# [JS 字符串操作函数 往指定位置插入字符 删除指定位置字符 替换指定位置字符](http://blog.csdn.net/yeping090815/article/details/45191411)

2015-04-22 10:21 26458人阅读 [评论](http://blog.csdn.net/yeping090815/article/details/45191411" \l "comments)(0) [收藏](http://blog.csdn.net/yeping090815/article/details/javascript:void(0);" \o "收藏" \t "http://blog.csdn.net/yeping090815/article/details/_blank) [举报](http://blog.csdn.net/yeping090815/article/details/45191411" \l "report" \o "举报)

版权声明：本文为博主原创文章，未经博主允许不得转载。

1. · 插入   
   参数说明：str表示原字符串变量，flg表示要插入的字符串，sn表示要插入的位置  
   function insert\_flg(str,flg,sn){  
       var newstr="";  
       for(var i=0;i<str.length;i+=sn){  
           var tmp=str.substring(i, i+sn);  
           newstr+=tmp+flg;  
       }  
       return newstr;  
   }
2. 删除最后一位字符  
   s=s.substring(0,s.length-1)
3. 删除指定位置的字符 x代表要删除的位置 代表删除字符的个数
4. str = str.substring(0,x) + str.substring(x+num,str.length);
5. 删除指定字符  
   var str="xxxxxxxxabcxxxxxxxxxxxx";  
   var pattern = "abc";  
   str = str.replace(new RegExp(pattern), "");  
   console.log(str);  
   运行结果就是xxxxxxxxxxxxxxxxxxxx  
     
   指定位置替换字符串   
   <pre name="code" class="javascript">function changeStr(allstr,start,end,str,changeStr){ //allstr:原始字符串，start,开始位置,end：结束位  置,str：要改变的字，changeStr:改变后的字  
    if(allstr.substring(start-1,end-1) == str){  
         return allstr.substring(0,start-1)+changeStr+allstr.substring(end,allstr.length);   
    }else{  
         allstr;   
      }  
   }
6. concat
7. 将两个或多个字符的文本组合起来，返回一个新的字符串。
8. **var** a = "hello";
9. **var** b = ",world";
10. **var** c = a.concat(b);
11. alert(c);
12. //c = "hello,world"
13. indexOf
14. 返回字符串中一个子串第一处出现的索引（从左到右搜索）。如果没有匹配项，返回 -1 。
15. **var** index1 = a.indexOf("l");
16. //index1 = 2
17. **var** index2 = a.indexOf("l",3);
18. //index2 = 3
19. charAt
20. 返回指定位置的字符。
21. **var** get\_char = a.charAt(0);
22. //get\_char = "h"
23. lastIndexOf
24. 返回字符串中一个子串最后一处出现的索引（从右到左搜索），如果没有匹配项，返回 -1 。
25. **var** index1 = lastIndexOf('l');
26. //index1 = 3
27. **var** index2 = lastIndexOf('l',2)
28. //index2 = 2
29. match
30. 检查一个字符串匹配一个正则表达式内容，如果么有匹配返回 **null**。
31. **var** re = **new** RegExp(/^\w+$/);
32. **var** is\_alpha1 = a.match(re);
33. //is\_alpha1 = "hello"
34. **var** is\_alpha2 = b.match(re);
35. //is\_alpha2 = null
36. substring
37. 返回字符串的一个子串，传入参数是起始位置和结束位置。
38. **var** sub\_string1 = a.substring(1);
39. //sub\_string1 = "ello"
40. **var** sub\_string2 = a.substring(1,4);
41. //sub\_string2 = "ell"
42. substr
43. 返回字符串的一个子串，传入参数是起始位置和长度
44. **var** sub\_string1 = a.substr(1);
45. //sub\_string1 = "ello"
46. **var** sub\_string2 = a.substr(1,4);
47. //sub\_string2 = "ello"
48. replace
49. 用来查找匹配一个正则表达式的字符串，然后使用新字符串代替匹配的字符串。
50. **var** result1 = a.replace(re,"Hello");
51. //result1 = "Hello"
52. **var** result2 = b.replace(re,"Hello");
53. //result2 = ",world"
54. search
55. 执行一个正则表达式匹配查找。如果查找成功，返回字符串中匹配的索引值。否则返回 -1 。
56. **var** index1 = a.search(re);
57. //index1 = 0
58. **var** index2 = b.search(re);
59. //index2 = -1
60. slice
61. 提取字符串的一部分，并返回一个新字符串（与 substring 相同）。
62. **var** sub\_string1 = a.slice(1);
63. //sub\_string1 = "ello"
64. **var** sub\_string2 = a.slice(1,4);
65. //sub\_string2 = "ell"
66. split
67. 通过将字符串划分成子串，将一个字符串做成一个字符串数组。
68. **var** arr1 = a.split("");
69. //arr1 = [h,e,l,l,o]
70. length
71. 返回字符串的长度，所谓字符串的长度是指其包含的字符的个数。
72. **var** len = a.length();
73. //len = 5
74. toLowerCase
75. 将整个字符串转成小写字母。
76. **var** lower\_string = a.toLowerCase();
77. //lower\_string = "hello"
78. toUpperCase
79. 将整个字符串转成大写字母。
80. **var** upper\_string = a.toUpperCase();
81. //upper\_string = "HELLO"
83. /\*
84. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
85. 字符串函数扩充
86. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
87. \*/
89. /\*
90. ===========================================
91. //去除左边的空格
92. ===========================================
94. \*/
95. String.prototype.LTrim = **function**()
96. {
97. **return** **this**.replace(/(^\s\*)/g, "");
98. }

101. /\*
102. ===========================================
103. //去除右边的空格
104. ===========================================
105. \*/
106. String.prototype.Rtrim = **function**()
107. {
108. **return** **this**.replace(/(\s\*$)/g, "");
109. }
111. /\*
112. ===========================================
113. //去除前后空格
114. ===========================================
115. \*/
116. String.prototype.Trim = **function**()
117. {
118. **return** **this**.replace(/(^\s\*)|(\s\*$)/g, "");
119. }
121. /\*
122. ===========================================
123. //得到左边的字符串
124. ===========================================
125. \*/
126. String.prototype.Left = **function**(len)
127. {
129. **if**(isNaN(len)||len==**null**)
130. {
131. len = **this**.length;
132. }
133. **else**
134. {
135. **if**(parseInt(len)<0||parseInt(len)>**this**.length)
136. {
137. len = **this**.length;
138. }
139. }
141. **return** **this**.substr(0,len);
142. }

145. /\*
146. ===========================================
147. //得到右边的字符串
148. ===========================================
149. \*/
150. String.prototype.Right = **function**(len)
151. {
153. **if**(isNaN(len)||len==**null**)
154. {
155. len = **this**.length;
156. }
157. **else**
158. {
159. **if**(parseInt(len)<0||parseInt(len)>**this**.length)
160. {
161. len = **this**.length;
162. }
163. }
165. **return** **this**.substring(**this**.length-len,**this**.length);
166. }

169. /\*
170. ===========================================
171. //得到中间的字符串,注意从0开始
172. ===========================================
173. \*/
174. String.prototype.Mid = **function**(start,len)
175. {
176. **return** **this**.substr(start,len);
177. }

180. /\*
181. ===========================================
182. //在字符串里查找另一字符串:位置从0开始
183. ===========================================
184. \*/
185. String.prototype.InStr = **function**(str)
186. {
188. **if**(str==**null**)
189. {
190. str = "";
191. }
193. **return** **this**.indexOf(str);
194. }
196. /\*
197. ===========================================
198. //在字符串里反向查找另一字符串:位置0开始
199. ===========================================
200. \*/
201. String.prototype.InStrRev = **function**(str)
202. {
204. **if**(str==**null**)
205. {
206. str = "";
207. }
209. **return** **this**.lastIndexOf(str);
210. }
212. /\*
213. ===========================================
214. //计算字符串打印长度
215. ===========================================
216. \*/
217. String.prototype.LengthW = **function**()
218. {
219. **return** **this**.replace(/[^\x00-\xff]/g,"\*\*").length;
220. }
222. /\*
223. ===========================================
224. //是否是正确的IP地址
225. ===========================================
226. \*/
227. String.prototype.isIP = **function**()
228. {
230. **var** reSpaceCheck = /^(\d+)\.(\d+)\.(\d+)\.(\d+)$/;
232. **if** (reSpaceCheck.test(**this**))
233. {
234. **this**.match(reSpaceCheck);
235. **if** (RegExp.$1 <= 255 && RegExp.$1 >= 0
236. && RegExp.$2 <= 255 && RegExp.$2 >= 0
237. && RegExp.$3 <= 255 && RegExp.$3 >= 0
238. && RegExp.$4 <= 255 && RegExp.$4 >= 0)
239. {
240. **return** **true**;
241. }
242. **else**
243. {
244. **return** **false**;
245. }
246. }
247. **else**
248. {
249. **return** **false**;
250. }
252. }

255. /\*
256. ===========================================
257. //是否是正确的长日期
258. ===========================================
259. \*/
260. String.prototype.isLongDate = **function**()
261. {
262. **var** r = **this**.replace(/(^\s\*)|(\s\*$)/g, "").match(/^(\d{1,4})(-|\/)(\d{1,2})\2(\d{1,2}) (\d{1,2}):(\d{1,2}):(\d{1,2})$/);
263. **if**(r==**null**)
264. {
265. **return** **false**;
266. }
267. **var** d = **new** Date(r[1], r[3]-1,r[4],r[5],r[6],r[7]);
268. **return** (d.getFullYear()==r[1]&&(d.getMonth()+1)==r[3]&&d.getDate()==r[4]&&d.getHours()==r[5]&&d.getMinutes()==r[6]&&d.getSeconds()==r[7]);
270. }
272. /\*
273. ===========================================
274. //是否是正确的短日期
275. ===========================================
276. \*/
277. String.prototype.isShortDate = **function**()
278. {
279. **var** r = **this**.replace(/(^\s\*)|(\s\*$)/g, "").match(/^(\d{1,4})(-|\/)(\d{1,2})\2(\d{1,2})$/);
280. **if**(r==**null**)
281. {
282. **return** **false**;
283. }
284. **var** d = **new** Date(r[1], r[3]-1, r[4]);
285. **return** (d.getFullYear()==r[1]&&(d.getMonth()+1)==r[3]&&d.getDate()==r[4]);
286. }
288. /\*
289. ===========================================
290. //是否是正确的日期
291. ===========================================
292. \*/
293. String.prototype.isDate = **function**()
294. {
295. **return** **this**.isLongDate()||**this**.isShortDate();
296. }
298. /\*
299. ===========================================
300. //是否是手机
301. ===========================================
302. \*/
303. String.prototype.isMobile = **function**()
304. {
305. **return** /^0{0,1}13[0-9]{9}$/.test(**this**);
306. }
308. /\*
309. ===========================================
310. //是否是邮件
311. ===========================================
312. \*/
313. String.prototype.isEmail = **function**()
314. {
315. **return** /^\w+((-\w+)|(\.\w+))\*\@[A-Za-z0-9]+((\.|-)[A-Za-z0-9]+)\*\.[A-Za-z0-9]+$/.test(**this**);
316. }
318. /\*
319. ===========================================
320. //是否是邮编(中国)
321. ===========================================
322. \*/
324. String.prototype.isZipCode = **function**()
325. {
326. **return** /^[\\d]{6}$/.test(**this**);
327. }
329. /\*
330. ===========================================
331. //是否是有汉字
332. ===========================================
333. \*/
334. String.prototype.existChinese = **function**()
335. {
336. //[\u4E00-\u9FA5]為漢字﹐[\uFE30-\uFFA0]為全角符號
337. **return** /^[\x00-\xff]\*$/.test(**this**);
338. }
340. /\*
341. ===========================================
342. //是否是合法的文件名/目录名
343. ===========================================
344. \*/
345. String.prototype.isFileName = **function**()
346. {
347. **return** !/[\\\/\\*\?\|:"<>]/g.test(**this**);
348. }
350. /\*
351. ===========================================
352. //是否是有效链接
353. ===========================================
354. \*/
355. String.prototype.isUrl = **function**()
356. {
357. **return** /^http[s]?:\/\/([\w-]+\.)+[\w-]+([\w-./?%&=]\*)?$/i.test(**this**);
358. }

361. /\*
362. ===========================================
363. //是否是有效的身份证(中国)
364. ===========================================
365. \*/
366. String.prototype.isIDCard = **function**()
367. {
368. **var** iSum=0;
369. **var** info="";
370. **var** sId = **this**;
372. **var** aCity={11:"北京",12:"天津",13:"河北",14:"山西",15:"内蒙古",21:"辽宁",22:"吉林",23:"黑龙 江",31:"上海",32:"江苏",33:"浙江",34:"安徽",35:"福建",36:"江西",37:"山东",41:"河南",42:"湖 北",43:"湖南",44:"广东",45:"广西",46:"海南",50:"重庆",51:"四川",52:"贵州",53:"云南",54:"西藏",61:"陕西",62:"甘肃",63:"青海",64:"宁夏",65:"新疆",71:"台湾",81:"香港",82:"澳门",91:"国外"};
374. **if**(!/^\d{17}(\d|x)$/i.test(sId))
375. {
376. **return** **false**;
377. }
378. sId=sId.replace(/x$/i,"a");
379. //非法地区
380. **if**(aCity[parseInt(sId.substr(0,2))]==**null**)
381. {
382. **return** **false**;
383. }
385. **var** sBirthday=sId.substr(6,4)+"-"+Number(sId.substr(10,2))+"-"+Number(sId.substr(12,2));
387. **var** d=**new** Date(sBirthday.replace(/-/g,"/"))
389. //非法生日
390. **if**(sBirthday!=(d.getFullYear()+"-"+ (d.getMonth()+1) + "-" + d.getDate()))
391. {
392. **return** **false**;
393. }
394. **for**(**var** i = 17;i>=0;i--)
395. {
396. iSum += (Math.pow(2,i) % 11) \* parseInt(sId.charAt(17 - i),11);
397. }
399. **if**(iSum%11!=1)
400. {
401. **return** **false**;
402. }
403. **return** **true**;
405. }
407. /\*
408. ===========================================
409. //是否是有效的电话号码(中国)
410. ===========================================
411. \*/
412. String.prototype.isPhoneCall = **function**()
413. {
414. **return** /(^[0-9]{3,4}\-[0-9]{3,8}$)|(^[0-9]{3,8}$)|(^[0−9]3,4[0-9]{3,8}$)|(^0{0,1}13[0-9]{9}$)/.test(**this**);
415. }

418. /\*
419. ===========================================
420. //是否是数字
421. ===========================================
422. \*/
423. String.prototype.isNumeric = **function**(flag)
424. {
425. //验证是否是数字
426. **if**(isNaN(**this**))
427. {
429. **return** **false**;
430. }
432. **switch**(flag)
433. {
435. **case** **null**:        //数字
436. **case** "":
437. **return** **true**;
438. **case** "+":        //正数
439. **return**                /(^\+?|^\d?)\d\*\.?\d+$/.test(**this**);
440. **case** "-":        //负数
441. **return**                /^-\d\*\.?\d+$/.test(**this**);
442. **case** "i":        //整数
443. **return**                /(^-?|^\+?|\d)\d+$/.test(**this**);
444. **case** "+i":        //正整数
445. **return**                /(^\d+$)|(^\+?\d+$)/.test(**this**);
446. **case** "-i":        //负整数
447. **return**                /^[-]\d+$/.test(**this**);
448. **case** "f":        //浮点数
449. **return**                /(^-?|^\+?|^\d?)\d\*\.\d+$/.test(**this**);
450. **case** "+f":        //正浮点数
451. **return**                /(^\+?|^\d?)\d\*\.\d+$/.test(**this**);
452. **case** "-f":        //负浮点数
453. **return**                /^[-]\d\*\.\d$/.test(**this**);
454. **default**:        //缺省
455. **return** **true**;
456. }
457. }
459. /\*
460. ===========================================
461. //是否是颜色(#FFFFFF形式)
462. ===========================================
463. \*/
464. String.prototype.IsColor = **function**()
465. {
466. **var** temp        = **this**;
467. **if** (temp=="") **return** **true**;
468. **if** (temp.length!=7) **return** **false**;
469. **return** (temp.search(/\#[a-fA-F0-9]{6}/) != -1);
470. }
472. /\*
473. ===========================================
474. //转换成全角
475. ===========================================
476. \*/
477. String.prototype.toCase = **function**()
478. {
479. **var** tmp = "";
480. **for**(**var** i=0;i<**this**.length;i++)
481. {
482. **if**(**this**.charCodeAt(i)>0&&**this**.charCodeAt(i)<255)
483. {
484. tmp += String.fromCharCode(**this**.charCodeAt(i)+65248);
485. }
486. **else**
487. {
488. tmp += String.fromCharCode(**this**.charCodeAt(i));
489. }
490. }
491. **return** tmp
492. }
494. /\*
495. ===========================================
496. //对字符串进行Html编码
497. ===========================================
498. \*/
499. String.prototype.toHtmlEncode = **function**()
500. {
501. **var** str = **this**;
503. str=str.replace(/&/g,"&");
504. str=str.replace(/</g,"<");
505. str=str.replace(/>/g,">");
506. str=str.replace(/\'/g,"'");
507. str=str.replace(/\"/g,""");
508. str=str.replace(/\n/g,"<br>");
509. str=str.replace(/\ /g," ");
510. str=str.replace(/\t/g,"    ");
512. **return** str;
513. }
515. /\*
516. ===========================================
517. //转换成日期
518. ===========================================
519. \*/
520. String.prototype.toDate = **function**()
521. {
522. **try**
523. {
524. **return** **new** Date(**this**.replace(/-/g, "\/"));
525. }
526. **catch**(e)
527. {
528. **return** **null**;
529. }
530. }