替换指定字符串

```
var str2 = str.replace(/a/g, 'o');
```

此处replace的第一个参数为正则表达式，/g是全文匹配标识。

javascript中的正则表达式有两种书写方式；

1. new RegExp();可以接受变量的 第二个参数为g代表全局匹配
2. //不接受变量的//g代表全局匹配

第一种是可以接受变量的，第二种不行，因为会把//里的内容当字符串处理。

```
<script>
    var str = 'sfsffdgdsrgergsdga';
    var sRex = 'ff';
    var reg = new RegExp(sRex, 'g');
    alert(str.match(reg));
</script>
```

demo

```
$("#inpt_btn").on('click',function(){
                var msg = $("#msg").text();
                var inpt = $("#inpt").val();
                var strArr = [];
                for(var i=0;i<inpt.length;i++){
                        var temprary = inpt[i];
                        for(var j=0;j<msg.length;j++){
                                if(msg[j] == temprary){
                                        console.log(msg[j])
                                        strArr.push(msg[j]);
                                }
                        }
                }
                for(var i=0;i<strArr.length;i++){
                        var reg = new RegExp(strArr[i],'g');
                        msg = msg.replace(reg,'<span style="color:red;">'+
strArr[i] +'</span>')
                }
                $("#msg").html(msg);
```

```
})
```

多行字符串

`string1

string2

string3`

1、charCodeAt方法返回一个整数，代表指定位置字符的Unicode编码。

strObj.charCodeAt(index)

说明：

index将被处理字符的从零开始计数的编号。有效值为0到字符串长度减1的数字。

如果指定位置没有字符，将返回NaN。

例如：

```
var str = "ABC";
str.charCodeAt(0);
```

结果：65

2、fromCharCode方法从一些Unicode字符串中返回一个字符串。

String.fromCharCode([code1[,code2...]])

说明：

code1，code2...是要转换为字符串的Unicode字符串序列。如果没有参数，结果为空字符串。

例如：

```
String.fromCharCode(65,66,112);
```

结果：ABp

3、charAt方法返回指定索引位置处的字符。如果超出有效范围的索引值返回空字符串。

strObj.charAt(index)

说明：

index想得到的字符的基于零的索引。有效值是0与字符串长度减一之间的值。

例如：

```
var str = "ABC";
str.charAt(1);
```

结果：B

4、slice方法返回字符串的片段。

strObj.slice(start[,end])

说明：

start下标从0开始的strObj指定部分其实索引。如果start为负，将它作为length+start处理，此处length为字符串的长度。

end小标从0开始的strObj指定部分结束索引。如果end为负，将它作为length+end处理，此处length为字符串的长度。

例如：

012345

var str = "ABCDEF";

str.slice(2,4);

结果：CD

5、substring方法返回位于String对象中指定位置的子字符串。

strObj.substring(start,end)

说明：

start指明子字符串的起始位置，该索引从0开始起算。

end指明子字符串的结束位置，该索引从0开始起算。

substring方法使用start和end两者中的较小值作为子字符串的起始点。如果start或end为NaN或者为负数，那么将其替换为0。

例如：

012345

var str = "ABCDEF";

str.substring(2,4); // 或 str.substring(4,2);

结果：CD

6、substr方法返回一个从指定位置开始的指定长度的子字符串。 包括 start 所指的字符

strObj.substr(start[,length])

说明：

start所需的子字符串的起始位置。字符串中的第一个字符的索引为0。

length在返回的子字符串中应包括的字符个数。

例如：

012345

var str = "ABCDEF";

str.substr(2,4);

结果：CDEF

7、indexOf方法放回String对象内第一次出现子字符串位置。如果没有找到子字符串，则返回-1。

strObj.indexOf(substr[,startIndex])

说明：

substr要在String对象中查找的子字符串。

startIndex该整数值指出在String对象内开始查找的索引。如果省略，则从字符串的开始处查找。

例如：

01234567

```
var str = "ABCDECDF";
str.indexOf("CD", 1); // 由1位置从左向右查找 123...
```

结果：2

8、lastIndexOf方法返回String对象中字符串最后出现的位置。如果没有匹配到子字符串，则返回-1。

strObj.lastIndexOf(substr[,startindex])

说明：

substr要在String对象内查找的子字符串。

startindex该整数值指出在String对象内进行查找的开始索引位置。如果省略，则查找从字符串的末尾开始。

例如：

01234567

```
var str = "ABCDECDF";
str.lastIndexOf("CD",6); // 由6位置从右向左查找 ...456
```

结果：5

9、search方法返回与正则表达式查找内容匹配的第一个字符串的位置。

strObj.search(reExp)

说明：

reExp包含正则表达式模式和可用标志的正则表达式对象。

例如：

```
var str = "ABCDECDF";
str.search("CD"); // 或 str.search(/CD/i);
```

结果：2

10、concat方法返回字符串值，该值包含了两个或多个提供的字符串的连接。

str.concat([string1[,string2...]])

说明：

string1，string2要和所有其他指定的字符串进行连接的String对象或文字。

例如：

var str = "ABCDEF";

str.concat("ABCDEF","ABC");

结果：ABCDEFABCDEFABC

11、将一个字符串分割为子字符串，然后将结果作为字符串数组返回。

strObj.split([separator[,limit]])

说明：

separator字符串或 正则表达式 对象，它标识了分隔字符串时使用的是一个还是多个字符。如果忽略该选项，返回包含整个字符串的单一元素数组。

limit该值用来限制返回数组中的元素个数。

例如：

var str = "AA BB CC DD EE FF";

alert(str.split(" ", 3));

结果：

AA,BB,CC

12、toLowerCase方法返回一个字符串，该字符串中的字母被转换成小写。

例如：

var str = "ABCabc";

str.toLowerCase();

结果：abcabc

13、toUpperCase方法返回一个字符串，该字符串中的所有字母都被转换为大写字母。

例如：

var str = "ABCabc";

str.toUpperCase();

结果：ABCABC