**学习目标**

1、掌握数组的迭代方法：

filter()

forEach()

map()

说明：

1、只有IE9及以上浏览器支持。自己封装

2、不修改原数组。

**filter()**

语法：arrayObject.filter(function(item,index,array){},scope)

功能：创建一个新的匹配过滤条件的数组。

参数： function：必需，要在每一项上运行的函数，该函数必须要返回一个值，由这个值组成新的数组； scope：可选，运行该函数的作用域对象----影响this的值。 item:数组项的值；index:该项在数组中的位置；array:数组本身

返回值：数组

例：

var nums=[1,2,5,8,10,9,3,6];

自己封装一个方法：将nums这个数组中所有大于3的值都取出来存到一个新数组。

function filterArray(arr) {

var newArr = [],

i, len = arr.length;

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

if(arr[i] > 3) {

newArr.push(arr[i]);

}

}

return newArr;

}

var newNums = filterArray(nums);

//newNums.sort(function(a,b){return a>b});

//console.log(newNums);

使用filter方法。讲明函数做为参数。（参数可以是任何数据类型）

var newNums2 = nums.filter(function(item, index, arr) {

return item > 3;

})

// 将nums这个数组中所有的偶数打印出来

var newNums3 = nums.filter(function(item, index, arr) {

if(item % 2 === 0) return item;

})

console.log(newNums3);

**map()**

语法：arrayObject.map(function(item,index,array){},scope)

功能：对数组的每个元素进行一定操作（映射）后，会返回一个新数组。

参数： function：必需，要在每一项上运行的函数； scope：可选，运行该函数的作用域对象----影响this的值。 item:数组项的值；index:该项在数组中的位置；array:数组本身

说明：map()是处理服务器返回数据时是一个非常实用的函数。

例：返回数组中的每一项乘以2的结果组成的新数组，可以自己实现一个这个方法。

var nums = [3, 8, 5, 7, 6];

var newNums = nums.map(function(item, index, array) {

return item \* 2;

})

console.log(nums);

例二：

var strs = ["display", "transform", "transition"];

var newstrs = strs.map(function(item, index, array) {

return "-webkit-" + item;

})

console.log(newstrs);

**forEach()**

语法：arrayObject.forEach(function(item,index,array){},scope)

功能：对数组中的每一项运行传入的函数。

参数： function：必需，要在每一项上运行的函数； scope：可选，运行该函数的作用域对象----影响this的值。 item:数组项的值；index:该项在数组中的位置；array:数组本身

说明：forEach()无返回值。 类似于for循环

例：

var colors = ["red", "green", "blue"];

colors.forEach(function(item, i, array) {

console.log(item);

})