正课:

1. ES5

2. ES6

1. ES5:

数组函数:

1. 判断:

1. 判断数组中是否所有元素都符合条件

var bool=arr.every(function(elem,i,arr){

return 判断条件

})

原理: every会自动遍历arr中每个元素。每遍历一个元素，就自动调用一次回调函数。然后自动给回调函数传输三个参数: 1. 当前元素值->elem, 2. 当前位置->i, 3. 当前数组->arr。然后，every收集每次回调函数执行的返回值。如果每个元素经过回调函数的检查，都返回true，则整个every返回true。只要碰上一个元素检查返回false，则整个every返回false

2. 判断数组中是否包含符合条件的元素

var bool=arr.some(function(elem,i,arr){

return 判断条件

})

原理: some自动遍历arr中每个元素，然后自动在每个元素上调用回调函数。只要一个元素经过回调函数的检查可以返回true，则整个some就可以返回true。

除非所有元素经过回调函数的检查，都返回false，some才返回false！

2. 遍历:

什么是: 对数组中每个元素依次执行相同的操作

包含: 2个代替for循环执行遍历的函数:

1. 遍历数组中每个元素，对元素执行相同的操作

对 每个

arr.forEach(function(elem,i,arr){

对数组中当前元素执行的操作

})

原理: forEach也会自动遍历数组中给每个元素。对每个元素都调用一次回调函数。

强调: 回调函数不需要返回值！

2. 依次取出数组中每个元素，执行相同操作后，放入新数组返回。

何时: 希望保护原数组不被改变

如何:

var newArr=arr.map(

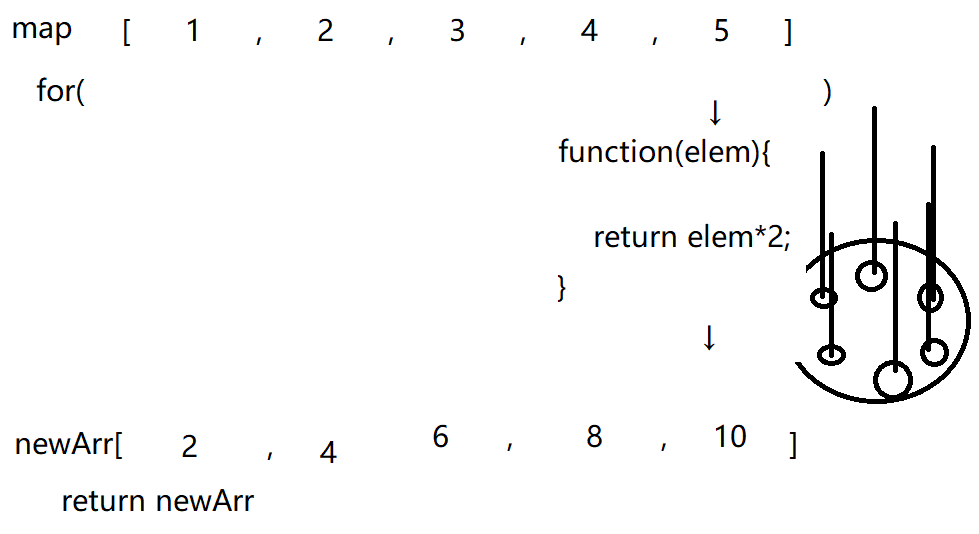
function(elem,i,arr){

return 加工后的新元素值

}

)

原理: arr.map()会先创建一个空数组等着！然后map才自动遍历原数组arr中每个元素值。不修改原数组。而是复制出原数组中的元素值，修改后，自动放到新数组相同位置上去。



3. 过滤和汇总:

1. 过滤: 复制出数组中符合条件的元素组成新数组

如何: var subArr=arr.filter(

function(elem,i,arr){

return 判断条件

}

)

原理: filter先创建一个空数组等待。然后还是遍历arr中每个元素。每遍历一个元素，就调用一次回调函数。在函数内判断当前元素是否符合要求。如果符合要求，就放入新数组中。最后，返回新数组。

强调: 原数组保持不变！

2. 汇总: 针对数组中的元素，进行统计，最后得出统计结果。

var result=arr.reduce(

function(prev, elem, i, arr){

//prev是截止到目前，临时的汇总值。

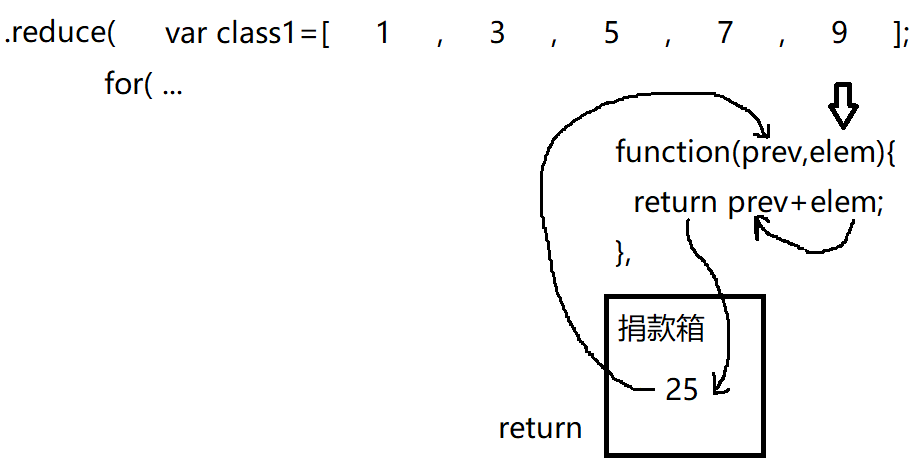
return 将当前值汇总到临时结果中产生的新结果。

},

起始值

)

原理: reduce()会自动遍历arr中每个元素，然后将当前元素，汇总到统计结果中。最后，返回统计结果。



强调: 这六个数组函数因为内部内置的只是普通for循环，所以支持索引数组！不支持关联数组！

2. ES6: 在不改变原理的基础上，让代码更简化！

1. let:

什么是: 代替var，用于声明变量的关键词。

为什么: var有两个广受诟病的缺点:

1. 声明提前！

2. 没有块级作用域！

何时: 今后所有var，都要用let代替！

优点: 1. 不会被声明提前

2. 添加了块级作用域: 块内和块外即使有相同的变量，也不会互相影响！

原理: let做了两件事:

1. let等效于匿名函数自调！

2. let为了进一步避免可能的冲突将变量悄悄改了名字。比如: let t=0 -> \_t=0

比如: let t=0;

等效于:

(function(){

var t=0;

... ...

})()

所以, let有三个小脾气:

1. 全局写let t=0, 用window.t却找不到

2. 同一级作用域内，不能在let t=0前，提前使用使用t。即使外层作用域有t变量，也不行！

3. 相同作用域内，不允许重复let同一个变量。

let有个兄弟, const: 专门用于声明一个值不能修改的常量。同时也具有let的特点：不会被声明提前; 添加了块级作用域。

2. 箭头函数:

什么是: 专门代替function来简化创建函数的新语法。其它语言中也叫拉姆达（lambda）表达式

为什么: 简化！

何时: 今后，能用箭头函数简写的地方，优先使用箭头函数简写！

如何: 3句话:

1. 去掉function，在()和{}之间加=>

2. 如果只有一个形参，可省略()

3. 如果函数体只有一句话 ，可省略{}

但是，如果这句话是return，则必须省略return

提醒: 去掉{}后，记得去掉结尾的分号;

坑: 双刃剑:

箭头函数内外this是相通的。

如果希望函数内外的this保持一致时，可以不用bind()换，而直接改为箭头函数即可！

如果反而不希望函数内外的this一致，就是希望内部的this和外部的this不一样时，不能改为箭头函数的。

比如: 对象中的方法的function，就不能改为=>

如果改为=>，则this等效于全局的this->window。

单词列表:

1. initialization 初始化: 第一次声明+赋值 是初始化

比如: var a=10;

2. access 访问

3. Identifier 标识符, 变量名, 函数名, ...

4. declare 声明

5. operator 运算符

6. arrow 箭