```
/*是否带有小数*/
function    isDecimal(strValue )  {
    var  objRegExp= /^\d+\.\d+$/;
   return  objRegExp.test(strValue);
}

/*校验是否中文名称组成 */
function ischina(str) {
        var reg=/^[\u4E00-\u9FA5]{2,4}$/;    /*定义验证表达式*/
        return reg.test(str);      /*进行验证*/
}

/*校验是否全由8位数字组成 */
function isStudentNo(str) {
        var reg=/^[0-9]{8}$/;    /*定义验证表达式*/
        return reg.test(str);      /*进行验证*/
}

/*校验电话码格式 */
function isTelCode(str) {
        var reg= /^((0\d{2,3}-\d{7,8})|(1[3584]\d{9}))$/;
        return reg.test(str);
}

/*校验邮件地址是否合法 */
function IsEmail(str) {
        var reg=/^([a-zA-Z0-9_-])+@([a-zA-Z0-9_-])+(\.[a-zA-Z0-9_-])+/;
        return reg.test(str);
}

var reg=/test/;          //是否包含test
s.indexOf("th");        //查看角标
s.search("th");          //查看角标
s.substring(1,s.length);   //截取字符串
```

http://www.runoob.com/js/js-regexp.html

JavaScript 正则表达式

正则表达式（英语：Regular Expression，在代码中常简写为regex、regexp或RE）使用单个字符串来描述、匹配一系列符合某个句法规则的字符串搜索模式。
搜索模式可用于文本搜索和文本替换。

什么是正则表达式？

正则表达式是由一个字符序列形成的搜索模式。
当你在文本中搜索数据时，你可以用搜索模式来描述你要查询的内容。
正则表达式可以是一个简单的字符，或一个更复杂的模式。
正则表达式可用于所有文本搜索和文本替换的操作。

语法

/正则表达式主体/修饰符(可选)
其中修饰符是可选的。

实例：

```
var patt = /runoob/i
```

实例解析：
/runoob/i　是一个正则表达式。
runoob　是一个**正则表达式主体**（用于检索）。
i　是一个**修饰符**（搜索不区分大小写）。

使用字符串方法

在 JavaScript 中，正则表达式通常用于两个字符串方法 : search() 和 replace()。
search() 方法 用于检索字符串中指定的子字符串，或检索与正则表达式相匹配的子字符串，并返回子串的起始位置。
replace() 方法 用于在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子串。

search() 方法使用正则表达式

实例

使用正则表达式搜索 "Runoob" 字符串，且不区分大小写：

```
var str = "Visit Runoob!"; var n = str.search(/Runoob/i);
```

输出结果为：

6

尝试一下 »

---

search() 方法使用字符串

search 方法可使用字符串作为参数。字符串参数会转换为正则表达式：

实例

检索字符串中 "Runoob" 的子串：

```
var str = "Visit Runoob!"; var n = str.search("Runoob");
```

尝试一下 »

---

replace() 方法使用正则表达式

实例

使用正则表达式且不区分大小写将字符串中的 Microsoft 替换为 Runoob ：

```
var str = document.getElementById("demo").innerHTML; var txt =
str.replace(/microsoft/i,"Runoob");
```

结果输出为:

Visit Runoob!

尝试一下 »

replace() 方法使用字符串

replace() 方法将接收字符串作为参数：

```
var str = document.getElementById("demo").innerHTML; var txt =
str.replace("Microsoft","Runoob");
```

尝试一下 »

你注意到了吗？

	正则表达式参数可用在以上方法中 (替代字符串参数)。 正则表达式使得搜索功能更加强大(如实例中不区分大小写)。

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。

正则表达式模式

方括号用于查找某个范围内的字符：

表达式	**描述**
[abc]	查找方括号之间的任何字符。

[0-9]	查找任何从 0 至 9 的数字。
(x|y)	查找任何以 | 分隔的选项。

元字符是拥有特殊含义的字符：

元字符	**描述**
\d	查找数字。
\s	查找空白字符。
\b	匹配单词边界。
\uxxxx	查找以十六进制数 xxxx 规定的 Unicode 字符。

量词：

量词	**描述**
n+	匹配任何 包含至少一个 n 的字符串。
n*	匹配任何包含零个或多个 n 的字符串。
n?	匹配任何包含零个或一个 n 的字符串。

使用 RegExp 对象

在 JavaScript 中，RegExp 对象是一个预定义了属性和方法的正则表达式对象。

使用 test()

test() 方法是一个正则表达式方法。

test() 方法用于检测一个字符串是否匹配某个模式，如果字符串中含有匹配的文本，则返回 true，否则返回 false。

以下实例用于搜索字符串中的字符 "e"：

实例

```
var patt = /e/;
patt.test("The best things in life are free!");
```

字符串中含有 "e"，所以该实例输出为：

```
true
```

尝试一下 »

你可以不用设置正则表达式的变量，以上两行代码可以合并为一行：

```
/e/.test("The best things in life are free!")
```

使用 exec()

exec() 方法是一个正则表达式方法。

exec() 方法用于检索字符串中的正则表达式的匹配。

该函数返回一个数组，其中存放匹配的结果。如果未找到匹配，则返回值为 null。

以下实例用于搜索字符串中的字母 "e":

Example 1

```
/e/.exec("The best things in life are free!");
```

字符串中含有 "e"，所以该实例输出为:

```
e
```

尝试一下 »