### Regular expression：找什么，找多少，从哪里找

正则表达式通过一类特殊的字符，我们称为元字符或通配符(meta character)来进行字符的模糊表示，它们在匹配的过程中代表特殊的含义。如果要表示原义，就要用 **\** 进行转义，去除它的特殊含义。

**\**转义字符

|  |  |  |  |
| --- | --- | --- | --- |
| **\**\ | \ | **\**. | . |
| **\**(**\**) | () | **\**[**\**] | [] |

|  |  |  |
| --- | --- | --- |
|  |  | **含义** |
| 基本元字符 | . | 表示任一个非换行的字符，[\s\S] 也包括换行 |
| () | 表示分组和提高优先级 |
| [] | 表示一个字符，如 [abc] 匹配a/b/c |
| | | 或（优先级最低）用法：正则表达式 | 正则表达式 |
| 次数限定元字符 | \* | 紧挨前面的一个字符或一组字符出现几次都行 |
| + | 紧挨它前面的字符出现 1 次到多次 |
| ？ | 紧挨它前面的字符出现 0 次或1次，等价于（a|） |
| {n} | 紧挨它前面的字符出现n次 |
| {n, m} | 紧挨它前面的字符出现n~m次 |
| 首尾元字符 | ˆ | 以某字符开头 |
| $ | 以某字符结尾 |
| 否定元字符 | [^字符]  hat | 出现在第一位代表“以...开头”，其他位置都代表“非” |

1. { } 中的量词是自定义出现次数，修饰的是紧挨着的字符
2. 引用通配符：反斜杠加上数字是所谓back reference: \1 \2 \3 等，它的作用是引用前面的某个括号元组，如:  "twoonwo" =~ /t(wo)on\1/   为true

|  |  |  |  |
| --- | --- | --- | --- |
|  | **单词** | **代表** | **含义** |
| [^\n\r] | newline  return |  | 除了换行和回车之外的任意字符 |
| \d | digit | [0-9] | 一个数字 |
| \D |  | [^0-9] | 非数字字符 |
| \w | word | [a-zA-Z0-9\_] | 一个单词字符(所有的数字、字母、下划线) |
| \W |  | [^a-zA-Z0-9\_] | 非单词字符 |
| \s | space | [\f\r\n\t\v] | 不可见字符：空格 / tab / enter |
| \S |  | [^\f\r\n\t\v] | 可见字符 |

方法：

|  |  |
| --- | --- |
| test() | 用于检测一个字符串是否匹配某个模式  RegExpObject.test(string) |
| exec() | 用于检索字符串中的正则表达式的匹配  RegExpObject.exec(string)  返回一个数组，其中存放匹配的结果，如果未找到则返回null |
| replace() | 在1中找到2, 将其替换成3, 返回替换后的字符串。特点: 只替换第一个找到的字符串 |

/(\d+)(\s)/.exec("ab 12 ce 34 ed 55");

["12 ", "12", " "]

/((\d+)(\s))/.exec("ab 12 ce 34 ed 55");

["12 ", "12 ", "12", " "]

正则提取时，数组的第0项永远是整个正则匹配的结果，从第1项开始，分别是每一个小括号代表的匹配结果，有几个括号就有几个结果。

/((\d+)(**?:**\s))/.exec("ab 12 ce 34 ed 55");

["12 ", "12 ", "12"]

匹配但不捕获，也就是说带“**？：**”的括号匹配到的内容不会放到数组里面。

var s = "暖";

var reg = new RegExp("(" + s + ")", "g");

var str = "CSDN暖枫无敌，暖枫无敌CSDN";

var newstr = str.replace(reg, "<span style='color:green'>$1</span>");

document.write(newstr + "<br />");