正则表达式

1. .基础语法

**创建一个正则表达式**

第一种方法：

var reg = /pattern/;

第二种方法：

var reg = new RegExp(“pattern”);

**正则表达式的exec方法简介**

语法：

reg.exec(str);

其中str为要执行正则表达式的目标字符串。

例如：

<script>

var reg = /test/;

var str = “testString”;

var result = reg.exec(str);

alert(result);

</script>

将会输出test，因为正则表达式reg会匹配str中的‘test’子字符串，并且将其返回。

我们使用下面的函数来做匹配正则的练习：

function execReg(reg,str){

var result = reg.exec(str);

alert(result);

}

函数接收一个正则表达式参数reg和一个目标字符串参数str，执行之后会alert出正则表达式与字符串的匹配结果。

用这个函数则是上面的例子就是:

<script>

function execReg(reg,str){

var result = reg.exec(str);

alert(result);

}

var reg = /test/;

var str = “testString”;

execReg(reg,str);

</script>

上面的例子用正则里的test去匹配字符串里的test，实在是很无聊，同样的任务用indexOf()方法就可以完成了，用正则，自是

**一片两片三四片，落尽正则全不见**

上面的小标题翻译成正则就是{1}，{2}，{3,4}，{1，}。

c{n}

{1}表示一个的意思。

/c{1}/只能匹配一个c。

/c{2}/则会匹配两个连续的c。

以此类推，

/c{n}/则会匹配n个连续的c。

看下面的例子：

reg = /c{1}/;

str = ‘cainiao’;

execReg(reg,str);

返回结果c

reg = /c{2}/;

str = ‘cainiao’;

execReg(reg,str);

返回结果null,表示没有匹配成功。

reg = /c{2}/;

str = ‘ccVC果冻爽’;

execReg(reg,str);

返回结果cc。

c{m,n}

c{3,4}的意思是，连续的3个c或者4个c。

例如

reg = /c{3,4}/;

str = ‘ccVC果冻爽’;

execReg(reg,str);

返回结果null，表示没有匹配成功。

reg = /c{3,4}/;

str = ‘cccTest’;

execReg(reg,str);

返回结果ccc。

reg = /c{3,4}/;

str = ‘ccccTest’;

execReg(reg,str);

返回结果cccc，这表明正则会尽量多匹配，可3可4的时候它会选择多匹配一个。

reg = /c{3,4}/;

str = ‘cccccTest’; //前四个c

execReg(reg,str);

仍然只匹配4个c。

由以上例子可以推断出，c{m,n}表示m个到n个c，且m小于等于n。

c{n,}

c{1,}表示1个以上的c。例如：

reg = /c{1，}/;

str = ‘cainiao’;

execReg(reg,str);

返回结果c。

reg = /c{1,}/;

str = ‘cccccTest’;

execReg(reg,str);

返回结果ccccc，再次说明了正则表达式会尽量多的匹配。

reg = /c{2,}/;

str = ‘cainiao’;

execReg(reg,str);

返回结果null，c{2,}表示2个以上的c，而cainiao中只有1个c。

由以上例子可知，c{n,}表示最少n个c，最多则不限个数。

\*，+，？ //量词

\*表示0次或者多次，等同于{0，}，即

c\* 和c{0，}是同一个意思。

+表示一次或者多次，等同于{1，}，即

c+和c{1，}是同一个意思。

？表示0次或者1次，等同于{0，1}，即

c？和c{0，1}是同一个意思。

贪心（贪婪）与非贪心（非贪婪）

人都是贪婪的，正则也是如此。我们在例子reg=/c{3,4}/;str=’ccccTest’中已经看到了，能匹配4个c，上面所介绍的所有正则都是这样，只要在合法的情况下，它们会尽量多去匹配字符，这就叫贪心（贪婪），如果我们希望正则尽量少的匹配字符，那么就可以在表示数字的符号后面加上一个?。组成如下的形式：{n,}?, \*?, +?, ??, {m,n}? //非贪婪

同样来看一个例子：

reg = /c{1,}?/;

str = ‘ccccc’;

execReg(reg,str);

返回结果只有1个c，尽管有5个c可以匹配，但是由于正则表达式时非贪心模式，所以只会匹配一个。

**/^开头,结尾$/**

^表示值匹配字符串的开头。看下面的例子：

reg = /^c/;

str = ‘维生素c’;

execReg(reg,str);

返回结果null，因为字符串“维生素c”的开头不是c，所以匹配失败。

reg = /^c/;

str = ‘cainiao’;

execReg(reg,str);

返回结果c，匹配成功，因为’cainiao’恰恰是c开头的。

与^相反，$则只匹配字符串结尾的字符，同样，看例子：