- `>`表示输入的代码
- `...`表示这是一个代码块
- 以上两种都没有表示输出内容/函数返回值

map

- 对数组中每一个元素做映射操作
- 不改变原数组，返回新数组

```
> arr
[ 1, 2, 3, 4 ]
> arr.map((item, index) => {
... return item * 2
... })
[ 2, 4, 6, 8 ]
```

filter

- 过滤数组中的内容
- 不改变原数组，返回新数组

```
> arr
[ 1, 2, 3, 4 ]
> arr.filter((item, index) => {
... return item % 2 === 0  // 结果为true的保留，为false的删除
... })
[ 2, 4 ]
```

forEach

- 遍历数组，可取得数组的每个元素以及其下标
- 不改变原数组

```
> arr
[ 1, 2, 3, 4 ]
> arr.forEach((item, index, arr) => {
... console.log(item);
... console.log(index);
... console.log(arr);
... })
1
0
[ 1, 2, 3, 4 ]
2
1
[ 1, 2, 3, 4 ]
3
2
[ 1, 2, 3, 4 ]
4
3
[ 1, 2, 3, 4 ]
undefined
```

reduce

- 数组累计操作，具体效果见例子
- 不改变原数组，返回最终值

```
> arr
[ 1, 2, 3, 4 ]
// 计算数组中所有元素之和再加上10
// reduce第二个参数为sum的初始值，若设置了该参数，则item从数组第一个元素开始
> arr.reduce((sum, item, index) => {
... sum += item;
... return sum;
... }, 10)
20
// 计算数组中所有元素之和
// 未设置sum的初始值，则sum的初始值设为数组的第一个元素，item从数组第二个元素开始
> arr.reduce((sum, item, index) => {
... sum += item;
... return sum;
... })
```

```
10
```

sort

- 对数组进行排序（具体原理是插入排序加快速排序）
- 改变原数组

```
> arr
[ 1, 2, 3, 4 ]
// 降序
> arr.sort((x, y) => y - x)
[ 4, 3, 2, 1 ]
// 升序
> arr.sort((x, y) => x - y)
[ 1, 2, 3, 4 ]
```

slice

- 切片，从数组中取出一部分
- 不改变原数组，返回一个新数组

```
> arr
[ 1, 2, 3, 4 ]
// 从第0个开始，一直切到第2个
> arr.slice(0, 3)
[ 1, 2, 3]
> arr
[ 1, 2, 3, 4 ]
```

splice

- 删除数组中指定位置、个数元素，也可以填充数组
- 改变原数组，返回被删除的内容

```
> arr
[ 1, 2, 3, 4 ]
// 从0开始，删除两元素
> arr.splice(0, 2)
[ 1, 2 ]
> arr
[ 3, 4 ]

> arr = [1, 2, 3, 4]
[ 1, 2, 3, 4 ]
// 从0开始，删除两个元素，并在删除的位置插入一个5
```

```
> arr.splice(0, 2, 5)
[ 1, 2 ]
> arr
[ 5, 3, 4 ]

> arr = [1, 2, 3, 4]
[ 1, 2, 3, 4 ]
// 从0开始，删除两个元素，并在删除的位置插入5, 6, 7
> arr.splice(0, 2, 5, 6, 7)
[ 1, 2 ]
> arr
[ 5, 6, 7, 3, 4 ]

> arr = [1, 2, 3, 4]
[ 1, 2, 3, 4 ]
// 从0开始，删除两个元素，并在删除的位置插入5, 6, 7, 8, 9
> arr.splice(0, 2, ...[5, 6, 7, 8, 9])
[ 1, 2 ]
> arr
[
  5, 6, 7, 8,
  9, 3, 4
]
```

reverse

- 翻转数组
- 改变原数组

```
> arr = [1, 2, 3, 4]
[ 1, 2, 3, 4 ]
> arr.reverse()
[ 4, 3, 2, 1 ]
```

join

- 将数组拼接为字符串
- 不改变原数组，返回字符串

```
> arr
[ 1, 2, 3, 4 ]
// 不传参数，默认用逗号分隔
> arr.join('**')
'1**2**3**4'
```

push, pop, unshift, shift

- push：在数组尾添加元素
- pop：删除数组尾一个元素
- unshift：在数组头添加元素
- shift：删除数组头一个元素

```
> arr
[ 1, 2, 3, 4 ]
> arr.push(5)
5
> arr
[ 1, 2, 3, 4, 5 ]
// 可添加多个元素
> arr.push(6, 7)
7
> arr
[
  1, 2, 3, 4,
  5, 6, 7
]
// 添加一个数组的办法
> arr.push(...[8, 9, 10])
10
> arr
[
  1, 2, 3, 4,  5,
  6, 7, 8, 9, 10
]
> arr.pop()
10
> arr
[
  1, 2, 3, 4, 5,
  6, 7, 8, 9
]
> arr.unshift(0)
10
> arr
[
  0, 1, 2, 3, 4,
  5, 6, 7, 8, 9
]
// 同样可以添加多个元素
> arr.unshift(-2, -1)
12
> arr
[
```

```
  -2, -1, 0, 1, 2,
   3,  4, 5, 6, 7,
   8,  9
]
> arr.unshift(...[-5, -4, -3])
15
> arr
[
  -5, -4, -3, -2, -1, 0,
   1,  2,  3,  4,  5, 6,
   7,  8,  9
]
> arr.shift()
-5
> arr
[
  -4, -3, -2, -1, 0, 1,
   2,  3,  4,  5, 6, 7,
   8,  9
]
```

fill

- 填充数组，常用于初始化一个数组

```
> arr = new Array(10).fill(0)
[
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0
]
> arr
[
  0, 0, 0, 0, 0,
  0, 0, 0, 0, 0
]
```

concat

- 拼接数组
- 不改变原数组，返回拼接后的数组

```
> arr
[ 1, 2, 3, 4 ]
> arr.concat([5, 6, [7, 8]])
[ 1, 2, 3, 4, 5, 6, [ 7, 8 ] ]
> arr
[ 1, 2, 3, 4 ]
```

indexOf

- 查找一个元素在数组中的位置

```
> arr
[ 1, 2, 3, 4 ]
> arr.indexOf(3)
2
> arr.indexOf(5)
-1
```