如果看不见函数的执行过程，该怎么办: 在函数内添加console.log()输出函数的执行和参数，以及返回值

比如:

str=str.replace(/\b[a-z]/,function(kw){

console.log("===内层函数被调用一次===");

console.log("=====自动传入kw=${kw}");

console.log("=====本次调用返回${kw.toUpperCase()}")

return kw.toUpperCase()   
});

再比如:

var arr=[12,3,123,2,1,23];

arr.sort(function(a,b){

console.log("\*\*\*内层函数被调用了一次\*\*\*");

console.log(`\*\*\*\*\*本次用${a}和${b}做比较`);

console.log(`\*\*\*\*\*本次调用返回${a-b}`)

return a-b

})

提示: 在浏览器控制台中，按上下键可找回曾经执行过的代码。

console.log的输出问题：

var arr=[12,3,123,2,1,23];

console.log(arr); //排序前输出数组的内容

arr.sort(function(a,b){ return a-b })

console.log(arr); //排序后再输出数组的内容

执行结果:

▶ arr

1

2

3

12

23

123

▶ arr

1

2

3

12

23

123

问题: 两次输出的结果都是排序后的。

原因: console.log在首次输出对象时，仅是输出的对象引用地址。当我们点旁边的三角时，才自动去内存中找到对象现在的样子，展开对象。但此时，程序执行完了。

—— console.log()不是拍照！无法留住对象的瞬时内容

解决: 将对象的内容转为字符串才是拍照/打印

console.log(String(arr)) [ 1, 2, 3, 12, 23, 123 ]

|| "1,2,3,12,23,123"

arr.toString()

转 字符串

基础好的同学:

<http://code.tarena.com.cn/WEBCode/wfd1812/>

正课:

1. RegExp对象  
2. Function对象

1. RegExp对象

1. 验证: var bool=reg.test()

2. 查找关键词: 查找敏感词的内容和位置

var arr=reg.exec(str)

在字符串str中查找一个符合正则表达式reg要求的关键词

返回值: 跟str.match(reg)不加g，返回的值完全相同

arr [ 0: 关键词内容, index: 关键词的位置i ]

问题：每次只返回一个敏感词的内容和位置

解决：只要反复调用，reg.exec()会自动跳到下一个继续查找

找所有：用循环:

var reg=/小[\u4e00-\u9fa5]/g;

do{//反复do

//查找敏感词

var arr=reg.exec(str);

//如果找到敏感词，才输出

if(arr!=null){

console.log("找到敏感词!");

console.log(arr);

}else{//否则(找不到)，就退出循环

console.log("找不到了，就退出！")

break;

}

}while(true);

//true 不使用循环条件控制退出，因为我也不知道循环几次！

//而是在循环体内根据条件，用break随时可能退出循环。

2. \*\*\*\*\*Function对象:

什么是: 内存中保存一段可重用的代码段的对象，再起一个名字。

为什么: 重用！

何时: 如果一段代码段需要反复使用，就应该保存在一个函数中，反复调用函数。

如何:

创建: 3种:

1. 声明方式:

function 函数名(形参列表){

函数体;

return 返回值;

}

问题: 声明提前: 在程序开始执行前，会自动将所有var声明的变量和function声明的函数，都提前到当前作用域的顶部，集中创建。赋值留在原地

2. 用赋值的方式:

var函数名=function (形参列表){

函数体;

return 返回值;

}

优点: 仅函数名提前，函数体留在原地。

揭示了函数的重要秘密:

1. 函数名本质就是一个普通的变量

2. 而函数定义其实是一个对象(独立的存储空间)

3. 用new创建:

var fun=new Function("形参","形参",...,"...")

调用: var 返回值=函数名(实参值列表)

调用函数，等效于执行函数中的代码段

调用几次，就重复执行几次

如果不调用，是不会执行函数的内容的

重载:overload

什么是: 相同函数名不同形参列表的多个函数，在调用时，可根据传入实参的不同，自动选择匹配的函数调用。

为什么: 调用不方便！

何时: 同一件事，根据传入实参的不同，自动选择不同的执行过程执行时。

如何:

问题: js语言不支持多个同名函数同时存在，所以默认和不支持重载。

解决: 借助: arguments

什么是arguments: 每个函数中自动包含的可直接使用的接收所有传入函数实参值列表的类数组对象

类数组对象是: 长得像 数组的 对象

vs 数组: 相同: 1. 下标, 2. length, 3. for遍历

不同: 根本不同 类型不同 导致arguments不能使用数组家的任何函数。

匿名函数:

什么是: 创建函数时，不指定函数名

为什么: 2大作用:

1. 节约内存

2. 避免全局污染

何时: 2种:

1. 只要一个函数只使用一次

2. 划分临时作用域的时候，避免全局污染

如何: 2种:

1. 回调函数: 虽然我们自己定义的，但是不是我们自己调用执行的函数。

比如: arr.sort(function(a,b){return a-b})

str.replace(/\b[a-z]/g, function(kw){ return ... })

都是为了节约内存！

2. 匿名函数自调: 创建函数后，立刻调用函数

(function(){

var t=new Date();

document.write("<h1>页面加载完成，at: "+t.toLocaleString()+"</h1>")

})();

创建函数后，自己调用自己

优点: 不但自己调用后释放，且函数内的局部变量调用后，也会自动释放。

总结: 1. 今后所有js代码都要写在匿名函数自调中

2. 今后禁止使用全局变量：

1. 浪费内存; 2. 全局污染

今日英语单词:

1. sort 排序 将多个数值，按从大到小或从小到大的顺序排列。

2. while 只要...的时候

while(条件) 只要 满足条件 的时候 就做什么事儿

3. break 中断 用于中途停止循环。

4. function 功能 程序中所有反复使用的功能都定义在function 功能名(参数){ —— 像极了数学里的函数

}

5. declare 声明 其实就是"创建"

var a 创建一个变量a

function fun() 创建一个fun函数对象

6. return 返回 用于从函数内，向函数外返回一个结果值

7. hoist 吊车、起重机 将物体从低处，提高到高处

8. argument 参数

parameter 也叫参数

调用函数时()中定义的和传入的，都成为参数变量

只不过，定义时的变量成为 形参

调用时的变量称为 实参