### push

#### push()

方法将一个或多个元素添加到数组的末尾，并返回新数组的长度  
语法：arr.push(element1, ..., elementN)  
参数:被添加到数组末尾的元素  
返回值：当调用该方法时，新的 length 属性值将被返回  
是否影响老数组：影响  
**var *arr*** =[1,2,3,4];  
**var *res*** = ***arr***.push(**"5"**,**true**,{})  
**console**.log(***arr***,***res***)

### unshift

#### unshift() 方法将一个或多个元素添加到数组的开头，并返回新数组的长度 //语法：arr.unshift(element1, ..., elementN) //参数：要添加到数组开头的元素 //返回值：当一个对象调用该方法时，返回其 新的length 属性值 //是否影响老数组：影响 **var *arr*** =[1,2,3,4]; **var *result*** = ***arr***.unshift(0,-9); **console**.log(***result***,***arr***);

### pop

pop()方法从数组中删除最后一个元素，并返回该元素的值。此方法更改数组的长度  
语法：arr.pop()

参数：无  
返回值：从数组中删除的元素(当数组为空时返回undefined)  
是否影响老数组：影响**var *arr*** =[1,2,3,**"abc"**];  
**var *result*** = ***arr***.pop();  
**console**.log(***result***,***arr***);

### Shift

*shift() 方法从数组中删除第一个元素，并返回该元素的值。此方法更改数组的长度  
语法：arr.shift()  
参数：无  
//返回值：从数组中删除的元素; 如果数组为空则返回undefined 。   
//是否影响老数组：影响***var *arr*** =[**"biubiubiu"**,2,3,4];  
**var *result*** = ***arr***.shift();  
**console**.log(***result***,***arr***);

### Indexof

*indexOf()方法返回在数组中可以找到一个给定元素的第一个索引，如果不存在，则返回-1。  
语法：arr.indexOf(searchElement)  
参数：searchElement-->要查找的元素  
返回值：首个被找到的元素在数组中的索引位置; 若没有找到则返回 -1  
//是否影响老数组：不影响***var *arr*** =[1,2,1,4];  
**var *result*** = ***arr***.indexOf(1);  
**console**.log(***result***,***arr***);

### 6.lastindexof

*lastIndexOf() 方法返回指定元素在数组中的最后一个的索引，如果不存在则返回 -1  
语法：arr.lastIndexOf(searchElement)  
参数：searchElement-->要查找的元素  
返回值：数组中最后一个元素的索引，如未找到返回-1  
是否影响老数组：不影响***var *arr*** =[1,2,1,4];  
**var *result*** = ***arr***.lastIndexOf(1);  
**console**.log(***result***,***arr***);

### 7.Concat

*concat() 方法用于合并两个或多个数组。此方法不会更改现有数组，而是返回一个新数组。  
语法：arr.concat(arr1[, arr2[, ...[, arrN]]])  
参数：valueN-->待链接的数组  
返回值：新的 Array 实例  
是否影响老数组：不影响***var *arr*** =[1,2,1,4];  
**var *arr1***=[5];  
**var *arr2***=[6];  
**var *result*** = ***arr***.concat(***arr1***,***arr2***);  
**console**.log(***result***,***arr***,***arr1***,***arr2***)

### Join

*join() 方法将一个数组的所有元素连接成一个字符串并返回这个字符串  
语法：arr.join(separator)  
参数：separator ---> 指定一个字符串来分隔数组的每个元素。如果省略()，数组元素用逗号分隔。默认为 ","。  
返回值：一个所有数组元素连接的字符串。如果 arr.length 为0，则返回空字符串  
是否影响老数组：不影响***var *arr*** =[1,2,1,4];  
**var *result*** = ***arr***.join(**"damu"**);  
**console**.log(***result***,***arr***);

### tostring

*toString() 返回一个字符串，表示指定的数组及其元素。  
语法：arr.toString()  
参数：无  
返回值：返回一个字符串。   
是否影响老数组：不影响***var *arr*** =[1,2,3,4];  
**var *result*** = ***arr***.toString();  
**console**.log(***result***,***arr***);

### Foreach

**var *arr*** = [**"a"**,**"b"**,**"c"**,**"d"**,**"e"**,**"f"**,**"g"**];  
  
*隐式迭代  
回调函数  
 异步的回调函数(异步编程的核心)  
 同步的回调函数  
   
 什么是回调函数  
 一般回调函数是一个匿名函数  
 函数 是不是你定义的?  
 是  
 函数 你有没有调用?  
 没有  
 函数 最终有没有执行?  
 有****arr***.forEach(**function**(){  
 **console**.log(**"xx"**)  
})  
**console**.log(**"after"**)

*forEach() 方法对数组的每个元素执行一次提供的函数。  
参数1:callback  
 为数组中每个元素执行的函数，该函数接收三个参数：  
 currentValue(当前值):数组中正在处理的当前元素。  
 index(索引):数组中正在处理的当前元素的索引。  
 array:forEach()方法正在操作的数组。  
参数2:thisArg  
 可选参数。当执行回调 函数时用作this的值  
返回值：undefined  
是否影响老数组：由callback决定*

### Map

*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  
 参数1:callback  
 为数组中每个元素执行的函数，该函数接收三个参数：  
 currentValue(当前值):数组中正在处理的当前元素。  
 index(索引):数组中正在处理的当前元素的索引。  
 array:forEach()方法正在操作的数组。  
 参数2:thisArg  
 可选参数。当执行回调 函数时用作this的值。  
 返回值：一个新数组，每个元素都是回调函数的结果。  
 是否影响老数组：不影响  
 var arr = ["a","b","c","d","e","f","g"];* **var *arr*** = [9,16,36,9,4,81,49]  
 *var newArr = arr.map(function(item,index,arr){  
 return Math.sqrt(item);  
 var a =0;  
 return a++;  
 })*

*//伪代码  
 var Math ={  
 sqrt:function(item){  
 return val;  
 }  
 }* **var *newArr*** = ***arr***.map(***Math***.sqrt);  
  
 *var newArr = arr.map(function(item,index,arr){  
 return Math.sqrt(item);  
 })* **console**.log(***newArr***)

### Every

*every() 方法测试数组的所有元素是否都通过了指定函数的测试   
参数1:callback  
 /\*为数组中每个元素执行的函数，该函数接收三个参数：  
 currentValue(当前值):数组中正在处理的当前元素。  
 index(索引):数组中正在处理的当前元素的索引。  
 array:forEach()方法正在操作的数组。\*/  
参数2:thisArg  
 /\*可选参数。当执行回调 函数时用作this的值。\*/  
  
返回值：true false  
  
是否影响老数组：不影响***var *arr*** =[1,2,3,4,5,6,7,8];  
**var *flag*** = ***arr***.every(**function**(item,index){  
 **if**(item > 2){  
 **return true**;  
 }  
})  
  
**console**.log(***flag***)

### Some

*some() 方法测试数组中的某些元素是否通过由提供的函数实现的测试。\  
参数1:callback  
 为数组中每个元素执行的函数，该函数接收三个参数：  
 currentValue(当前值):数组中正在处理的当前元素。  
 index(索引):数组中正在处理的当前元素的索引。  
 array:forEach()方法正在操作的数组。  
参数2:thisArg  
 可选参数。当执行回调 函数时用作this的值。  
  
返回值：true false  
  
是否影响老数组：不影响***var *arr*** =[1,2,3,4,5,6,7,8];  
**var *result*** = ***arr***.some(**function**(item,index){  
 **console**.log(**"xxx"**)  
 **if**(item > 1){  
 **return true**;  
 }  
})  
  
**console**.log(***result***);

### Filter

*filter() 方法创建一个新数组, 其包含通过所提供函数实现的测试的所有元素。  
参数1:callback  
 为数组中每个元素执行的函数，该函数接收三个参数：  
 currentValue(当前值):数组中正在处理的当前元素。  
 index(索引):数组中正在处理的当前元素的索引。  
 array:forEach()方法正在操作的数组。  
参数2:thisArg  
 可选参数。当执行回调 函数时用作this的值。  
  
返回值： 一个新的通过测试的元素的集合的数组  
  
是否影响老数组：不影响***var *arr*** =[1,2,3,4,5,6,7,8,9,10,11];  
**function** *fn*(item,index){  
 **console**.log(**"xxx"**)  
 **if**(item > 1){  
 **return true**;  
 }  
}  
**var *result*** = ***arr***.filter(*fn*)  
**console**.log(***result***);

### Slice

*slice() 方法返回一个从开始到结束（不包括结束，左闭右开）选择的数组的一部分浅拷贝到一个新数组对象  
参数  
   
 begin 可选  
 end   
 slice(begin,end):[begin,end)  
   
   
返回值：一个含有提取元素的新数组  
  
是否影响老数组：不影响***var *arr*** =[1,25,34,14,26,16,47,28];  
**var *newArr*** = ***arr***.slice(2, 6);  
**console**.log(***newArr***)

### Splice

*splice() 方法通过删除现有元素和/或添加新元素来更改一个数组的内容  
参数  
   
 start 从start开始删  
 deleteCount 删几个  
 item列表 删完之后在相同位置处加哪几个  
 返回值：由被删除的元素组成的一个数组  
  
是否影响老数组：影响  
  
  
//数组的CUD***var *arr*** =[1,25,34,14,26,16,47,28];  
*// C U D****arr***.splice(1,1,**"A"**,**"B"**,**"C"**)  
**console**.log(***arr***);

### Sort

*sort() 方法使用in-place算法对数组的元素进行排序，并返回数组。  
 默认排序顺序是根据字符串Unicode码点。  
 //参数compareFunction 可选。  
   
 用来指定按某种顺序进行排列的函数。如果省略，  
 元素按照转换为的字符串的各个字符的Unicode位点进行排序。  
   
   
 返回值：返回排序后的数组。  
   
 是否影响老数组： 影响 原数组已经被排序后的数组代替。* **var *arr*** =[1,2,3,4,26,16,147,28];  
 **var *result*** = ***arr***.sort(**function**(a,b){  
 *//从小到大* **return** a-b;  
 *//从大到小  
 return b-a;* })  
 **console**.log(***result***);

### Reverse

*reverse 方法颠倒数组中元素的位置，并返回该数组的引用  
语法： arr.reverse()  
参数：无  
返回值：该数组的引用。   
是否影响老数组：影响***var *arr*** =[1,2,3,4];  
**var *result*** = ***arr***.reverse();  
**console**.log(***result***,***arr***);  
**console**.log(***arr*** == ***result*** )

### Reduce

*reduce() 方法对累加器和数组中的每个元素（从左到右）应用一个函数，将其减少为单个值。  
参数1:callback  
   
 为数组中每个元素执行的函数，该函数接收三个参数：  
 accumulator(当前值):累加器累加回调的返回值; 它是上一次调用回调时返回的累积值  
 currentValue:数组中正在处理的元素。  
 currentIndex可选:数组中正在处理的当前元素的索引。 如果提供了initialValue，则索引号为0，否则为索引为1。  
 array可选：调用reduce的数组。  
   
参数2:initialValue  
 用作第一个调用 callback的第一个参数的值。 如果没有提供初始值，则将使用数组中的第一个元素。  
 在没有初始值的空数组上调用 reduce 将报错。  
 返回值： 累加器  
是否影响老数组：不影响  
  
  
//数组里所有值的和***var *arr*** =[2, 1, 2, 3];  
**var *sum*** = ***arr***.reduce(**function** (a,b,index,arr) {  
 **console**.log(a)  
 **return** a + b;  
},0);  
**console**.log(***sum***);  
**console**.log(***arr***);