String字符串处理参考网址：

http://www.jb51.net/article/29547.htm

**字符串处理在以后的工作中还是会经常碰到**的，掌握了之后，也就是10来个方法用来用去。

我认为**英语在编程里面特别重要**，学英文的方法不就是一个记记背背的事情么，编程也是一个记记背背的事，就是熟能生巧的事情。我之所以认为英语特别重要的原因，是因为编程语言就是给人用的，是人去编写各式各样的程序，只有看得懂，才能去使用它写程序。只不过发明这个语言的人是老外，所以，我们看起来会累一些。举个例子吧，**很多很多的方法名就可以用自己对英文的优势来理解从而背住它！而且我写了不少程序了...我觉得手敲代码是最容易帮助记忆的，今天敲了记住了，明天就会忘记，明天就继续敲，重复敲，每天边学新知识，边敲代码复习前几天的，效果特别好。之后隔个一段时间回头敲一敲复习一下，以后就记得很牢了。而且作为程序员，手速还是重要的...敲代码好处很多...都是为了熟能生巧，到后期，就轻松了，加油~**

**string的常用方法：**

**1.strObj.charCodeAt(index)：(方法具体作用不重要，但是借此理解很重要！！！！)**

**表示：返回一个指定字符的unicode码**

这个方法其实不是很重要，**但是从这个方法我们可以看出老外在发明这些方法名的时候是怎么样一个命名规则的，理解了，就容易记忆。**首先说一下strObj是一个什么，是一个string类型的字符串，什么意思，”123”,”baedgsfas”,”2354e5234” 像以上这种字符串就是**string类型的字符串**，也就是strObj是一个字符串参数，在这句话的前面我们会定义好strObj,比如：

var strObj=”werasdsef”;所以相当于是”werasdsef”.**charCodeAt(index)**，那后面的”.”在英文中不就是”的”的意思吗，charCodeAt是一个驼峰命名法，就是后面的单词虽然连在一起，但是用大写来告诉你这里已经是另外的单词了，charCodeAt就是 char code at翻译成中文就是”在...的字符码”,后面的参数index不就是一个位置吗，”werasdsef”的index为0取得就是w;是1，取的就是e;是2,取得就是r;直到index是 ”werasdsef”.length-1就是最大值了吧。所以

**strObj.charCodeAt(index)该怎么中文理解？？？**

**strObj这个字符串在index位置的字符的字符码。**

那到底字符码是什么，就是unicode码，具体是什么，有兴趣就百度，要学会百度解决问题。

例子：(自己手敲，自己编)

var str="tongwl";

console.log(str.charCodeAt(1)); //打印出来看一下 ”o”字符的unicode码是111

**2.String.fromCharCode(code1);(不重要，但是借此理解很重要！！！！)**

这个方法也不重要，**但是有注意点，可以借此理解一下！！！注意它是String. 不是strObj.**

为什么是String.?先来了解一下这个方法是为了获取什么，这个方法和 **strObj.charCodeAt(index)相反，strObj.charCodeAt(index)**是为了获取strObj的index位置的这个字符的unicode码，说明白了，不就是为了获得单个字符的unicode码，有可能是”a”,有可能是”b”,有可能是”我”,”你”,”@”这种字符，字符到底是什么，取决于strObj是什么字符串和index是几，这两个才能决定出取出来的字符到底是什么，然后再去获取这个字符串的

unicode码，而**String.fromCharCode(code1)**是为了通过code1这个字符码来获取对应的字符，那我们知道，一个字符在unicode码中对应的字符码是唯一的，比如unicode码是111的字符就是”o”,所以和具体的字符串(“abc”,”124”)是没有关系的，所以用一个抽象的String来开头。

举个例子：

console.log(String.fromCharCode(111)); //输出是o

**在用中文来理解下String.fromCharCode(code1)是什么意思？**

**From char code(code1)：从字符码code1得到的值(字符)**

1. **strObj.charAt(index) (要记住！)**

char at index ： 在index位置的字符是什么；若index错误(>length-1或则负数)，返 回空字符串。

例子：

var str="tongwl";

console.log(str.charAt(2));//输出n

1. **strObj.slice(start[,end])  (要记住！)**

截取字符串，**含头不含尾；start和end可以为负数；end省略的话会截取到最后；**

**start肯定是比end小的；**

例子：

var str=”tongwl”;

这3个根据情况可以选择使用

console.log(str.slice(1,2));//输出o

console.log(str.slice(2));//输出ngwl

console.log(str.slice(-2,-1));//输出w

1. **strObj.substring(start,[end])  (要记住！)**

截取字符串，功能和slice的一模一样，**除了负数并不会从最后开始截取，而是相当于 从0开始截取。**

例子：

var str=”tongwl”;

console.log(str.substring(1,3));//on

console.log(str.substring(3));//gwl

1. **strObj.substr(start[,length])  (要记住！)**

截取字符串，**第二个参数是截取的长度!!! start可以为负数；**

例子：

var str=”tongwl”;

console.log(str.substr(2));//ngwl

console.log(str.substr(2,2));//ng

console.log(str.substr(-2,2));//wl5

1. **strObj.indexOf(substr[,startIndex])  (要记住！)**

**(中文翻译:index of substr：(在strObj里面)substr的index是什么；)**

**找到字符串substr在strObj字符串中的第一次出现的索引，从startIndex开始搜索。**

**可以用来判断字符串中是否包含某字符串，没有匹配到则返回-1；**

**startIndex若有意义，则从startIndex开始往后寻找(包含startIndex);**

**startIndex如果是负数，自动省略掉；如果大于length-1，返回-1；**

例子：

var str="heheheda";

这3个可以用来判断字符串中是否包含某字符串

console.log(str.indexOf("eh"));//1

console.log(str.indexOf("eh",5));//-1

console.log(str.indexOf("eh",3));//3

1. **strObj.lastIndexOf(substr[,endIndex])  (要记住！)**

**返回String对象中字符串substr最后出现的索引，搜索到endIndex为止。**

例子：

var str="heheheda";

console.log(str.lastIndexOf('he'));//4

**console.log(str.lastIndexOf('he',2));//2，注意虽然endIndex是2，但是它会匹配后 面所有的**

1. **strObj.search(reExp)  (要记住,正则只需了解，有空了才复习)**

**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，找不到返回-1；**

**和indexOf的区别就是search可以用正则，indexOf可以有第二个参数startIndex;**

例子：

var str="heheheda";

console.log(str.search("da"));//6

console.log(str.search(/da/i));//6,**正则加一个i可以ignore大小写**

console.log(str.search("ha"));//-1

1. **strObj.concat([string1[,string2...]])  (感觉用处不大，了解下...字符串拼接用+号就可以完成)**

**字符串拼接；**

例子：

var str="heheheda";

console.log(str.concat("123","456"));//heheheda1234564

1. **strObj.split([separator[,length]])  (要记住，这个有些时候实用)**

**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。**

例子：

str="a,b,c,d,e,f";

var arr=str.split(",");//获取到的是一个数组

console.log(arr);//["a","b","c","d","e","f"]

console.log(str.split(",",3));//获取到的是一个限制了长度的数组["a","b","c"]

console.log(str.split("haha"));//获取到的是一个字符串的数组["a,b,c,d,e,f"]

1. **strObj.toLowerCase() (要记住)**

**转换为小写**

例子：

var str="xiaoxie";//注意所有字符串一旦创建，不可修改！！！！

str.toUpperCase();//这里是形成了新的字符串，并不会对原来的str进行修改

console.log(str);//不会变,仍旧是"xiaoxie"

console.log(str.toUpperCase());//"XIAOXIE"

1. **strObj.toUpperCase() (要记住)**

**转换为大写**

例子：

var str="DAXIE";

console.log(str.toLowerCase());

1. **match() (要记住)**

**语法**

str.match(regexp);

**参数**

**regexp**

一个[正则表达式](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/RegExp" \o "JavaScript/Reference/Global Objects/RegExp)对象。如果传入一个非正则表达式对象，则会隐式地使用 new RegExp(obj) 将其转换为一个 [RegExp](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/RegExp" \o "此页面仍未被本地化, 期待您的翻译!) 。如果你未提供任何参数，直接使用 match() ，那么你会得到一个包含空字符串的 [Array](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Array" \o "entries() 方法返回一个新的Array Iterator对象，该对象包含数组中每个索引的键/值对。) ：[""] 。

**返回值**

**array**

一个包含了整个匹配结果以及任何括号捕获的匹配结果的 [Array](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Array" \o "entries() 方法返回一个新的Array Iterator对象，该对象包含数组中每个索引的键/值对。) ；如果没有匹配项，则返回 [null](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/null" \o "值 null 是一个 JavaScript 字面量，表示空值（null or an \"empty\" value），即没有对象被呈现（no object value is present）。它是 JavaScript 原始值 之一。) 。

**描述**

1).**如果正则表达式没有g标志**，如果没有匹配到，则返回null，如果匹配到了，则str.match()返回值是一个关联数组，

数组的格式如下:

**[**

**0:匹配值，**

**index:匹配值在原字符串中的索引,**

**input:原字符串**

**]**

2).**如果正则表达式包含 g 标志，**则该方法返回一个 索引数组 ，它包含所有匹配的子字符串而不是匹配对象。捕获组不会被返回。如果没有匹配到，则返回  [null](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/null" \o "值 null 是一个 JavaScript 字面量，表示空值（null or an \"empty\" value），即没有对象被呈现（no object value is present）。它是 JavaScript 原始值 之一。) 。

**参看：RegExp 方法**

如果你需要知道一个字符串是否匹配一个正则表达式 [RegExp](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/RegExp" \o "此页面仍未被本地化, 期待您的翻译!) ，可使用 [search()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/String/search" \o "search() 方法执行正则表达式和 String对象之间的一个搜索匹配。) 。

如果你只是需要第一个匹配结果，你可能想要使用 [RegExp.exec()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/RegExp/exec" \o "exec() 方法在一个指定字符串中执行一个搜索匹配。返回一个结果数组或 null。) 。

如果你想要获得捕获组，并且设置了全局标志，你需要用 [RegExp.exec()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/RegExp/exec" \o "exec() 方法在一个指定字符串中执行一个搜索匹配。返回一个结果数组或 null。) 。

例子1：

**var *str***=**"aaga.sdfsf.sdfsdf.sdfsdf.sdftg.a.e.zsas"**;  
 **var *arr***=***str***.match(/[a-z]+/g); *//["aaga", "sdfsf", "sdfsdf", "sdfsdf", "sdftg", "a", "e", "zsas"]*

例子2：

**var *str***=**"aaga.sdfsf.sdfsdf.sdfsdf.sdftg.a.e.zsas"**;  
 **var *arr***=***str***.match(/[a-z]+/); *//["aaga", index: 0, input: "aaga.sdfsf.sdfsdf.sdfsdf.sdftg.a.e.zsas"]*

1. **string.replace(regexp/str,newStr/function) (要记住)**

replace() 方法返回一个由替换值替换一些或所有匹配的模式后的新字符串。模式可以是一个字符串或者一个[正则表达式](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/RegExp), 替换值可以是一个字符串或者一个每次匹配都要调用的函数。

**参数**

**regexp (pattern)**

一个 [RegExp](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/RegExp" \o "此页面仍未被本地化, 期待您的翻译!) 对象或者其字面量。该正则所匹配的内容会被第二个参数的返回值替 换掉。

**substr (pattern)**

一个要被 newSubStr 替换的[字符串](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/String" \o "此页面仍未被本地化, 期待您的翻译!)。其被视为一整个字符串，而不是一个正则表达 式。**仅仅是第一个匹配会被替换**。

**newSubStr (replacement)**

  用于替换掉第一个参数在原字符串中的匹配部分的 [字符串](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/String" \o "此页面仍未被本地化, 期待您的翻译!)。该字符串中可以内插一些 特殊的变量名。参考下面的[使用字符串作为参数](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/String/replace" \l "使用字符串作为参数)。

**function (replacement)**

一个用来创建新子字符串的函数，该函数的返回值将替换掉第一个参数匹配到的结果。 参考下面的[指定一个函数作为参数](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/String/replace" \l "指定一个函数作为参数)。

### 返回值

一个部分或全部匹配由替代模式所取代的新的字符串。

## **描述**

该方法并不改变调用它的字符串本身，而只是**返回一个新的替换后的字符串**。

**在进行全局的搜索替换时，正则表达式需包含 g 标志。**

### 使用字符串作为参数

替换字符串可以插入下面的特殊变量名：

|  |  |
| --- | --- |
| 变量名 | 代表的值 |
| $$ | 插入一个 "$"。 |
| $& | 插入匹配的子串。 |
| $` | 插入当前匹配的子串左边的内容。 |
| $' | 插入当前匹配的子串右边的内容。 |
| $n | 假如第一个参数是 [RegExp](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/RegExp" \o "此页面仍未被本地化, 期待您的翻译!)对象，并且 n 是个小于100的非负整数，那么插入**第 n 个括号匹配的字符串。** |

**例子：**

**var *str***=**"1990.11.23"**;  
 **var *newStr***=***str***.replace(/(\d{4})\.(\d{2})\.(\d{2})/g,**"$1年$2月$3日"**); *//输出 1990年11月23日*

### 指定一个函数作为参数:function(math,$1,$2....index,string)

你可以指定一个函数作为第二个参数。在这种情况下，当匹配执行后， 该函数就会执行。 函数的返回值作为替换字符串。 (注意:  上面提到的特殊替换参数在这里不能被使用。) 另外要注意的是， 如果第一个参数是正则表达式， 并且其为全局匹配模式， 那么这个方法将被多次调用， 每次匹配都会被调用。

下面是该函数的参数：

|  |  |
| --- | --- |
| 变量名 | 代表的值 |
| match | 匹配的子串。（对应于上述的$&。） |
| p1,p2, ... | 假如replace()方法的第一个参数是一个[RegExp](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/RegExp" \o "此页面仍未被本地化, 期待您的翻译!) 对象，则代表第n个括号匹配的字符串。（对应于上述的$1，$2等。） |
| offset | 匹配到的子字符串在原字符串中的偏移量。（比如，如果原字符串是“abcd”，匹配到的子字符串是“bc”，那么这个参数将是1） |
| string | 被匹配的原字符串 |

例子：

**var *str***=**"John Smith@#"**;  
 **var *reg***=**new** RegExp(**"(\\w+)\\s(\\w+)"**,**"g"**);

*//仅仅是将John Smith替换成了姓：Smith,名：John，@#并不会改变(消失)*  
 **var *newStr***=***str***.replace(***reg***,**function**(match,$1,$2,index,old){  
 **console**.log(match); *//John Smith* **console**.log(old); *//John Smith@#* **return "姓："**+$2+**",名："**+$1;  
 });  
 **console**.log(***newStr***); *//姓：Smith,名：John@#*

1. **reg.exec() (要记住)**

返回数组的格式如下:

**[**

**0:匹配值，**

**index:匹配值在原字符串中的索引,**

**input:原字符串**

**]**

例子：

**var *reg***=/ab\*/g;  
 **var *str***=**"abbcdefabh"**;  
 **var *myArray***;  
 **while**((***myArray***=***reg***.exec(***str***))!==**null**){  
 **console**.log(***myArray***); //myArray的数组和str.match(reg)不带g的数组一样  
 **console**.log(***reg***.lastIndex);  
 }

1. **reg.test(str) (要记住)**

str是否符合reg正则内容。

返回值：true，表示reg与str匹配，false，表示不匹配。

Tongwl