正课:

1. String的正则函数

2. RegExp对象

3. Function对象

1. String的正则函数：

3件事:

1. 查找敏感词：4种场景:

1. 查找一个固定的关键词的位置

var i=str.indexOf("敏感词",fromi)

2. 用正则表达式查找多种敏感词的位置

var i=str.search(/正则/i)

问题: 只能获得位置，无法获得敏感词的内容

3. 查找敏感词的内容:

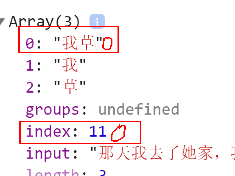
var arr=str.match(/正则/i)

在str中查找符合正则表达式要求的敏感词，返回其内容和位置。

返回值:

如果找到，返回一个数组arr:

|  |  |  |
| --- | --- | --- |
| 0 | ... | "index" |
| "敏感词" | ... | 位置i |



本质: 无论是0，还是index其实都是房间号（下标）。访问时都是用[下标]。只不过数字不加""，而字符串下标要加""。

如果想获得本次找到的敏感词内容:

arr[0]

\ 如果想获得本地找到的敏感词的位置:

arr["index"] 可简写为 arr.index

在运行时，arr.index会被自动翻译为["index"]

强调: 必须加""，如果不加，index就变成变量，而在全局没有名为index的变量，就会报错。

如果没找到，返回null ——重大隐患

null和undefined之后什么都不能跟，比如: 不能加. 不能加()，只要加就报错。

所以，今后，只要碰上可能返回null的函数，都要先判断不是null，再使用。

问题: 只能查找一个敏感词，就下班了！

解决: 正则还有一个后缀g, 意为global全部

如果正则加了g，就不能只找一个，而是必须找完所有才能下班。

坑: 一旦match中的正则加了后缀g，返回值就会有巨大变化:

如果找到，还是返回数组，只不过数组中就不包含敏感词的位置了，而仅包含多个敏感词的列表：

["敏感词1","敏感词2","敏感词3",...]

0 1 2

问题: 不加g，只能找一个，但是有位置

加g，可找所有，但是没有位置了

解决:

4. 即查找每个关键词的内容，又查找每个关键词的位置。

String自己做不了，依靠下午将的regexp对象

2. 替换：

1. 简单替换: 将所有敏感词都替换为统一的新值

str=str.replace(/正则/ig,"新值")

将str中所有符合正则要求的敏感词都替换为一致的"新值"

坑: 字符串都是不可变类型。一旦创建，内容就无法修改了。所有对字符串的修改，都无权直接修改原字符串。只能返回新字符串。所以，必须自己用=赋值的方式，将返回的新字符串保留下来。如果不赋值，就丢弃。

2. 高级替换: 根据不同的敏感词，选择替换成不同的新值

str=str.replace(

/正则/ig,

//第二个参数变成了一个匿名函数

function(kword){//要求1: 必须有一个形参

//要求2: 必须返回修改后的新值

return kword.toUpperCase();

}

)

原理:

replace会拿着正则表达式取str中找每个符合条件的敏感词。

每找到一个，就自动调用匿名函数。并自动将本次找到的敏感词传给形参。

在匿名函数内，加工形参，得到新值，被return返回到匿名函数外部。

返回的新值再被replace替换到str中当前位置。

衍生操作: 删除敏感词，其实即使替换为""

str=str.replace(/正则/ig,"")

3. 切割：按字符串中一个指定的切割符，将一个字符串，切成多段子字符串

何时: 将一个字符串的内容转为数组

如何: 2种:

1. 简单切割: 分隔符是固定的

var arr=str.split("分隔符")

强调: 切割后的数组内容中，不包含切割符

2. 复杂切割: 分隔符不是固定的

var arr=str.split(/正则/)

执行时，会将一切符合正则的字符都当做分隔符

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验证和查找功能的函数

何时: 今后只要在程序中使用正则表达式，就都需要创建正则表达式对象来保存一条正则表达式。

如何:

创建: 2种:

1. 用双/： var reg=/正则/ig

何时: 如果正则表达式是固定不变的，则首选着第一种方式快速创建。

问题: 如果正则不是固定的，需要根据其他数据动态生成，则//之间不能写js表达式。因为//之间是正则的地盘。

2. 用new: var reg=new RegExp("正则","ig");

强调: 用""，就不用加/了

何时: 如果正则表达式不是固定的，需要根据其他变量或数组动态拼接生成，就必须用new RegExp()

因为new RegExp的参数是字符串。在js中可以有无数种方法拼接出任意自己想要的字符串后，再给RegExp()

函数:

1. 验证字符串格式:

var bool=reg.test(str)

用正则表达式reg去检查字符串str是否符合正则的格式要求。返回bool值。所以，reg.test()可直接当做判断条件。

坑: 其实test默认只要在字符串中找到符合条件的内容，就返回true。不要求字符串和正则从头到尾完整匹配。

解决: 今后凡是验证，必须都要前加^，后加$

2. 第四种查找方式: 即找每个敏感词的内容，又找每个敏感词的位置。

var arr=reg.exec(str)

在字符串str中查找下一个符合正则表达式reg要求的敏感词的内容和位置。

强调: 1. 执行一次，只能找一个敏感词

2. 要想找所有，必须加g

3. 返回值: 同match不加g的情况

|  |  |  |
| --- | --- | --- |
| 0 | ... | index |
| 敏感词 | ... | 敏感词的位置i |

vs match:

match，无论反复调用多少次都只能找第一个敏感词。

exec，如果反复调用，会自动向后查找，每次都找下一个敏感词。如果找不到，就返回null

单词列表

1. match 匹配

2. property 属性 程序中特指内存中，对象里的属性

比如: arr.length

3. trim 去空格，掐头去尾

4. slice: 选取 切片 比如从生日蛋糕上切下一角

5. splice: 修改 (插入，删除，替换)

6. split: 切割