创建数组：

var arr = new Array( );

|  |  |
| --- | --- |
| Array( ) 括号里面如果是 | 代表 |
| 一个数字参数 | 这个数组的长度 |
| 一个字符串参数 | arr[0]的内容 |
| 多个参数 | 数组里面的内容 |
| 使用arguments实现的参数不同，意义不同，可以用该属性模拟重载 | |

稀疏数组：索引号不连续。

for ( var i = 0; i <= 100; i ++) {

if ( i % 2 !== 0 ) {

arr [ arr.length ] = i; } }

.length属性是动态改变的，通过arr.length可以动态追加；

.length属性会对数组的长度产生影响。

出现undefined的情况：声明变量但是未赋值；声明了length，但是未给数组的项赋值。

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | | | |
|  |  | 返回值 | 示例 |
| join(分隔符) | 把数组的所有元素（忽略空格）放入一个字符串。元素通过指定的分隔符进行分隔 |  | arr.join("-") |
| **valueOf** |  |  |  |
| **toString** |  |  | arr.toString() |
| **indexOf**(searchvalue,fromindex) | 返回指定值在数组中首次出现的位置 |  |  |
| **lastIndexOf** |  |  | arr.lastIndexOf() |
| arr instanceof Array | 判断arr是不是数组类型 | true / false |  |
| Array.isArray(arr) |  |
| reverse() | 颠倒数组中元素的顺序 | 改变原来的数组，而不会创建新的数组 |  |
| **concat** | 连接两个或更多的数组，并返回结果。  参数可以是数组，也可以是具体的值 | 不改变现有的数组，而是返回被连接数组的副本 | arr1.concat(arr2,arr3);  arr.concat(1); |
| sort() | 对数组的元素进行排序 |  |  |
| push() | 向数组末尾添加元素 | 新的长度 |  |
| pop() | 删除并返回数组的最后一个元素 | 最后一个元素 |  |
| unshift() | 从前面加入元素 | 返回数组的长度 |  |
| shift() | 从前面删除元素 | 返回被删除的元素 |  |
| forEach(function(v, i){ }) |  |  |  |
| **slice**(start, end)  切成薄片 | 从已有的数组中返回选定的元素，原数组保持不变 | 返回切出来的新数组 |  |
| splice ( index, howmany, element1, ....., elementX)  捻接、粘接 | 删除从 index 开始的零个或多个元素，并且用参数列表中声明的一个或多个值来替换那些被删除的元素 | 返回被删除项组成的新数组，原数组也改变了。 |  |

\*六种红色方法是string和array共有的。

str.split() 是把字符串分割成数组，arr.join()是把数组放入字符串。

字符串没有splice方法，只有数组有。但是字符串和数组都有slice() / indexOf() / 方法。

arr.sort() 如果不传参数，默认按照每一项的第一个字符的unicode编码来排列

arr.sort(compareFunction)

1. 如果 compareFunction(a, b) 小于 0 ，那么 a 会被排列到 b 之前；
2. 如果 compareFunction(a, b) 等于 0 ， a 和 b 的相对位置不变；
3. 如果 compareFunction(a, b) 大于 0 ， b 会被排列到 a 之前。

希望比较数字且按升序排列，代码如下：

function compareNumbers (a,b) {

return a-b; }

对象可以按照某个属性排序：

var arr = [  
 { name: "Lily", age: 18},  
 { name: "Andy", age: 8},  
 { name: "Joy", age: 30}  
];  
arr.sort (function (a, b) {  
 return a.age - b.age;  
});  
console.log(arr);

console.log(Object.prototype.toString.call(arr));

讲以下两个方法之前，先给大家举两个日常的例子。

1. 开学照证件照，照完之后做了个照片墙。然后来上JavaScript，铃声响了，老师说“下课了，大家都吃饭去吧！”，每个人就都欢欣雀跃地去食堂了。吃完之后用不用给老师带一碗？不用。每个学生的长相有没有变化？没有，还是照片墙上证件照的样子。



（天津大学学三食堂）

1. 老师说：“放假了，都去旅游吧”，然后班上的每一个同学都去了趟韩国，顺便整了个容，然后开开心心地带着你的盛世容颜回来了。那原来照片墙上的照片变了吗？没有，还记录着你的黑历史。但是你的容貌变了，就要再重新做一个新的照片墙了。

### forEach方法

**forEach()** 方法对数组的每个元素执行一次提供的函数。

*arr.forEach(function callback(currentValue, index, array) {*

*console.log(index);*

*}[, thisArg]);*

*返回值是undefined，第三个参数可写可不写，都可以输出arr*

示例：

var arr = [55,33,6,22,6];

arr.forEach(function (v, i, arr) {

console.log("arr[" + i + "] = " + v);

})

**map方法**

**map()** 方法创建一个新数组，其结果是该数组中的每个元素都调用一个提供的函数后返回的结果。

*var new\_array = arr.map(function callback(currentValue[, index[, array]]) {*

*// Return element for new\_array*

*}[, thisArg])*

*The map() method creates a new array with the results of calling a provided function on every element in the calling array.*

示例：

var arr = [55,33,6,22,6];

arr.map(x => x \* 2)

所以这两个方法最大的区别就是看元素有没有改变。