正则表达式由普通字符(a~z)以及特殊字符(称为元字符)组成的文字模式。

语法：/匹配对象的模式/

位于“/”定界符之间的部分就是将要在目标对象中进行匹配的模式。

定位符“^”的使用

var reg\_expression=/^accel wor/; //注意不要留多余的空格，否则会出错

var textString="accel world";

var result=reg\_expression.test(textString);

document.write(result+"<br>");

if(result)

{

document.write("正则表达式"+reg\_expression+"匹配字符串\""+textString+"\".<br>");

} //转义字符要放在双引号里面才行

else

alert("未找到匹配的模式！");

匹配字符$的使用，匹配必须出现在目标字符串的结尾处。

var reg\_expression=/world$/;

var textString="accel world";

var result=reg\_expression.test(textString);

document.write(result+"<br>");

if(result)

{

document.write("正则表达式"+reg\_expression+"匹配字符串\""+textString+"\".<br>");

}

else

alert("未找到匹配的模式！");

“\b”的使用：匹配一个字边界，包含了字与空格间的位置，以及目标字符串的开始和结束位置等。

var reg\_expression=/\bworld\b/; //一个"\b"代表一个边界

var textString="accel world yes";

var result=reg\_expression.test(textString);

document.write(result+"<br>");

if(result)

{

document.write("正则表达式"+reg\_expression+"匹配字符串\""+textString+"\".<br>");

}

else

alert("未找到匹配的模式！");

**特殊字符转义：**在表达式中用到的一些元字符不再表示原来的字面意义，如果要匹配这些元字符，必须使用“\”将这些字符转义为原义字符，例：匹配字面意义的“\”，需要使用“\\”表示。**贪婪匹配与非贪婪匹配**

默认正则表达式使用最长匹配原则(贪婪原则)

例如，要将“book”中匹配“bo?”的部分替换成“l”，替换后的结果是“lok”，而不是“look”;

如果将“book”中匹配“bo\*”的部分替换成“l”，替换过的结果是“lk”，而不是“lok”或“look”。

如果当字符“?”紧跟任何其他限定符(“\*”，“+”，“?”，“{n,})之后时，匹配模式会变成使用最短匹配原则(非贪婪原则)

例如：在字符串“booook”中，“bo+?”只匹配“bo”部分，而“bo+”匹配“boooo”部分。

选择匹配符：用于选择匹配两个选项之中的任意一个，其两个选项是“|”字符两边尽可能最大的表达式。

例如：“accel|worlds”匹配的是“accel”或“world”，而不是“accels”或“worlds”；如果要匹配“accels” 或“worlds”，应该使用括号创建子表达式，即“(accel|world) s”。

**RegExp对象的方法**

1.exec()方法：用正则表达式模式在字符串中运行查找，并返回包含该查找结果的一个数组。

语法：rgExp.exec( str)

rgExp：必选，包含正则表达式模式和可用标志的正则表达式对象。

str：必选，要在其中执行查找的String对象或字符串文字。

如果exec()方法没有找到匹配，则它返回null；如果它找到匹配，则exec返回一个数组，并更新全局**RegExp**对象的属性，以反映匹配结果。数组的0元素包含了完整的匹配，而第一到n元素中包含的是匹配中出现的任意一个子匹配，这相当于没有设置全局标志(g)的match()方法。

如果为正则表达式设置了全局标志，exec从以lastIndex的值指示的位置开始查找；如果没有全局标志，exec忽略lastIndex的值，从字符串的起始位置开始搜寻。

exec返回的数组有3个属性：input属性包含了整个被查找的字符串。index属性中包含了整个被查找字符串中被匹配的子字符串的位置。lastIndex属性中包含了匹配中最后一个字符的下一个位置。

实例：应用exec返回一个数组。

function RegExpTest()

{

var ver=Number(ScriptEngineMajorVersion()+"."+ScriptEngineMajorVersion());

if(ver>=5.5){ //测试js的版本

var src="I'm a good boy !";

var re=/\w+/g; // 创建正则表达式模式

var arr;

while((arr=re.exec(src))!=null)

document.write(arr.index+"-"+arr.lastIndex+arr+"\t");

}

else

{

alert("请使用js的更新版本");

}

}

document.write(RegExpTest());

</script>

目前还看不懂，以后再解决好了。

2.test()方法：返回一个Boolean值，它指出在被查找的字符串中是否存在模式。

rgexp.test( str )

实例：应用test()查询指定字符串是否存在

function td(re,s){

var s1;

//检查字符串是否存在正则表达式

if(re.test(s)){

s1="contains";

}

else{

s1="does not contain";

}

return("'"+s+"'"+s1+"'"+re.source+"'"); //返回字符串

}

document.write(td(/boy+/,"i am a good boy!")); //输出'i am a good boy!'contains'boy+'

3.match()方法：使用正则表达式模式对字符串执行查找，并将包含查找的结果作为数组返回。

stringObj.match( rgExp )

实例：match()方法来查询字符串

function md(){

var r,re;

var s="i'm a good boy";

re=/boy/i;

r=s.match(re);

return(r); //返回第一次出现“boy”的地方

}

document.write(md()); 输出boy

实例：带(g)标志设置的match()方法返回多次出现同一字符串

function md(){

var r,re;

var s="i'm a good boy,but not boys!";

re=/boy/ig;

r=s.match(re);

return(r); //返回的数组中包含了所有“boy”出现的两个匹配

}

document.write(md()); 输出boy,boy

**4.search()方法**

返回与正则表达式查找内容匹配的第一个子字符串的位置

stringObj.search( rgExp )

实例：

function sd(){

var r,re;

var s="accel world!";

re=/world/ig;

r=s.search(re);

return(r);

}

document.write(sd()); //输出7

似乎只是返回距离查找的字符串还有多远。。。

**5.replace()方法**

使用表达式模式对字符串执行搜索，并对搜索到的内容用指定的字符串替换，返回字符串对象，包含了替换后的内容。

replace( rgExp.replaceText )

reExp：为搜索时要使用的表达式对象，如果是字符串，则不按正则表达式的方法进行模糊搜索，而进行精确搜索。

replaceText：为用于替换搜索到的内容的字符串，其中可以使用一些特殊的字符组合来表示匹配变量。其中，“$&”是整个表达式模式在被搜索字符串中所匹配的字符串，“$”是表达式模式在被搜索字符串中所匹配的字符串左边的所有内容，“$’”是表达式模式在被搜索字符串中所匹配的字符串右边的所有内容，“$$”则是普通意义的“$”字符。

<script type="text/javascript">

var strsrc="ab12cd34ef56";

var re=/(\d)(\d)/gi;

var strdest=strsrc.replace(re,"$2$1");

alert("字符串"+strsrc+"被转换为："+strdest);

</script>

返回ab21cd43ef65。

**6.split()方法**

返回按照某种分隔标志符将一个字符串拆成若干个子字符串时所产生的子字符串数组。

split( [ separator [ , limit ] ] )

<script type="text/javascript">

var splitarray=new Array();

var string="javascript、asp、jsp、java";

var regex=/、/;

splitarray=string.split(regex);

for(i=0;i<splitarray.length;i++)

{

document.write(splitarray[i]+" ");

}

</script>

输出javascript asp jsp java

触发onerror事件处理异常

windows与图像对象都可触发

<script type="text/javascript">

window.onerror=function(ms,Url,Line)

{

alert("调用的函数不存在！\n"+ms+"\n"+Url+"\n"+Line+"\n");

return true;

}

function imgload(){

document.images[0].onerror=function()

{

alert("调用的图像不存在！");

}

document.images[0].src="test233.gif";

}

onHave();

</script>

</head>

<body onload="imgload()">

</body>