day1

张东

18301092802

1. 正则表达式: 见ftp中regexp.pdf

2. String的正则API

查找: 4个场景:

1. 查找一个固定的敏感词出现的位置:

回顾: str.indexOf()

var i=str.indexOf("敏感词",starti)

在str中，从starti位置开始，查找下一个“敏感词”的位置

省略第二个参数starti，默认从0位置开始查找

返回值: 如果找到，返回敏感词第一个字的位置

如果找不到，返回-1

问题: 只能查找一种固定的敏感词

2. 模糊查找符合正则表达式要求的敏感词:

var i=str.search(/正则/);

在str中查找第一个符合正则要求的敏感词的位置

返回值: 如果找到，返回敏感词第一个字的位置

如果找不到，返回-1

问题1: 正则默认都是区分大小写的

解决: 在第二个/后加后缀i, ignore

问题2: 只能返回位置i，无法返回敏感词的内容

3. 查询敏感词的内容: 2种情况:

1. 只查看第一个敏感词的内容和位置:

var arr=str.match(/正则/i);

在str中查找第一个符合正则要求的敏感词的内容和位置

返回值: 如果找到: arr:[ 0:"敏感词", index: 位置i ]

补充: js中其实一切都是关联数组

如果找不到: 返回null

强调: 如果一个函数有可能返回null，都要先验证不是null，再使用。

问题: 正则表达式默认只找到第一个就退出

解决: 在第二个/后加后缀g, global(全部)

2. 查找所有敏感词的内容:

var arr=str.match(/正则/ig)

在str中查找所有符合正则要求的敏感词的内容

返回值: 如果找到: arr: [ 敏感词1, 敏感词2,... ]

如果没找到，返回null

问题: 只能获得内容，无法获得位置

4. 查找每个敏感词的内容和位置: reg.exec()

替换: 2种:

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")

查找str中符合正则要求的所有敏感词，替换为统一的新值

强调: 所有字符串API是无权直接修改原字符串，总是返回新字符串作为处理结果，所以必须用变量接住返回的新字符串。

2. 高级替换: 可根据每次找到的敏感词不同，动态生成替换的新值

str=str.replace(/正则/ig,function(kword){

return 新值

});

查找str中每个符合正则要求的敏感词，每找到一个敏感词，就自动调用一次回调函数。每次调用时，都自动传入本次的敏感词作为参数kword，再将返回的新值，替换回字符串中。

衍生操作: 删除找到的敏感词，其实就是替换为""

切割: 将一个字符串按指定的分隔符切割为多段子字符串:

2种:

1. 简单切割: 切割符是固定的:

回顾: var arr=str.split("切割符")

2. 复杂切割: 切割符不是固定的

var arr=str.split(/正则/)

day2

1. RegExp对象: 正则表达式(Regular Expression)

什么是: 在程序中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API 的对象

何时: 只要在程序中保存和使用正则表达式时，都要先创建正则表达式对象

如何:

创建正则表达式对象: 2种

1. 用//直接量方式创建: var reg=/正则/ig;

何时: 如果正则表达式不需要动态生成时

2. 用new创建: var reg=new RegExp("正则"[,"ig"])

何时: 如果正则表达式需要用程序动态生成时

API:

1. 验证: var bool=reg.test(str)

用正则表达式reg检查一个字符串str的格式是否符合要求。

返回值: true/false

问题: 正则表达式，默认只要找到符合规则的内容，就返回true，不要求从头到尾完整匹配——所以，也有人用reg.test()做查找关键词

解决: 今后凡是验证必须前加^后加$，表示从头到尾完整匹配

2. 查找敏感词: reg.exec(str)

什么是: 查找str中每个符合reg要求的敏感词的内容和位置

何时: 只要既查找每个关键词的内容，又查找每个关键词的位置

如何:

var arr=reg.exec(str);

执行过程:

每次只能查找一个敏感词，

返回: arr: [ 0: "敏感词", index: 位置i ] ——同match不加g的情况

反复调用reg.exec可自动向后找！

找不到，返回null

2. Function:

什么是: 专门保存一段可重用的代码段的程序结构，再起一个名字

何时: 只要一段代码可能被反复使用，都要定义为函数

如何:

创建函数: 3种:

1. 声明方式: function 函数名(形参列表){

函数体;

return 返回值;

}

问题: 会被声明提前

解决: 改声明方式为赋值方式

2. 赋值方式: var函数名=function (形参列表){

函数体;

return 返回值;

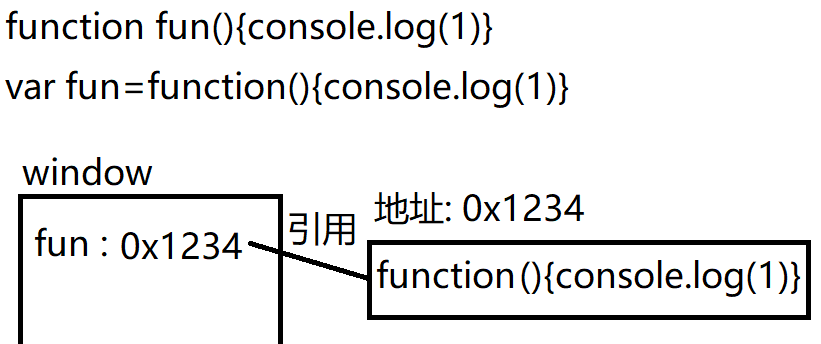
}

优点: 因为是赋值方式，不会被声明提前

揭示了js中函数的本质:

1. 函数其实也是一个保存一段代码的对象

2. 函数名其实只是一个普通的变量而已。函数名变量通过函数对象的地址引用着函数对象。所以使用函数名变量等效于使用函数对象本身。



3. 用new Function构造函数创建函数对象:

var 函数名=new Function("形参1","形参2",...,"函数体")

重载: overload

什么是: 相同函数名，不同参数列表的多个函数，在调用时，可根据传入实参的不同，自动选择匹配的函数执行。

为什么: 减少API的个数，减轻调用者的负担

何时: 一件事，可能根据传入参数的不同，执行不同的操作时

比如: arr.splice():

删除元素: arr.splice(i,n)

插入元素: arr.splice(i,0,新值1,...)

替换元素: arr.splice(i,n,新值1,...)

如何:

问题: js语法默认不支持重载

因为js中不允许多个同名函数同时存在

如果多个同名函数同时存在，则只有最后一个生效

解决: 借助arguments

arguments: 每个函数中自带的

自动接收所有传入函数的参数值的

类数组对象

类数组对象: 长得像数组的对象

vs 数组: 相同: 1. 下标, 2. length, 3. 遍历

不同: 类型不同！API不通用！

如何: 2步:

1. 仅定义一个函数，且不要定义形参

2. 在函数内判断arguments的length或内容，根据不同的参数，执行不同的操作

匿名函数:

什么是: 定义函数时，不指定函数名的函数

为什么: 节约内存

何时: 如果一个函数只使用一次，不会重用时，都要用匿名函数

如何: 2种；

1. 回调函数: 自己定义的函数，但是不是自己亲自调用，而是给别的函数去调用。

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

setInterval/setTimeout(function(){...},ms)

str.replace(/正则/ig,function(kword){return 新值})

2. 匿名函数自调:

什么是: 定义函数后，立刻执行！

为什么: 避免使用全局变量，造成全局污染

何时: 几乎所有的js代码都要装在匿名函数自调中

如何: (function(){ ... })()

作用域(scope)/作用域链: 参考2\_function\_4.pdf

闭包: 参考2\_function\_4.pptx