[i], false);
result.push(arr[i]);
}
}
**return** result;
}
**let** arr = [1, 1, 1, 2, 3, 3, 4, 5, 5, "a", "b", "a"];
console.log(arrayNonRepeatfy(arr)); // [ 1, 2, 3, 4, 5, 'a', 'b' ]

**数组去重简单的方法：21ms**



//返回arr数组过滤后的结果，结果为一个数组

//过滤条件是，如果res中没有某个键，就设置这个键的值为1

//filter() 方法创建一个新的数组，新数组中的元素 是 通过检查 指定数组 中 符合条件的所有元素。

**4.Array.prototype.includes()方法**

