**{ 正则表达式 }**

var reg1 = // 正则写法

\ 转义符 \b 单词边界

正则表达式－获取控制 量词：

n+ : 匹配任何包含至少一个 n 的字符串。{1,}

n? : 匹配任何包含零个或一个 n 的字符串。{0,1}

n\* : 匹配任何包含零个或多个 n 的字符串。{0,}

n{X} : 匹配包含 X 个 n 的序列的字符串。

n{X,Y} : 匹配包含 X 到Y 个 n 的序列的字符串。（y可以不写）

^n : 匹配任何开头为 n 的字符串。

n$ : 匹配任何结尾为 n 的字符串。

开头

g全局 / i不区分大小写

邮箱验证：**var reg1 = /^\w+@\w+(\.[a-zA-Z]{2,3}){1,2}$/g**

匹配[ ]内内容

结尾

正则表达式－转义字符：

\s : 匹配任意一个空白字符

\S：匹配非空格

\d : 匹配任意一个数字，0~9 中的任意一个

\D : 匹配任意一个非数字

\w : 匹配任意一个字符( 字母、 数字、下划线 )

\W：匹配非字母、 数字、下划线

\1：表示重复子项（重复的内容必须一样）

\n : 查找换行符

|  |  |
| --- | --- |
| 正则测试方法 | 返回值 |
| reg1.test( str ) | true / false |
| reg1.exec( str ) | 数组 / 没匹配到返回null |
| str.search( reg1 ) | Index / -1 |
| str.match( reg1 ) | 数组 / null |
| str.replace( reg1, function (s1, s2) {} ) | s1匹配项，s2是s1子项，拿s1再进行匹配 |