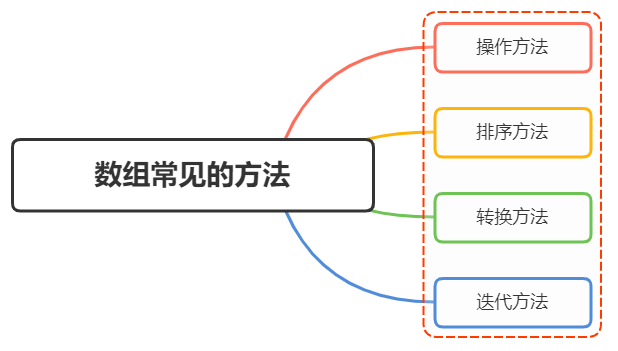
# 面试官：数组的常用方法有哪些？



## 一、操作方法

数组基本操作可以归纳为 增、删、改、查，需要留意的是哪些方法会对原数组产生影响，哪些方法不会

下面对数组常用的操作方法做一个归纳

### 增

下面前三种是对原数组产生影响的增添方法，第四种则不会对原数组产生影响

* push()
* unshift()
* splice()
* concat()

#### push()

push()方法接收任意数量的参数，并将它们添加到数组末尾，返回数组的最新长度

let colors = []; // 创建一个数组  
let count = colors.push("red", "green"); // 推入两项  
console.log(count) // 2

#### unshift()

unshift()在数组开头添加任意多个值，然后返回新的数组长度

let colors = new Array(); // 创建一个数组  
let count = colors.unshift("red", "green"); // 从数组开头推入两项  
alert(count); // 2

#### splice

传入三个参数，分别是开始位置、0（要删除的元素数量）、插入的元素，返回空数组

let colors = ["red", "green", "blue"];  
let removed = colors.splice(1, 0, "yellow", "orange")  
console.log(colors) // red,yellow,orange,green,blue  
console.log(removed) // []

#### concat()

首先会创建一个当前数组的副本，然后再把它的参数添加到副本末尾，最后返回这个新构建的数组，不会影响原始数组

let colors = ["red", "green", "blue"];  
let colors2 = colors.concat("yellow", ["black", "brown"]);  
console.log(colors); // ["red", "green","blue"]  
console.log(colors2); // ["red", "green", "blue", "yellow", "black", "brown"]

### 删

下面三种都会影响原数组，最后一项不影响原数组：

* pop()
* shift()
* splice()
* slice()

#### pop()

pop() 方法用于删除数组的最后一项，同时减少数组的length 值，返回被删除的项

let colors = ["red", "green"]  
let item = colors.pop(); // 取得最后一项  
console.log(item) // green  
console.log(colors.length) // 1

#### shift()

shift()方法用于删除数组的第一项，同时减少数组的length 值，返回被删除的项

let colors = ["red", "green"]  
let item = colors.shift(); // 取得第一项  
console.log(item) // red  
console.log(colors.length) // 1

#### splice()

传入两个参数，分别是开始位置，删除元素的数量，返回包含删除元素的数组

let colors = ["red", "green", "blue"];  
let removed = colors.splice(0,1); // 删除第一项  
console.log(colors); // green,blue  
console.log(removed); // red，只有一个元素的数组

### slice()

slice() 用于创建一个包含原有数组中一个或多个元素的新数组，不会影响原始数组

let colors = ["red", "green", "blue", "yellow", "purple"];  
let colors2 = colors.slice(1);  
let colors3 = colors.slice(1, 4);  
console.log(colors) // red,green,blue,yellow,purple  
concole.log(colors2); // green,blue,yellow,purple  
concole.log(colors3); // green,blue,yellow

#### 改

即修改原来数组的内容，常用splice #### splice()

传入三个参数，分别是开始位置，要删除元素的数量，要插入的任意多个元素，返回删除元素的数组，对原数组产生影响

let colors = ["red", "green", "blue"];  
let removed = colors.splice(1, 1, "red", "purple"); // 插入两个值，删除一个元素  
console.log(colors); // red,red,purple,blue  
console.log(removed); // green，只有一个元素的数组

#### 查

即查找元素，返回元素坐标或者元素值

* indexOf()
* includes()
* find()

#### indexOf()

返回要查找的元素在数组中的位置，如果没找到则返回 -1

let numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];  
numbers.indexOf(4) // 3

#### includes()

返回要查找的元素在数组中的位置，找到返回true，否则false

let numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];  
numbers.includes(4) // true

#### find()

返回第一个匹配的元素

const people = [  
 {  
 name: "Matt",  
 age: 27  
 },  
 {  
 name: "Nicholas",  
 age: 29  
 }  
];  
people.find((element, index, array) => element.age < 28) // // {name: "Matt", age: 27}

## 二、排序方法

数组有两个方法可以用来对元素重新排序：

* reverse()
* sort()

### reverse()

顾名思义，将数组元素方向反转

let values = [1, 2, 3, 4, 5];  
values.reverse();  
alert(values); // 5,4,3,2,1

### sort()

sort()方法接受一个比较函数，用于判断哪个值应该排在前面

function compare(value1, value2) {  
 if (value1 < value2) {  
 return -1;  
 } else if (value1 > value2) {  
 return 1;  
 } else {  
 return 0;  
 }  
}  
let values = [0, 1, 5, 10, 15];  
values.sort(compare);  
alert(values); // 0,1,5,10,15

## 三、转换方法

常见的转换方法有：

### join()

join() 方法接收一个参数，即字符串分隔符，返回包含所有项的字符串

let colors = ["red", "green", "blue"];  
alert(colors.join(",")); // red,green,blue  
alert(colors.join("||")); // red||green||blue

## 四、迭代方法

常用来迭代数组的方法（都不改变原数组）有如下：

* some()
* every()
* forEach()
* filter()
* map()

### some()

对数组每一项都运行传入的测试函数，如果至少有1个元素返回 true ，则这个方法返回 true

let numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];  
let someResult = numbers.some((item, index, array) => item > 2);  
console.log(someResult) // true

### every()

对数组每一项都运行传入的测试函数，如果所有元素都返回 true ，则这个方法返回 true

let numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];  
let everyResult = numbers.every((item, index, array) => item > 2);  
console.log(everyResult) // false

### forEach()

对数组每一项都运行传入的函数，没有返回值

let numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];  
numbers.forEach((item, index, array) => {  
 // 执行某些操作  
});

### filter()

对数组每一项都运行传入的函数，函数返回 true 的项会组成数组之后返回

let numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];  
let filterResult = numbers.filter((item, index, array) => item > 2);  
console.log(filterResult); // 3,4,5,4,3

### map()

对数组每一项都运行传入的函数，返回由每次函数调用的结果构成的数组

let numbers = [1, 2, 3, 4, 5, 4, 3, 2, 1];  
let mapResult = numbers.map((item, index, array) => item \* 2);  
console.log(mapResult) // 2,4,6,8,10,8,6,4,2