

test() 方法

`test()` 方法用于检测一个字符串是否匹配某个模式。

如果字符串中有匹配的值返回 `true`，否则返回 `false`

参数	描述
string	必需。要检测的字符串。

```
1  <script>
2      var str="Hello world!";
3      // 正则，查找是否有 hello
4      var patt=/Hello/g;
5      // test() 方法用于检测一个字符串是否匹配某个模式。匹配返回ture，不匹配返回false
6      var result = patt.test(str);
7      console.log(result);
8  </script>
```

正则表达式

作用:使用指定的规则验证字符串的格式是否正确,多用于表单验证

```
1  var 标识符 = /正则表达式/修饰符
2
3  正则表达式:验证字符串的规则
4
5  修饰符(匹配规则,可选)
6  1.不区分大小写 i
7  2.全局匹配 g
```

例子

```
1  zhengze = /Hello/g;
```

字符组 []

字符组 (`[]`) 允许匹配一组可能出现的字符。

表达式

```
/[Pp]ython/g
```

文本

```
I like Python3 and I like python2.7  
人生苦短我用Python  
python
```

区间 [0-9]、[a-z]、[A-Z]

如果要匹配从 a-z 的字母呢？我想你肯定不愿意从 a 写到 z 了！

为了适应这一点，正则表达式引擎在字符组中使用**连字符 (-)** 代表**区间**，依照这个规则，我们可以总结出三点：

1. 要匹配任意数字可以使用 [0-9]；
2. 如果想要匹配所有小写字母，可以写成 [a-z]；
3. 想要匹配所有大写字母可以写成 [A-Z]。

例子

表达式

```
/[0-9]/g
```

文本

```
python3  
0123456789  
ruby5  
java8
```

匹配特殊字符 \-

正则表达使用了 - 号代表了**区间**，但是我们有时候需要匹配的符号就是 - 号，该怎么办呢？

这个时候我们需要对 - 号进行**转义**操作，即 \-。

在正则中使用 \ 就可以进行对特殊符号进行转义，对 - 进行转义就可以表示为 \-，即 \- 就代表了 - 号本身。

例子

表达式
<code>/[0-9\-]/g</code>
文本
<pre>0731-8848↵ code-jiaonang↵ 0-master-123</pre>

取反 ^

通过在字符数组开头使用 `^` 字符实现取反操作，从而可以反转一个字符组（意味着会匹配任何指定字符之外的所有字符）。

例子：匹配不包含数字的字符组

表达式
<code>/[^0-9]/g</code>
文本
<pre>code123↵ python456↵ java78910↵ ruby546789</pre>

可以通过在字符数组开头使用 `^` 字符实现取反操作，从而可以反转一个字符组（意味着会匹配任何指定字符之外的所有字符）。

例子：这里的 `n[^e]` 的意思就是 `n` 后面的字母不能为 `e`。

表达式
<code>/n[^e]/g</code>
文本
<pre>python3↵ final↵ ne↵</pre>

快捷匹配数字和 \d 字母 \w

正则表达式引擎提供了一些快捷方式如：\w 可以与任意单词字符匹配。

当我们想要匹配任意数字的时候也可以使用快捷方式 \d，d 即 digit 数字的意思，等价于 [0-9]

快捷方式	描述
\w	与任意单词字符匹配，任意单词字符表示 [A-Z]、[a-z]、[0-9]、_
\d	与任意数字匹配

匹配空白 \s

\s 快捷方式可以匹配空白字符，比如空格，tab、换行等。

单词边界 \b

\b 匹配的是单词的边界

例子：\bmaster\b 就仅匹配有边界的 master 单词。

表达式
<code>/\bmaster\b/g</code>
文本
<code>amastercode-master-xxx-master-mastera- abc-master-abc"master"</code>

当然其他类型的数据，比如数字也能匹配：

表达式
<code>/\b\d\b/g</code>
文本
<code>1,2,3a456 1→2→3456</code>

快捷方式取反 \w、\D

快捷方式也可以取反，例如对于 \w 的取反为 \W，将小写改写成大写即可，其他快捷方式也遵循这个规则

开始 ^ 和结束 \$

正则表达式中 ^ 指定的是一个字符串的开始，\$ 指定的是一个字符串的结束。

例如：

表达式
<code>/^python/g</code>
文本
<code>python is my favorite this code in python</code>

指定字符串的结束：

表达式
<code>/python\$/g</code>
文本
<code>python is my favorite this code in python</code>

任意字符 .

. 字符代表匹配任何单个字符，它只能出现在方括号以外。

注意：. 字符只有一个不能匹配的字符，也就是换行符（\n）

例子：

表达式
<code>/a..g</code>
文本
<pre> aii.all.hello- aim.ak47.all </pre>

可选字符？

有时，我们可能想要匹配一个单词的不同写法，比如 `color` 和 `colour`，或者 `honor` 与 `honour`

这个时候我们可以使用 `?` 符号指定一个字符、字符组或其他基本单元可选，这意味着正则表达式引擎将会期望该字符出现**零次或一次**。

例子

表达式
<code>/honou?r/g</code>
文本
<pre> He.Served.with.honour.and.distinction- He.Served.with.honor.and.distinction- Served.with.honouur.and.distinction </pre>

重复 {N}

到目前为止，我们只是学习了关于仅出现一次的字符串匹配，在实际开发中，肯定不能满足需求，比如要匹配电话号码、身份证的时候就无法满足需求了。

如果遇到这样的情况，我们可能期望一个字符组连续匹配好几次。

在一个字符组后加上 `{N}` 就可以表示在它之前的字符组出现 `N` 次。

例子

表达式
<code>/\d{4}-\d{7}/g</code>
文本
张三: 0731-8825951 李四: 0733-8794561

重复区间 {M,N}

可能有时候，我们不知道具体要匹配字符组要重复的次数，比如身份证有 15 位也有 18 位的。

这里重复区间就可以出场了，语法：`{M,N}`，`M` 是下限而 `N` 是上限。

例子

表达式
<code>/\d{3,4}/g</code>
文本
073 0731

`\d{3,4}` 既可以匹配 3 个数字也可以匹配 4 个数字，不过当有 4 个数字的时候，优先匹配的是 4 个数字，这是因为正则表达式默认是**贪婪模式**，即尽可能的匹配更多字符，而要使用**非贪婪模式**，我们要在**表达式后面加上 `?` 号**。

例子

表达式
<code>/\d{3,4}?/g</code>
文本
073 0731

开闭区间

有时候我们可能遇到字符组的重复次数没有边界，例如：

表达式
<code>/\d{1,}/g</code>
文本
<code>073</code>
<code>0731</code>
<code>123456789</code>

闭区间不写即可表示匹配一个或无数个

速写 `+` 等价于 `{1,}` `*` 等价于 `{0,}`

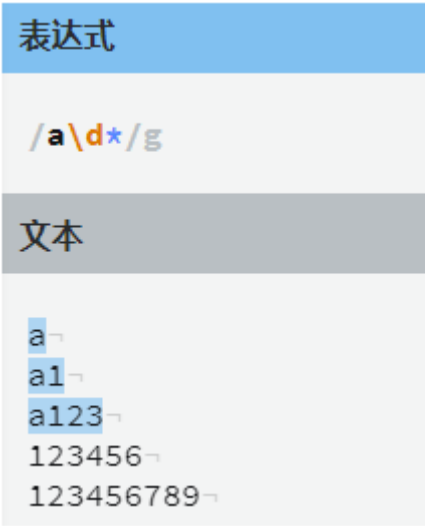
还可以使用两个速写字符指定常见的重复情况，可以使用 `+` 匹配 1 个到无数个，使用 `*` 代表 0 个到无数个。

即：`+` 等价于 `{1,}`，`*` 等价于 `{0,}`。

`+` 号示例：

表达式
<code>/a\d+/g</code>
文本
<code>a</code>
<code>a1</code>
<code>a123</code>
<code>123456</code>
<code>123456789</code>

`*` 号示例：



例子：匹配所有手机号码

现在请你使用正则表达式匹配手机号码，假设手机号码规则如下：

- 必须是 11 位的数字；
- 第一位数字必须以 1 开头，第二位数字可以是 [3,4,5,7,8] 中的任意一个，后面 9 个数是 [0-9] 中的任意一个数字。
- 1 | ^1[3,4,5,7,8]\d{9}

正则表达式总结

多种匹配模式

实例	描述
[Pp]ython	匹配 “Python” 或 “python”。
rub[ye]	匹配 “ruby” 或 “rube”。
[abcdef]	匹配中括号内的任意一个字母。
[0-9]	匹配任何数字。类似于 [0123456789]。
[a-z]	匹配任何小写字母。
[A-Z]	匹配任何大写字母。
[a-zA-Z0-9]	匹配任何字母及数字。
[^au]	除了au字母以外的所有字符。
[^0-9]	匹配除了数字外的字符。

实例	描述
.	匹配除“\n”之外的任何单个字符。要匹配包括‘\n’在内的任何字符，请使用象‘[\n]’的模式。
?	匹配一个字符零次或一次，另一个作用是非贪婪模式
+	匹配1次或多次
*	匹配0次或多次
\b	匹配一个长度为 0 的子串
\d	匹配一个数字字符。等价于 [0-9]。
\D	匹配一个非数字字符。等价于 [^0-9]。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于 [^\f\n\r\t\v]。
\w	匹配包括下划线的任何单词字符。等价于‘[A-Za-z0-9_]’。
\W	匹配任何非单词字符。等价于 ‘[^A-Za-z0-9_]’。

form表单初体验

例子

```

1      <!-- 必须需要return才可以实现阻拦form表单跳转 -->
2      <form action="success.html" method="get" onsubmit="return a()">
3          账号:<input type="text" id="input_text" onblur="zh()" value="Hello
world!">
4          <span id="span1"></span>
5          <br>
6          密码:<input type="password" onblur="ps()" id="input_ps">
7          <span id="span2"></span>
8          <br>
9          <input type="submit" value="提交按钮" id="sub">
10     </form>
11     <script>
12         // form表单提交的函数
13         function a(){
14             // 中间使用 & 相连接
15             if(zh()&ps()){
16                 alert('成功');
17                 return true;
18             }else{
19                 alert('失败');
20                 return false;
21             }
22         }
23     
```

```

24 // 账号验证
25 function zh(){
26     var zhValue = document.getElementById('input_text').value;
27     var span1 = document.getElementById('span1');
28     var zhegnze = /Hello/g;
29     if(zhegnze.test(zhValue)){
30         span1.innerHTML = "<font color='green'>√</font>";
31         return true;
32     }else{
33         span1.innerHTML = "<font color='red'>请输入和Hello相匹配的字符
34     </font>";
35         return false;
36     }
37 }
38 // 密码验证
39 function ps(){
40     var zhValue = document.getElementById('input_ps').value;
41     var span1 = document.getElementById('span2');
42     var zhegnze = /Hello/g;
43     if(zhegnze.test(zhValue)){
44         span1.innerHTML = "<font color='green'>√</font>";
45         return true;
46     }else{
47         span1.innerHTML = "<font color='red'>请输入和Hello相匹配的字符
48     </font>";
49         return false;
50     }
51 }
52 </script>

```

全选全不选反选（例子）

```

1 <form action="">
2     <input type="button" value="全选" id="btn1">
3     <input type="button" value="全不选" id="btn2">
4     <input type="button" value="反选" id="btn3">
5     <br>
6     HTML<input type="checkbox" name="a" id="">
7     CSS<input type="checkbox" name="a" id="">
8     JavaScript<input type="checkbox" name="a" id="">
9     jquery<input type="checkbox" name="a" id="" >
10 </form>
11
12 <script>
13     // 全选
14     var btn1 = document.getElementById('btn1');
15     btn1.onclick = function(){
16         var che = document.getElementsByName('a');
17         for(var i = 0;i<che.length;i++){
18             che[i].checked = true;
19         }
20     }
21     // 全不选

```

```
22     var btn2 = document.getElementById('btn2');
23     btn2.onclick = function(){
24         var che = document.getElementsByName('a');
25         for(var i = 0;i<che.length;i++){
26             che[i].checked = false;
27         }
28     }
29     // 反选
30     var btn3 = document.getElementById('btn3');
31     btn3.onclick = function(){
32         var che = document.getElementsByName('a');
33         for(var i = 0;i<che.length;i++){
34             che[i].checked = !che[i].checked;
35         }
36     }
37     </script>
```