**正则**

一般用来匹配查找字符串（模糊范围）

一种标准的写法：**new RegExp(str,修饰符)**  类似于new Array ....接收的是字符串。

另一种为简洁写法： /2/

正则中写入的是某套规则

**正则方法**

**test（）**: 语法： **正则对象.test(str)**; 返回的是布尔值。 作用：查找字符串中是否有满足正则的字符串，有返回true，反之false

**search（）:**  语法： **str . search(正则||str);** 返回查找字符串中某个字符串首次出现的位置，找不到就返回-1。

和字符串中的indexOf类似 区别：indexOf中只能接受字符串，search中不但能接受字符串还能接受正则。

**match（）:** 语法： **str.match(正则||str);** 找到匹配到的数据，将数据存入数组并返回。如果找到了返回的是一个数组，如果没找到返回一个null。

**replace();** 语法：**str.replace(查找字符串||正则,替换成的字符串||fn);** 找到要替换的内容，然后替换。

fn里的形参，第一个是查找到的字符串，从第二个开始都是子项**。**

**子项**：() 从第一个参数往后（从第二个参数起）都是子项 从左往右依次查找

**其他规则：**

**量词：**

**+**  最小一个，最多不限

**?** 前面的可有可无，最少0次，最多1次

**\*** 最小0次最多无限

**{n,m}** n:最小次数 m:最大次数 ；  **{n,}** 最小次数 最多不限，**{0,1}**和?一样；**{0,}**  \*最小0次最多无限

注意：量词只适用它前面的字符 a\d+ (a\d)+

**其他字符：**

**^** 在正则起始位置加^代表开始;如果在\*[]\*中用^ 那么意思是排除[]中的字符

**$** 在正则结尾位置加$代表的是结束

**()** 1.提权 2.每加一个括号就会多匹配一次

**.**  任意字符（除了\n）

**[]** 例：[abc] 可以是a或者可以是b或者可以是c 例[1-3])?[1-9] 可以是11到39

**[\u4e00-\u9fa5]**匹配中文

**|** 正则里的或者

**修饰符：**在正则的结尾，例：//g

**g**  global 全局的意思

**i**  不区分大小写

**转义符：**

当\ + 任意的指定字母，形成一种特殊含义的字符串。

\n 换行

\d 一个数字 \d+ 多个数字

\D 一个非数字 \D+ 多个非数字

\s 空格

\S 非空格

\w 字母、数字、下划线

\W 非字母、数字、下划线

\b 边界符 起始、结尾、空格两边

\B 非边界符

没有量词都是一个

正则的特性：

1.懒惰

叫它找一个绝对不会找多个

2.贪婪

只要有就要。