数组类型方法

1.`concat`，创建一个新数组，将`array`与任何数组 或 值连接在一起。

```
let arr = ['1','2','3','4']
let arr2 = ['9']
let arr3 = arr.concat(arr2)
console.log(arr3);
//[ '1', '2', '3', '4', '9' ]
```

2.pop，删除数组的最后一个元素，返回值为最后一个元素

```
let arr = ['1','2','3','4']
var a = arr.pop();
console.log(a);// 4
```

3.shift，删除数组的第一个元素，返回值为第一个元素

```
let arr = ['1','2','3','4']
var a = arr.shift();
console.log(a);// 1
```

4.splice，将数组中指定位置的数据替换为输入的数据，返回值为替换掉的元素。在数组中的位置（起始位置），要替换的个数，替换的数值

```
let arr = ['1','2','3','4']
var a3=arr.splice(1,1,"789")
console.log(a3) // [ '2' ]
console.log(arr)// [ '1', '789', '3' ]
```

5.join，数据的链接符，返回值为用输入的连接符链接后的数组元素，类型为string

```
let arr = ['1','2','3','4']
var a5=arr.join("-");
console.log(a5);//1-2-3-4
console.log(arr);//[ '1', '2', '3', '4' ]
```

6.push，在数组的最后面添加一个数据，返回值为数组最终的长度

```
var a6=arr.push("789");     //在数组的最后面添加一个数据，返回值为数组最终的长度。
console.log(a6);// 5
console.log(arr);// [ '1', '2', '3', '4', '789' ]
```

7.unshift，在数组的最开始添加一个数据，返回值为数据的最终长度

```
let arr = ['1','2','3','4']
var a7=arr.unshift("123");      //在数组的最开始添加一个数据，返回值为数据的最终长度。
console.log(a7);//5
console.log(arr);//[ '123', '1', '2', '3', '4' ]
```

8.reverse，将数组的元素倒序排列，返回值为倒序后的数组,原数组也被倒叙

```
let arr = ['1','2','3','4']
var a8=arr.reverse();       //将数组的元素倒序排列，返回值为倒序后的数组,原数组也被倒叙。
console.log(a8);//[ '4', '3', '2', '1' ]
console.log(arr);//[ '4', '3', '2', '1' ]
```

9.valueOf，`toString`，一般的数组的方法为打印数组的内容

```
console.log(arr.valueOf());       //[ '1', '2', '3', '4' ]
console.log(arr.toString());    //1,2,3,4
```

10.sort()，方法用于对数组的元素进行排序。

我们将创建一个数组，并按字母顺序进行排序：

```
var arr = new Array(6)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
arr[3] = "James"
arr[4] = "Adrew"
arr[5] = "Martin"

document.write(arr + "<br />")
document.write(arr.sort())

George,John,Thomas,James,Adrew,Martin
Adrew,George,James,John,Martin,Thomas
//来自w3school
```

我们将创建一个数组，并按数字大小顺序进行排序：

```
function sortNumber(a,b)
{
return a - b
}
```

```
var arr = new Array(6)
arr[0] = "10"
arr[1] = "5"
arr[2] = "40"
arr[3] = "25"
arr[4] = "1000"
arr[5] = "1"

document.write(arr + "<br />")
document.write(arr.sort())

10,5,40,25,1000,1
1,5,10,25,40,1000
//来自w3school
```

11.forEach，循环遍历数组

```
carsList.forEach(item => {
    for(var i=0;i<item.product_list.length;i++){
        let data ={
            user_id:localStorage.user_id,
            shop_cart_id:item.shop_cart_id,
            index:item.product_list[i].index
        }
        delinfo.push(data)
    }
})
```

forEach() 方法用于调用数组的每个元素，并将元素传递给回调函数。

注意: forEach() 对于空数组是不会执行回调函数的。

12.map 方法返回一个新数组，数组中的元素为原始数组元素调用函数处理后的值。

map() 方法返回一个新数组，数组中的元素为原始数组元素调用函数处理后的值。

map() 方法按照原始数组元素顺序依次处理元素。

注意： map() 不会对空数组进行检测。

注意： map() 不会改变原始数组。

```
const users=res.items.map(item => ({
    url: item.html_url,
    img: item.avatar_url,
    name: item.login,
    })
);
```

```
let arr = ['1','2','3','4']
function sum(num){
  return num * 5
}
let a8 = arr.map(sum)
console.log(a8);//[ 5, 10, 15, 20 ]
```

13.copyWithin，方法用于从数组的指定位置拷贝元素到数组的另一个指定位置中。

```
array.copyWithin(target, start, end)
target  必需。复制到指定目标索引位置。
start   可选。元素复制的起始位置。
end 可选。停止复制的索引位置 (默认为 array.length)。如果为负值，表示倒数。
```

```
var fruits = ["1", "2", "3", "4", "5", "6"];
fruits.copyWithin(2,1,3);

1,2,2,3,5,6
```

14.entries() 方法返回一个数组的迭代对象，该对象包含数组的键值对 (key/value)

```
let arra = ['a','b','c','d'];
arra.entries();

[0, "a"]
[1, "b"]
[2, "c"]
[3, "d"]
```

15.every() 方法用于检测数组所有元素是否都符合指定条件（通过函数提供）。

```
every() 方法用于检测数组所有元素是否都符合指定条件（通过函数提供）。
every() 方法使用指定函数检测数组中的所有元素：
如果数组中检测到有一个元素不满足，则整个表达式返回 false ，且剩余的元素不会再进行检测。
如果所有元素都满足条件，则返回 true。
注意： every() 不会对空数组进行检测。
注意： every() 不会改变原始数组。
```

```
var ages = [32, 33, 16, 40];
function checkAdult(age) {
    return age >= 18;
}
let flag = ages.every(checkAdult);
console.log(flag) //false
```

16.fill()，方法用于将一个固定值替换数组的元素。

```
array.fill(value, start, end)
value    必需。填充的值。
start    可选。开始填充位置。
end 可选。停止填充位置 (默认为 array.length)
```

```
let arra = ['a','b','c','d'];
arra.fill("a");
a,a,a,a
```

```
fruits.fill("e", 2, 4);
a,b,e,e
```

17.filter()，方法创建一个新的数组，新数组中的元素是通过检查指定数组中符合条件的所有元素。

```
var ages = [32, 33, 16, 40];
function checkAdult(age) {
    return age >= 18;
}
let arr = ages.filter(checkAdult);
console.log(arr) //32,33,40

let goodsList = this.goodsList.filter(item=>item.checked === true)
```

18.find()，方法返回通过测试（函数内判断）的数组的第一个元素的值。

```
find() 方法返回通过测试（函数内判断）的数组的第一个元素的值。
find() 方法为数组中的每个元素都调用一次函数执行：
当数组中的元素在测试条件时返回 true 时，find() 返回符合条件的元素，之后的值不会再调用执行函数。
如果没有符合条件的元素返回 undefined
注意: find() 对于空数组，函数是不会执行的。
注意: find() 并没有改变数组的原始值。
```

```
var ages = [32, 33, 16, 40];
function checkAdult(age) {
    return age >= 18;
}
let arr = ages.find(checkAdult);
console.log(arr) //32
```

19.findIndex() 方法返回传入一个测试条件（函数）符合条件的数组第一个元素位置。

findIndex() 方法为数组中的每个元素都调用一次函数执行：
当数组中的元素在测试条件时返回 true 时，findIndex() 返回符合条件的元素的索引位置，之后的值不会再调用执行函数。
如果没有符合条件的元素返回 -1
注意：findIndex() 对于空数组，函数是不会执行的。
注意：findIndex() 并没有改变数组的原始值。

```
var ages = [32, 33, 16, 40];
function checkAdult(age) {
    return age >= 18;
}
let arr = ages.findIndex(checkAdult);
console.log(arr) //0
```

20.from() 方法用于通过拥有 length 属性的对象或可迭代的对象来返回一个数组。

from() 方法用于通过拥有 length 属性的对象或可迭代的对象来返回一个数组。
如果对象是数组返回 true，否则返回 false。
Array.from(object, mapFunction, thisValue)
object 必需，要转换为数组的对象。
mapFunction 可选，数组中每个元素要调用的函数。
thisValue 可选，映射函数(mapFunction)中的 this 对象。

```
var setObj = new Set(["a", "b", "c"]);
var objArr = Array.from(setObj);
objArr[1] == "b";
console.log(objArr[1] == "b");
console.log(objArr);
//true
//[ 'a', 'b', 'c' ]
```

21.includes() 方法用来判断一个数组是否包含一个指定的值，如果是返回 true，否则false。

arr.includes(searchElement)
arr.includes(searchElement, fromIndex)
searchElement 必须。需要查找的元素值。
fromIndex 可选。从该索引处开始查找 searchElement。如果为负值，则按升序从 array.length + fromIndex 的索引开始搜索。默认为 0。

```
[1, 2, 3].includes(2);     // true
[1, 2, 3].includes(4);     // false
[1, 2, 3].includes(3, 3);  // false
[1, 2, 3].includes(3, -1); // true
[1, 2, NaN].includes(NaN); // true
```

22.indexOf() 方法可返回数组中某个指定的元素位置。

该方法将从头到尾地检索数组，看它是否含有对应的元素。开始检索的位置在数组 start 处或数组的开头（没有指定 start 参数时）。如果找到一个 item，则返回 item 的第一次出现的位置。开始位置的索引为 0。
如果在数组中没找到指定元素则返回 -1。
array.indexOf(item,start)
tem 必须。查找的元素。
start　可选的整数参数。规定在数组中开始检索的位置。它的合法取值是 0 到 stringObject.length - 1。如省略该参数，则将从字符串的首字符开始检索。

```
var fruits=["Banana","Orange","Apple","Mango","Banana","Orange","Apple"];
var a = fruits.indexOf("Apple",4);
//6
```

23.isArray() 方法用于判断一个对象是否为数组。

如果对象是数组返回 true，否则返回 false。
Array.isArray(obj)

24.lastIndexOf() 方法可返回一个指定的元素在数组中最后出现的位置，从该字符串的后面向前查找。

如果要检索的元素没有出现，则该方法返回 -1。
该方法将从尾到头地检索数组中指定元素 item。开始检索的位置在数组的 start 处或数组的结尾（没有指定 start 参数时）。如果找到一个 item，则返回 item 从尾向前检索第一个次出现在数组的位置。数组的索引开始位置是从 0 开始的。
如果在数组中没找到指定元素则返回 -1。

array.lastIndexOf(item,start)
item　必需。规定需检索的字符串值。
start　可选的整数参数。规定在字符串中开始检索的位置。它的合法取值是 0 到 stringObject.length - 1。如省略该参数，则将从字符串的最后一个字符处开始检索。

```
var fruits=["Banana","Orange","Apple","Mango","Banana","Orange","Apple"];
var a = fruits.lastIndexOf("Apple");
//6

var fruits=["Banana","Orange","Apple","Mango","Banana","Orange","Apple"];
var a = fruits.lastIndexOf("Apple",4);
//2
```

25.reduce() 方法接收一个函数作为累加器，数组中的每个值（从左到右）开始缩减，最终计算为一个值。

```
array.reduce(function(total, currentValue, currentIndex, arr), initialValue)

function(total,currentValue, index,arr) 必需。用于执行每个数组元素的函数。
函数参数:
参数  描述
total  必需。初始值，或者计算结束后的返回值。
currentValue  必需。当前元素
currentIndex  可选。当前元素的索引
arr 可选。当前元素所属的数组对象。
initialValue  可选。传递给函数的初始值
```

```
var ages = [32, 33, 16, 40];
function getSum(total, num) {
  return total + num;
}
let sum = ages.reduce(getSum);
console.log(sum); //121
```

26.reduceRight() 方法的功能和 reduce() 功能是一样的，不同的是 `reduceRight()` 从数组的末尾向前将数组中的数组项做累加。

```
array.reduceRight(function(total, currentValue, currentIndex, arr),
initialValue)
function(total,currentValue, index,arr)
total  必需。初始值，或者计算结束后的返回值。
currentValue  必需。当前元素
currentIndex  可选。当前元素的索引
arr 可选。当前元素所属的数组对象。
initialValue  可选。传递给函数的初始值
```

27.slice() 方法可从已有的数组中返回选定的元素。

```
//slice()方法可提取字符串的某个部分，并以新的字符串返回被提取的部分
//注意： slice() 方法不会改变原始数组。
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
var citrus = fruits.slice(1,3);
//Orange,Lemon
//返回一个新的数组，包含从 start 到 end （不包括该元素）的 arrayObject 中的元素。
```

```
参数  描述
start  可选。规定从何处开始选取。如果是负数，那么它规定从数组尾部开始算起的位置。也就是说，-1 指最后一个元素，-2 指倒数第二个元素，以此类推。
end 可选。规定从何处结束选取。该参数是数组片断结束处的数组下标。如果没有指定该参数，那么切分的数组包含从 start 到数组结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素。
```

28.some() 方法用于检测数组中的元素是否满足指定条件（函数提供）

some() 方法用于检测数组中的元素是否满足指定条件（函数提供）。
some() 方法会依次执行数组的每个元素：
如果有一个元素满足条件，则表达式返回true ，剩余的元素不会再执行检测。
如果没有满足条件的元素，则返回false。
注意： some() 不会对空数组进行检测。
注意： some() 不会改变原始数组。

```
var ages = [3, 10, 18, 20];

function checkAdult(age) {
    return age >= 18;
}
let arr = ages.some(checkAdult);
//true
```

参数 描述
function(currentValue, index,arr) 必须。函数，数组中的每个元素都会执行这个函数
函数参数:
参数 描述
currentValue 必须。当前元素的值
index 可选。当前元素的索引值
arr 可选。当前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"