2015年4月21日 星期二

Lab7

DeadLine: 2015.4.23 23:59:59

1. 正则表达式

正则表达式是一种文本模式，包括普通字符（例如，a到z之间的字母）和特殊字母（称为“元字符”）。模式描述在搜索文本时要匹配的一个或者多个字符串。

下面是正则表达式的一些示例:

表达式	匹配
/^\s*$/	匹配空行
/\d{2}-\d{5}/	验证由两位数字、一个连字符再加 5 位数字组成的 ID 号,如 12-34567。
/<\s*(\S+)(\s[^>]*)?>[\s \S]*< \s*\ /\1 \s*> /	匹配 HTML 标记。

下表包含了元字符的完整列表以及他们在正则表达式上下文中的行为:

字符	说明
\	将下一字符标记为特殊字符、文本、反向引用或八进制转义符。例如,“n”匹配字 符“n”。“\n”匹配换行符。序列“\\”匹 配“\”,“\(”匹 配“(”。
^	匹配输入字符串开始的位置。如果设置了 RegExp 对象的 Multiline 属性,^还 会与“\n”或“\r”之后的位置匹配。
$	匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性,$还会 与“\n”或“\r”之前的位置匹配。
*	零次或多次匹配前面的字符或子表达式。例如,zo* 匹配“z”和“zoo”。* 等效于 {0,}。
+	一次或多次匹配前面的字符或子表达式。例如,“zo+”与“zo”和“zoo”匹配,但与“z” 不匹配。+ 等效于 {1,}。
?	零次或一次匹配前面的字符或子表达式。例如,“do(es)?”匹配“do”或“does”中的 “do”。? 等效于 {0,1}。
{n}	n 是非负整数。正好匹配 n 次。例如,“o{2}”与“Bob”中的“o”不匹配,但与“food” 中的两个“o”匹配。

{n,}	n 是非负整数。至少匹配 n 次。例如,“o{2,}”不匹配“Bob”中的“o”,而匹配“foooood”中的所有 o。“o{1,}”等效于“o+”。“o{0,}”等效于“o*”。
{n,m}	M 和 n 是非负整数,其中 n <= m。匹配至少 n 次,至多 m 次。例如,“o{1,3}”匹配“fooooood”中的头三个 o。'o{0,1}' 等效于 'o?'。注意:您不能将空格插入逗 号和数字之间。
?	当此字符 紧随任 何其他 限定符(*、 + 、 ?、{ n} 、{n,} 、{n, m})之后时 ,匹配模式 是“非 贪心的”。“非贪心的”模式匹配搜索到的、尽可能短的字符串,而默认的“贪心的” 模式匹配搜索到的、尽可能长的字符串。例如,在字符串“oooo”中,“o+?”只匹配 单个“o”,而“o+”匹配所有“o”。
.	匹配除“\n”之外的任何单个字符。若 要匹配 包括“\ n”在内 的任意 字符, 请使 用诸如 “[\s\S]”之类的模式。
(pattern)	匹配 pattern 并捕获该匹配的子表达式。可以使用 $0...$9 属性从结果“匹配”集合 中检索捕获的匹配。若要匹配括号字符 ( ),请使用“\(”或者“\)”。
(?:pattern)	匹配 pattern 但不捕获该匹配的子表达式,即它是一个非捕获匹配,不存储供以 后使用的匹配。这对于用“or”字符 (|) 组合模式部件的情况很有用。例如, 'industr(?:ylies) 是比 'industrylindustries' 更经济的表达式。
(?=pattern)	执行正向预测先行搜索的子表达式,该表达式匹配处于匹配 pattern 的字符串的 起始点的字符串。它是一个非捕获匹 配,即 不能捕 获供以 后使 用的匹 配。例 如, 'Windows (?=95|98|NT|2000)' 匹配“Windows 2000”中“Windows”,但不匹配 “Windows 3.1”中的“Windows”。预测先行不占用字符,即发生匹配后,下一匹配 的搜索紧随上一匹配之后,而不是在 组成预 测先行 的字符 后。
(?!pattern)	执行反向预测先行搜索的子表达式,该表达式匹配不处于匹配 pattern 的字符串 的起始点的搜索字符串。它是一个非捕获匹配,即不能捕获供以后使用的匹配。例 如,'Windows (?!95|98|NT|2000)' 匹配“Windows 3.1”中的“Windows”,但不匹配 “Windows 2000”中的“Windows”。预测先行不占用字符,即发生匹配后,下一匹配 的搜索紧随上一匹配之后,而不是在 组成预 测先行 的字符 后。
x|y	匹配 x 或 y。例如,'z|food' 匹配“z”或“food”。'(z|f)ood' 匹配“zood”或“food”
[xyz]	字符集。匹配包含的任一字符。例如,“[abc]”匹配“plain”中的“a”。
[^xyz]	反向字符集。匹配未包含的任何字符。例如,“[^abc]”匹配“plain”中的“p”。
[a-z]	字符范围。匹配指定范围内的任何字符。例如,“[a-z]”匹配“a”到“z”范围内的任何 小写字母。

[^a-z]	反向范围 字符。匹 配不在 指定的 范围内 的任何 字符。例 如,“[^a-z]”匹配任何 不在“a” 到“z”范围内的任何字符。
\b	匹配一个字边界,即字与空格间的位置。例如,“er\b”匹配“never”中的“er”,但不 匹配“verb”中的“er”。
\B	非字边界匹配。“er\B”匹配“verb”中的“er”,但不匹配“never”中的“er”。
\cx	匹配 x 指示的控制字符。例如,\cM 匹配 Control-M 或回车符。x 的值必须在 A-Z 或 a-z 之间。如果不是这样,则假定 c 就是“c”字符本身。
\d	数字字符匹配。等效于 [0-9]。
\D	非数字字符匹配。等效于 [^0-9]。
\f	换页符匹配。等效于 \x0c 和 \cL。
\n	换行符匹配。等效于 \x0a 和 \cJ。
\r	匹配一个回车符。等效于 \x0d 和 \cM。
\s	匹配任何空白字符,包括空格、制表符、换页符等。与 [\f\n\r\t\v] 等效。
\S	匹配任何非空白字符。与 [^ \f\n\r\t\v] 等效。
\t	制表符匹配。与 \x09 和 \cI 等效。
\v	垂直制表符匹配。与 \x0b 和 \cK 等效。
\w	匹配任何字类字符,包括下划线。与“[A-Za-z0-9_]”等效。
\W	与任何非单词字符匹配。与“[^A-Za-z0-9_]”等效。
\xn	匹配 n,此处的 n 是一个十六进制转义码。十六进制转义码必须正好是两位数长。 例如,“\x41”匹配“A”。“\x041”与“\x04”&“1”等效。允许在正则表达式中使用 ASCII 码
\num	匹配 num,此处的 num 是一个正整数。到捕获匹配的反向引用。例如,“(.) \1” 匹配两个连续的相同字符。
\n	标识一个八进制转义码或反向引用。 如果 \n 前面至少有 n 个捕获子表达式,那么 n 是反向引用。否则,如果 n 是八进制数 (0-7),那么 n 是八进制转义码。
\nm	标识一个八进制转义码或反向引用。如果 \nm 前面至少有 nm 个捕获子表达式, 那么 nm 是反向引用。如果 \nm 前面至少有 n 个捕获,则 n 是反向引用,后 面跟有字符 m。如果两种前面的情况都不存在,则 \nm 匹配八进制值 nm,其中 n 和 m 是八进制数字 (0-7)。

\nml	当 n 是八进制数 (0-3),m 和 l 是八进制数 (0-7) 时,匹配八进制转义码 nml。
\un	匹配 n,其中 n 是以四位十六进制数表示的 Unicode 字符。例如,\u00A9 匹配 版权符号 (©);中文的范围为\u4E00-\u9FA5。

据上述规则,匹配 abc.com 邮箱的正则表达式就是以“@abc.com”为后缀的字符串,并且该后缀前面只能出现英文字母和数字。 你看看正确的判断邮箱格式的正则表达式是?

正则表达式修饰符

修饰符可以在全局搜索中不区分大小写:

修饰符	描述
i	执行对大小写不敏感的匹配。
g	执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。
m	执行多行匹配。

正则表达式的使用:

正则表达式的创建，有2种方法:

（1）var regex = /[a-z]+/g;
（2）var regex = new(“[a-z]+”，“g”);

主要方法:

(1) exec(string),对 string 进行正则处理,并返回匹配结果.

(2) test(string),测试 string 是否含有匹配结果

(3) match(pattern) 根据 pattern 进行正则匹配,如果匹配到,返回匹配结果,如匹配不到返回null

(4) search(pattern) 根据 pattern 进行正则匹配,如果匹配到一个结果,则返回它的索引数;否则返回-1

(5) replace(pattern,replacement) 根据 pattern 进行正则匹配,把匹配结果替换为 replacement

(6) split(pattern) 根据 pattern 进行正则分割,返回一个分割的数组

实例:使用正则表达式表达对象(test 的方法):

```
var mailFormat=”邮箱正则表达式”;
if(!mailFormat.test(mailValue)){
        errorMesg.innerHTML="邮箱格式错误";
}
```

2. 实验Part1

完善lab7_1.js, 使其支持利用正则表达式检查文本框输入的内容的形式。具体要完成的应该包括

1) 检查是否全为数字
2) 检查是否以字母开头、可带数字、"_"的字串
3) 检查是否符合邮箱格式
4) 检查是否全为中文
5) 如不为以上格式,则显示为其他格式

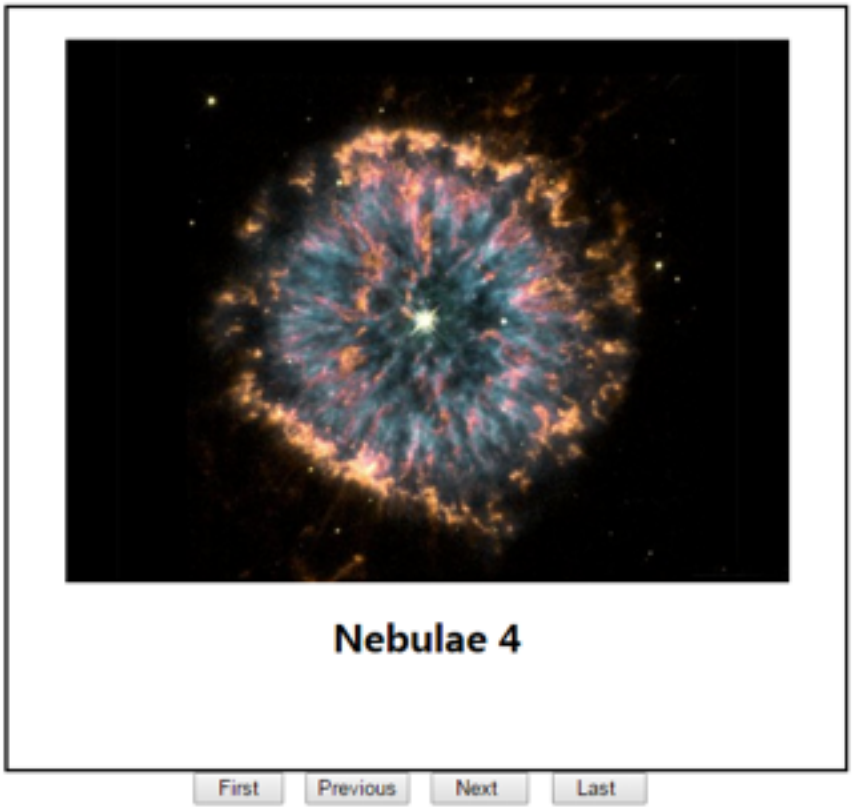
参考测试用例:

用例	显示
12345	全为数字
123ab	其他格式
abc123	以字母开头、可带数字、“_”的字符串
abc_123	以字母开头、可带数字、“_”的字符串
123abc@126.com	邮箱地址
11302010036@fudan.edu.cn	邮箱地址
123abc@126	其他格式
正则表达式	全为中文
正则表达式123	其他格式
-abc-acb	其他格式
…	…

3. 实验Part2

完成“slide.js”缺少的代码部分,实现一个在线图片查看器,可以点击 4 个按钮查看, 分别是“First”:点击这个按钮查看第一张图片。 “Previous”:点击这个按钮查看前一张图片, 不能超前第一张。 “Next”:点击这个按钮查看下一张图片,不能超过最后一张。 “Last”:点击查看最后一张图片。

效果如图:



关于图片切换时平滑效果的原理以及实现，如果感兴趣可课后自行Google，本次lab不对图片切换时的效果做出要求，只要能切换即可。