10-第十章 字符串方法和数组

> `String`即文本（字符串），字符串方法都`不改原字符串`；
> 创建字符串的三种办法: `new String()`，`String()`，`字面量`，三种方式创建出来都是一样的

```
//一
var str = new String('hello');
//二
var str = String('hello');
//三
var str = 'hello'; //直接量
```

string`.length`属性可以返回字符串长度

一、`String`方法

```
    var  str = 'hello';
    str.length //字符串中字符长度（个数）
```

方法一

1）str.`indexOf(`value,index`)`查找字符串，返回查找字符串首次出现的`位置`;

方法对 `大小写敏感` ！
value 匹配字符
index 开始检索的位置, 合法值是 0 到 string.length - 1,默认0
匹配失败返回-1

```
1.      var  str = 'hello';
2.      str.indexOf('he')
```

str.`lastIndexOf(` value,index `)` 从后向前搜索,合法值是 0 到 string.length - 1,默认string.length - 1匹配失败返回-1

---

2)str.`concat( str,str...)` 字符串拼接

用于把一个或`多个字符串连接 到一块`，`返回拼接好的字符串`

2)str.`charAt(` index `)` 返回字符串指定索引的字符串

```
1.      var  str = 'hello';
2.      str.charAt(3)
```

---

3)str.`charCodeAt(` index `)` 返回字符串指定索引的Unicode编码

```
1.      var str =  '哈喽'
2.      var a = str.charCodeAt(1);
3.      alert( a );
```

---

4)`String.fromCharCode(` unic,unic,unic `)` 返回指定Unicode编码的字符串,从字符编码创建一个字符串。一个或多个 Unicode 值

```
1.      var  str = 'hello';
2.      str.length //字符串中字符长度（个数）
```

---

5) str.`substring(` start,end `)` 截取字符串，从`start` 开始，截止到`end`前，不包含end

> 如果没有end则从num开始整个查找；
> 如果 start 比 stop 大，那么该方法在提取子串之前会先交换这两个参数。
> str.substring(1,4)
> ————

```
1.      var  str = 'hello';
2.      str.length //字符串中字符长度（个数）
```

6) str.`slice(` start,end `)`

> a)和substring用法一样,从左往右截取
> b)start / end可以为负值，负值时，倒着从最后数
> c)start 和 end任何时候都不会交换位置，能截取才有值，截取不到则无值

7) str.`toLocaleUpperCase()` / str.`toLocaleLowerCase()`

> str.`toLocaleUpperCase()` 把字符串转换为大写。
> str.`toLocaleLowerCase()` 把字符串转换为小写。

方法二 可用于字符串亦可以用正则(`RegEXP`)

> 以下方法 可以用正则（RegExp）代替字符串进行匹配，

2) str.`match( )`

> str.`match(` value/RegExp `)``查找` 指定的值，返回 匹配的值 。未找到返回 `null`.
> 正则可以找一个或多个表达式

```
    var  str = 'hello world';
    str.match('o')//字符串
    var  str = 'hello world';
    str.match(/o/i)//正则

```

2) str.`search( )`

> str.`search(` value/RegExp `)` 返回 检索字符串首次出现的位置;未找到返回 `-1`

```
    var  str = 'hello world';
    str.search('o')//字符串中字符长度（个数）

```

3) str.`replace( )`

> str`.replace(` value/RegExp,new `)` 用一些字符 替换 另一些字符,new可以是字符串， 也可以是函数

```
    var  str = 'hello world';
    str.replace('o','千寻')//字符串中字符长度（个数）

```

```
var  str = 'hello world';
    //字符串中字符长度（个数）
    function fn(){
        return '帅气';
    }
    var  call = str.replace(/o/g,fn)//字符串中字符长度（个数）
    alert( call );
```

4) str.`split( )`

> str.`split( value/RegExp,length-1 )` 方法用于把一个字符串 `分割` 成 `字符串数组`，返回分割后的数组

```
1. 第二个参数是可选参数，是指定返回数组的长度，最大为str.length-1 ， 不写默认st
   r.length-1
```

```
1.      var  str = 'hello world';
2.      str.split('o',str.length)//字符串中字符长度（个数）
3.
```

二、Array() 数组

> 创建数组的三种办法: `new Array()`，`Array()`，`[]` ，三种方式创建出来都是一样的

```
1.      //一
2.   var arr = new Array();
3.      //二
4.   var arr = Array();
5.      //三
6.   var arr = [];//直接量
```

1、 `arr.length` 可以访问数组的长度

2、创建即指定数组长度Array(`length`)及 new Array(`length`),length是 数字的时候，创建的并不是数组的项，而是数组的长度，项的内容为 `undefined`

```
1. var  arr = new Array(2);
2. arr.length // 2
```

3、 `[]` 通过数组索引，访问值

```
1.  var arr = [1,2,3,4,5]
2.  arr[0];//1
```

4、修改数组指定索引下的值

```
1.  var arr = [1,2,3,4,5]
2. arr[2] = 8888;
```

5、在数组后面添加项

```
1. var arr = [1,2,3,4,5]
2.     arr[arr.length] = 8888;
3.     console.log(arr);
```

三、Array() 数组方法

1、unshift( )

> arr.unshift(item1,item1,....) 向数组的头部添加一个或更多元素，并返回（新的长度）。

2、arr.push( )

> arr.push(item1,item1,....) 向数组的尾部添加一个或更多元素，并返回（新的长度）。

3、arr.shift( )

> arr.shift( ) 删除数组的第一个元素（返回删除对象）;。

4、`arr.pop( )`

> arr.pop( ) 删除数组的`最后一个`元素（返回删除对象）。

5、arr.splice(index,howmany,item1,.....,itemX)（删除/添加）元素，然后（只返回删除对象）。

> `index` 必需。整数，规定添加/删除项目的索引，可以使用负数,`如果是添加，原有元素会往高位移动`。
> `howmany` 必需。要删除的项目数量。如果设置为 `0` ，则不会删除项目。
> `item1, ..., itemX`可选。向数组添加的新项目。

```
1.      //一 删除
2.      var arr = [1,2,3,4,5,6];
3.       arr.splice(2,2);
```

```
1.      //二 添加
2.      var arr = [1,2,3,4,5,6];
3.       arr.splice(2,0,'hello');
```

```
1.      //二 删除添加
2.      var arr = [1,2,3,4,5,6];
3.       arr.splice(2,1,'hello');
```

6、 arr.`sort()` 排序

arr.`sort(function(a,b){ return a-b;})` ————————————————————————————————
`修改原数组`

- 默认arr.sort() 以首字符编码大小排序
- 数组length小于10以`冒泡排序`
    - 冒泡排序下依次比较，
      return `>0` 调换位置，`=0`不调换位置，`<0` 不调换位置
- 数组length大于10以`二分排序`

7、 arr.`reverse()` 反转数组

reverse() 颠倒 数组中元素的顺序————————————————————————————————
`修改原数组`

三、Array() 数组方法

!!!!以上方法不创建新的创建，而是`------------------------直接修改原有的数组`,同时索引会变化

8、arr.`concat()` 数组拼接

arr.concat(arr1,arr2,…,arrN)：`数组拼接` 连接两个或更多的数组，（`并返回拼接成的新数组`）

- 该数组是通过把所有 arrX 参数添加到 arr 中生成的。
- 如果要进行 concat() 操作的参数是数组，那么添加的是数组中的元素，而不是数组 —— `不修改原数组`

```
// 情况一，
        var a = [1,2,3];
        alert(a.concat(4,5));
 // 情况二，不会改变a1,而是生成一个新的数组！！！
            var a1 = [1,2,3];
            var a2 = [4,5,6];
            alert(a1.concat(a2));
```

---

9 arr.`slice()` 截取

> `arr.slice(` start,end `)` 方法从已有的数组中返回选定的元素。————————`不修改原数组`

10、arr.`join()` 拼接成字符串

> ——————————————————————————————————`不修改原数组`

11.`Array.isArray( )` 判断是不是数组

> `Array.isArray(` object `)` //返回一个布尔值

四、ECMAscript5 的遍历数组方法

> 以下下方法 `都能实现遍历` ，`语法也都一样` ，只是 `返回值不一样` ——————————不修改原数组
> array. `xxxx(` function(`currentValue` , `index` , `arr`), `thisValue` `)`

参数 描述
`currentValue` ——————必须。当前元素的`值`
`index` ——————————可选。当期元素的`索引`值
`arr` ——————————可选。当期元素属于的`数组对象`
`thisValue` ——————————可选。对象作为该执行回调时使用，传递给函数，`用作"this" 的值`。
如果省略了 thisValue ，”this” 的值为 “undefined”

function(currentValue, index,arr) 必须。函数，数组中的每个元素都会执行这个函数

1、`forEach()`

> arr.`forEach()` 从头至尾遍历数组
> ——————————————————————————————无返回值

2、`map()` 返回值数组

> arr.`map()` 返回一个数组，包含函数所有返回值
> ——————————————————————————-返回数组

```
var  arr = [1,2,3,4];
var _new = arr.map(function(x){
     return x * x;
})
```

3、`filter()` true数组

> arr.`filter()` 返回值是一个 return 值为true或能转化为true的值
> ——————————————————返回数组

```
var  arr = [1,2,3,4];
var _new = arr.filter(function(x){
     return x > 3 ;
})
```

> `every() ,some()` 是数组逻辑判定：返回`true`或`false`

4、`every()`

> arr.`every()` 针对所有元素，即都为true 则返回true
> ——————————————————————————————返回值

```
var  arr = [1,2,3,4];
var _new = arr.every(function(x){ return x < 10;})   // true  都小于10
var _new = arr.every(function(x){ return x%2 == 0;}) //false 是不是都是偶数
```

4、`some()`

> arr.`some()` 是否存在 即有一个是true则为true
> ——————————————————————————————返回值

```
var  arr = [1,2,3,4];
var _new = arr.some(function(x){ return x%2 === 0;})   // true  有偶数

```