// 没有返回值

// 对数组中的每一项运行给定函数，这个方法没有返回值

// 重要的一点是：可以添加第二参数，而且回调函数中的this会指向这个参数。而如果没有第二参数，则this会指向window。

let arr = [1, 2, 3, 4, 5, 6, 7];

let a = arr.forEach(function(item, index, array) {

// 当前元素，下标，数组

console.log(item);

});

console.log(a); // undefined

自己实现forEach

if (!Array.prototype.forEach) {

Array.prototype.forEach = function(callback, obj) {

var len = this.length; // 数组长度

// 如果数组长度为0，或者回调第一个参数不是函数，则返回

if (len < 1 || typeof(callback) !== 'function') {

return;

};

// 如果第二个参数没有传，则函数中的this指向window

obj = typeof(obj) === 'undefined' ? window : obj;

for (var i = 0; i < len; i++) {

callback.call(obj, this[i], i, this);

};

};

};

// 返回一个数组，长度同原数组

// 对数组中的每一项运行给定函数，返回每次函数调用的结果组成的数组

// 重要的一点是：可以添加第二参数，而且回调函数中的this会指向这个参数。而如果没有第二参数，则this会指向window。

let arr = [1, 2, 3, 4, 5, 6, 7];

let newArr = arr.map(function(item, index, array) {

return item \* 2;

});

console.log(newArr); // Array [ 2, 4, 6, 8, 10, 12, 14 ]

// 返回一个数组

// 对数组中的每一项运行给定函数，返回该函数会返回true的项组成的数组

// 重要的一点是：可以添加第二参数，而且回调函数中的this会指向这个参数。而如果没有第二参数，则this会指向window。

let arr = [1, 2, 3, 4, 5, 6, 7];

let newArr = arr.filter(function(item, index, array) {

return item > 4;

});

console.log(newArr); // Array [ 5, 6, 7 ]

// every(每一个) 返有布尔值

// 对数组中的每一项运行给定函数，如果该函数对每一项都返回true，则返回true

// 重要的一点是：可以添加第二参数，而且回调函数中的this会指向这个参数。而如果没有第二参数，则this会指向window。

let arr = [1, 2, 3, 4, 5, 6, 7];

let a = arr.every(function(item, index, array) {

return !isNaN(item) && typeof(item) === 'number'; // 每一项是否是数字

});

console.log(a); // true

// some(一些) 返有布尔值

// 对数组中的每一项运行给定函数，如果该函数对任一项返回true，则返回true

// 重要的一点是：可以添加第二参数，而且回调函数中的this会指向这个参数。而如果没有第二参数，则this会指向window。

let arr = [1, 2, 3, 4, 5, 6, 7];

let a = arr.some(function(item, index, array) {

return item >= 7;

});

console.log(a); // true

// reduce(归并) reduceRight

// 迭代数组的所有项，然后构建一个最终返回的值

// 参数有两个：一个在每一项上调用的函数和（可选的）作为归并基础的初始值

// 函数接收四个参数：前一个值、当前值、项的索引和数组对象

// 这个函数返回的任何值都会作为第一个参数自动传给下一项

// 当没有初始值时，第一次迭代发生在数组的第二项上，因此它的迭代次数为：数组.length - 1

// 当有初始值时，第一次迭代发生在数组的第一项上，迭代次数和数组的长度相等

let arr = [1, 2, 3, 4, 5];

var num = arr.reduce(function(prev, item, index, array) {

return prev + item;

}, 10);

console.log(num); // 25

var str = '我爱你';

var newStr = str.split('').reduceRight(function(prev, item) {

return prev + item;

});

console.log(newStr);

// find方法，用于找出第一个符合条件的数组成员。它的参数是一个回调函数，所有数组成员依次执行该回调函数，直到找出第一个返回值为true的成员，然后返回该成员。如果没有符合条件的成员，则返回undefined。

// 回调函数接收三个参数，当前项，下标，数组本身

let arr = [1, 4, -5, 10, -1];

let a = arr.find(function(item, index, array) {

return item < 0;

});

console.log(a); // -5，返回第一个小于0的项

同上

let a = [1, 4, -5, 10, -1].find(n => n < 0);

console.log(a); // -5，返回数组中第一个小于0的成员

let a = [1, 4, -5, 10, -1].find(n => n > 100);

console.log(a); // undefined，没有大于100的成员

// findIndex 方法的用法与find方法非常类似，返回第一个符合条件的数组成员的位置，如果所有成员都不符合条件，则返回-1

let a = [1, 5, 10, 15].findIndex(function(item, index, array) {

return item > 9;

});

console.log(a); // 2