案例复习:

每个案例三遍:

1. 代码+注释抄一遍

2. 保留注释，删除代码，自己试着把代码填回来

3. 都删掉，尝试将注释和代码一起写回来——自己会了

注释最重要！

正课：

1. String正则API

2. RegExp对象

3. \*\*\*Function对象

1. String正则API

查找：4种场景

1. 查找一个固定的关键词出现的位置:

var i=str.indexOf("敏感词",starti)

2. 用正则表达式模糊查找关键词的位置:

var i=str.search(/正则/i)

3. 查找关键词的内容:

1. 只查找第一个关键词的内容:

var arr=str.match(/正则/i)

arr:["0": 关键词内容, "index": 下标i ]

问题: 只能找一个敏感词

2. 查找所有敏感词的内容:

var arr=str.match(/正则/ig) global

arr:[ 敏感词1, 敏感词2, ... ... ]

返回所有找到的敏感词内容的列表

如果找不到返回null

强调: 凡是返回null的函数，都要先判断不是null，再使用。

4. 既查找所有关键词的内容，又能获得每个关键词的位置

reg.exec()

替换: 2种:

1. 简单替换: 将所有敏感词都替换为一致的新值

str=str.replace(/正则/ig,"新值")

2. 高级替换: 根据本次找到的敏感词的不同，动态选择不同的新值替换

str=str.replace(

/正则/ig,

//replace每找到一个敏感词，就自动调用一次回调函数

function(kword){ kword: y c y u

形参kword，自动获得本次找到的敏感词内容

return 返回的新值

kword.toUpperCase()

}// return Y C Y U

)

回调函数callback：咱们负责定义函数，然后将函数交给别的程序去自动执行。

特点: 1. 不一定立刻执行

2. 不止执行一次，可能反复执行

3. 自动传入参数

衍生: 删除敏感词: ——作业:

其实就是将敏感词替换为""

str=str.replace(/正则/ig,"")

切割: 将字符串按指定的分隔符，分割为多段子字符串

如何:

2种:

1. 简单切割: 分隔符是固定不变的

var arr=str.split("分隔符")

固定套路: 打散字符串为字符数组(按""切割)

var arr=str.split("")

2. 复杂切割: 分隔符不固定

var arr=str.split(/正则/)

2. RegExp对象:

什么是: 专门保存一条正则表达式，并提供用正则表达式执行查找和验证操作的方法。

何时: 只要在js中使用正则表达式，都要创建正则表达式对象

如何:

创建: 2种:

1. 用/方式:

var reg=/正则/ig;

问题: 不能在程序运行时，动态拼接正则，只能预先写死正则。

2. 用new:

var reg=new RegExp("正则","ig")

优势: 可在运行时动态拼接正则字符串

功能:

1. 验证:

var bool=reg.test(str)

验证str是否符合reg的格式要求

问题: 并不是完全匹配。只要在str中找到部分内容和reg匹配，就返回true！

解决: 要求从头到尾完整匹配

今后只要做验证，必须前加^，后加$

2. 查找: 查找所有敏感词的内容和位置

var arr=reg.exec(str)

在str中找下一个符合reg要求的敏感词的内容和位置

返回值: 同match不加g时一样

arr:[ "0": 敏感词内容, "index": 下标i ]

3. Function对象:

什么是: 保存一段可重用的代码段的对象，再起一个名字

为什么: 重用一段代码

何时: 只要发现一段代码需要反复使用时，都要定义为一个函数。

如何:

1. 声明方式:

function 函数名(形参列表){

函数体;

return 返回值;

}

var 返回值=函数名(实参列表)

问题: 会被声明提前:

2. 用赋值方式:

var函数名=function (形参列表){

函数体;

return 返回值;

}