1. arr1.every(function(elem,i,arr){ })内，如果想使用.前的arr1，为什么不能用this直接找到.前的对象，而只能用形参中的arr呢？

答：所有回调函数的执行，都是自动执行的，且没有.前缀。所以，绝大多数回调函数中的this，都指window。

再比如:

var lilei={

sname:"Li Lei",

intr:function(){ console.log(`I'm ${this.sname}`)}

};

lilei.intr(); //this->.前的lilei，所以可以正常输出lilei.sname

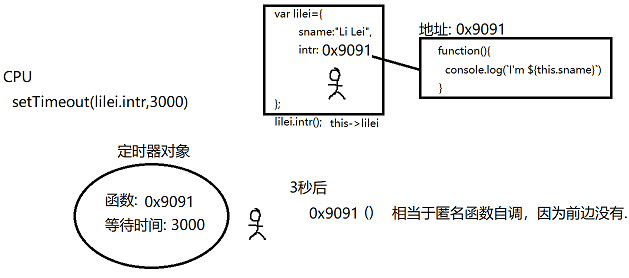
但是:

setTimeout(lilei.intr,3000); //3秒后输出的却是undefined

1. 为什么lilei.intr不加()？因为定时器不是立刻调用。而仅仅是给定时器拿走，等3秒后，由定时器自动调用。

2. 为什么输出undefined，而不是lilei.sname?

答: 因为定时器中的函数，也是回调函数。定时器在拿走函数时，只是单纯的拿走了函数的地址，而没有同时拿走.前的对象lilei。3秒后执行函数时，是直接用函数的地址加()调用的函数。早就不知道函数的所有者是谁了。仅相当于匿名函数自调。匿名函数自调中的this->window，而window.sname当然返回undefined。



2. 什么是回调函数(callback)?

答: 咱们自己定义的函数，却不由咱们自己亲自调用，而是交给别的函数去调用。

比如: setTimeout(function(){},3000)里的function(){}，虽然是咱们定义的，但是调用时setTimeout等三秒后调用的。

再比如: arr.every(function(elem,i,arr){})里的function(){}，虽然是咱们定义的，但是，调用几次咱们不知道，每次传入什么实参也不知道。完全由every()在内部自动调用。

还有arr.sort(function(a,b){}), str.replace(/正则/g,function(kword){}), xhr.onreadystatechange=function(res){}，这些都是回调函数。

3. 何时加()，何时不加()?

答: 主动想调用一次函数时，就要加()

不调用函数，只是将函数给别人拿走，将来由别人负责调用时。不加()!

4. 0.1+0.5==0.6 为true，而 0.1+0.2==0.3为什么为false？

答：舍入误差: 计算机中底层采用二进制计算，也有计算不尽的数值。就好像现实中10/3得几？3.3333333333....说几个3也无法精确。因为根本除不尽。计算机中的二进制更过分，有加不尽，减不尽，乘不尽。比如:

2-1.6=0.3999999999999999...

555.55\*100=55554.99999999999

等等，不确定什么时候，这种舍入误差就会出现。

只要是浮点数参数算数计算都有可能产生舍入误差，但是和十进制一样，不是所有情况都舍入误差。比如: 10/5是可以除尽的。10/2也是可以除尽的。所以，舍入误差不可预期。

现实中解决舍入误差：计算式保留过精度小数位数，比如保留8位小数进行计算。包括数据库中存储数字，也用八位小数。只有要显示给用户看时，再用toFixed()，按2位小数或者按整数四舍五入。

5. [{x:1},{x:2},...]数组中保存的实际上是对象的地址。但是[1,2,3]里保存的也是1,2,3的地址吗？

答: 其实数组中的一个元素，也是一个变量，只不过是放在数组中的一个变量而已。只要是变量，都只能存一个值。如果是原始类型的值，可直接保存在变量内。如果引用类型的对象，则只能保存在变量外部，通过地址值引用对象。

同理，对象的属性，其实也是一个变量，只不过保存在对象内部而已。

[{x:1},{x:2},...] 里边的对象，其实是存在数组外部的，每个元素只保存对象的地址。

[1,2,3]里边的原始类型的数值，其实是直接保存元素内部的。

6. 匿名函数和回调函数有什么差别和关系?

答: 回调函数不一定是匿名函数，匿名函数也不一定是回调函数。

只有当回调函数只使用一次时，才会定义为匿名函数。如果回调函数可能反复使用，也要定义为有名称的函数。

比如: setTimeout(function(){ ... }, 3000)

如果这句话需要反复使用:

不好的做法: 将setTimeout(连同里边的function(){}定义，反复复制多次！

就应该: 只定义一次函数，然后再定时器中用函数名反复引用函数:

function task(){ ... }

setTimeout(task,3000)

setTimeout(task,2000)

setTimeout(task,1000)

总结: 是否使用匿名函数，取决于将来这个函数被几处使用。如果只被一处使用，则就用匿名函数。如果多处都需要使用这个函数，则必须起名！

另一方面: 匿名函数也有主动自调的情况，就不属于回调。

总结: 之所以大多数回调函数都是匿名函数，只是因为多数回调函数只在一个位置使用一次！所以，没必要起名。

7. 闭包中的内层函数，算不算回调函数?

答: 不算！外层函数生出的内层函数，需要我们自己主动调用。调用几次，就反复执行几次。每次调用时，也需要我们主动传入实参值。

8. 数组新函数何时只用一个elem形参，何时三个形参都要用?

答: 首先，无论你写几个形参，人家一定会在调用时，自动给你三个实参值。如果你不接，大不了就不要了！

具体定义几个形参，取决于本次操作所需几个数据才能正常执行。

如果只靠elem一个值就可正常执行全部逻辑，就不要定义后两个形参。

如果只靠elem不能完成所有操作，必须还依赖位置i和当前数组对象arr，则必须提供三个形参。

9. 两次统计都用total作为变量名，难道不冲突吗？能不能起两个变量名？

答: 从语法上来说，是不冲突的。无非就是给一个变量，先后赋了两次值而已。

从用法上来说：如果整个程序只有一个捐款箱，最好只起一个名字。如果两个班的捐款分开统计，也就是有两个捐款箱时，反而应该起两个名字。

10. arr.reduce(function(prev,elem){

...

return 的是什么东西？

},

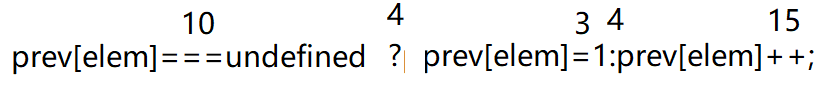
{}

)

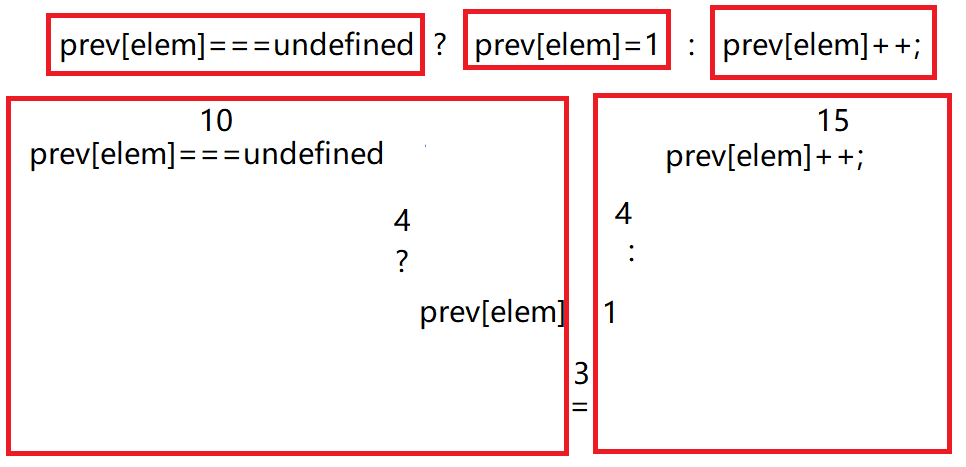
答: return的必须是修改后的整个捐款箱替换回去。而不应该是return一个表达式。所以 return的只能是prev

11. prev[elem]===undefined?prev[elem]=1:prev[elem]++;是错的？

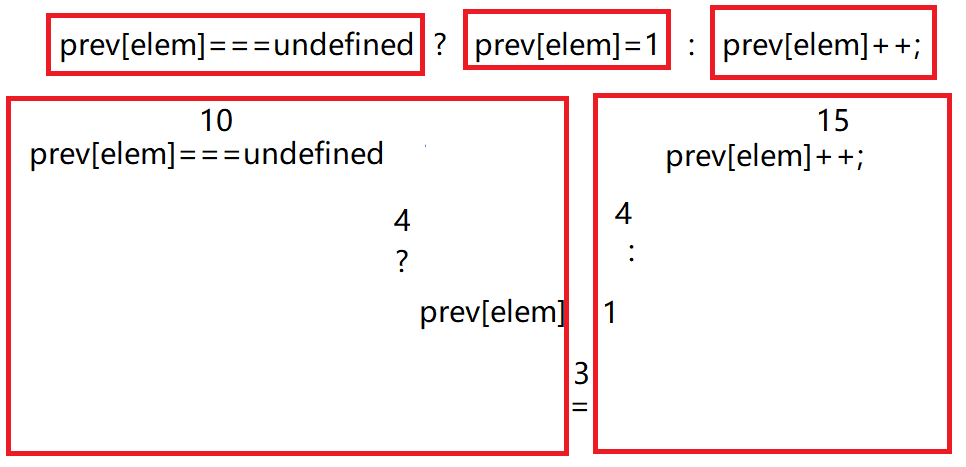
答: 因为运算符优先级导致程序自动断句错误！



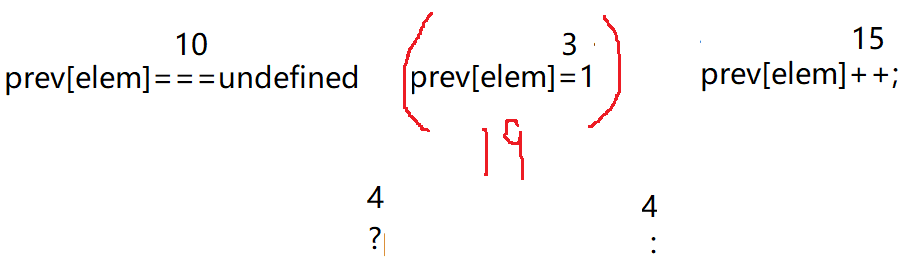
会被断句成:



但是我们希望



解决: 给希望整体优先执行的表达式加()，因为()优先级永远最高！

如何: 

12. prev.elem不对？

答: 复习！复习对象克隆！

13. 为什么reduce的第二个参数要用{}?

答: 这是由最终希望获得的结果的数据结构定的。

最终希望获得: h:1, e:1, l:3, o:2, w:1, r:1, d:1

在程序中保存键值对的集合，最好的办法是{}，其次可以用[]，因为在js中{}底层就是[]，两个的用法是完全一样的！

14. let t=0后，其余var s=10; 或function task1(){}，会不会也和t一样变成局部呢？

答：不会！let t一定会尽量保证不影响其他无关的变量或函数。比如，可以自动通过将var s=10，换成window.s=10，或者自动将function task1(){}改为window.task1()=function(){}等来保证其他非let的变量或函数，依然属于全局。

15. 箭头函数中，无法使用arguments

答: 解决: rest语法：

var add=(...nums)=>{

return nums.reduce((prev,elem)=>prev+elem,0)

}