Array 常见方法

1. 操作方法-- 增删改查
2. 排序方法
3. 转换方法
4. 迭代方法

操作方法

增

- push()
- unshift()
- splice()
- concat() 唯一的对原数组不产生影响

```
// 1. push()在数组结尾处添加一个新元素，返回新数组的长度
        let colors = []
        let count = colors.push('red', 'green')
        console.log(colors);//['red', 'green']
        console.log(count);//2
```

```
 // 2.unshift()在数组开头添加任意多个值，返回新数组的长度
        let fruits = ["Banana", "Orange", "Apple", "Mango"]
        let counts = fruits.unshift('Lemon', 'Pear')
        console.log(fruits);
```

```
// 3. splice()向数组添加新项   返回一个包含已删除项的数组
        let c = ["Banana", "Orange", "Apple", "Mango"]
        c.splice(2, 0, 'Kiwi', 'Lemon')
        // 参数2 代表的是新添加元素的位置
        // 参数0 代表的是删除多少个元素
        // 其余的参数是指要添加的新元素
        console.log(c);//['Banana', 'Orange', 'Kiwi', 'Lemon', 'Apple',
'Mango']
        console.log(c.splice(2, 0, 'Kiwi', 'Lemon')); //[] 什么也没删除 所以返回
```

```
 // 4. concat()通过合并现有数组来创建新数组 并不会影响原有数组， 总是返回一个新数组
        // 可以使用任意数量的数组参数
        let arr1 = ["Cecilie", "Lone"];
        let arr2 = ["Emil", "Tobias", "Linus"];
        let arr3 = arr1.concat(arr2, 'Puppy')
        console.log(arr3);//['Cecilie', 'Lone', 'Emil', 'Tobias', 'Linus',
'Puppy']
```

删

- pop()
- shift()
- splice()
- slice()不会对原数组产生影响

```
 // 1. pop()从数组中删除最后一项 同时减少数组的length 返回的是被删除的项
       let fruits = ["Banana", "Orange", "Apple", "Mango"]
       fruits.pop()
       console.log(fruits);//["Banana", "Orange", "Apple"]
```

```
// 2. shift() 删除首个数组元素 同时减少数组的length 返回的是被删除的项
       let colors = ['red', 'pink', 'yellow']
       let newColors = colors.shift()
       console.log(colors) //['pink', 'yellow']
       console.log(newColors); //red
```

```
// 3. 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 color1 = ['red', 'pink', 'yellow']
       let removed = color1.splice(2, 1)
       console.log(color1);//['red', 'pink']
       console.log(removed);['yellow']
```

```
 // 4. slice()创建一个包含原有数组中一个或多个元素的新数组（用数组的某个片段切出新数组）
        // 不会影响原数组
        let fruits2 = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
        let s = fruits2.slice(1);
        // 参数1 代表从数组元素1开始 切出一段新数组
        console.log(s);//["Orange", "Lemon", "Apple", "Mango"]

        let m = fruits2.slice(1, 4)
        // 参数1 表示从数组元素1开始 截取到元素4这一段的数组元素  但4不包括
        console.log(m);// ['Orange', 'Lemon', 'Apple']
```

改

修改原数组的内容 常用splice()

```
// splice() 修改原数组的内容  经常用这个
       // 参数1 表示起始位置
       // 第二个参数 表示删除的元素个数
       // 其余的参数 表示要加入的新元素
       let colors = ['red', 'green', 'yellow']
       let count = colors.splice(1, 1, 'pink', 'blue')
       console.log(colors);//['red', 'pink', 'blue', 'yellow']
       console.log(count);// ['green']
```

查

查找元素 返回坐标或者元素值

- indexOf()
- includes()
- find()

```
// 1. 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, 6, 7]
       console.log(numbers.indexOf(4));//3
       console.log(numbers.indexOf(8)); //-1
```

```
        // 2. includes 确定数组里是否包含某指定的元素， 找到则返回true 否则false
        let numbers2 = [1, 2, 7, 0, 4]
        console.log(numbers2.includes(8)); //false
        console.log(numbers2.includes(0));//true

        // 可以指定从哪里开始搜索  第2个参数
        console.log(numbers2.includes(4, 3)); //true
```

```
         // 3. find()返回数组中第一个符合要求的元素, 否则返回undefined
         const people = [{
             name: 'Matt',
             age: 27
         }, {
             name: 'Nicholas',
             age: 30
         }]
         console.log(people.find(element => element.age > 30)); //undefined
         console.log(people.find((element, index, array) => element.age > 26)); //{name: 'Matt',age: 27}
         // 如果是找到对应元素的索引 用findIndex()
         console.log(people.findIndex(element => element.age > 28));//1
```

排序方法

数组有两个方法可以用来对元素重新排序

- reverse()
- sort()

```
// 1. sort() 以字母顺序对数组进行排序
       let fruits = ['Banana', 'Orange', 'Apple', 'Mango']
       fruits.sort()
       console.log(fruits);//['Apple','Banana', 'Orange','Mango' ]
       // 常用来判断哪个值应该排在前面
       let points = [40, 100, 1, 5]
       points.sort(function (a, b) { return a - b }) //[1, 5, 40, 100]
       console.log(points);
```

```
        // 2. reverse() 反转数组中的元素
        fruits.reverse()
        console.log(fruits); // ['Orange', 'Mango', 'Banana', 'Apple']
```

转换方法

- join()

```
 // join()方法 将一个数组（或者类数组对象）的所有元素连接成一个字符串并返回这个字符串
        // 用逗号或者指定的分隔符字符分隔字符串，如果数组只有一个元素，那么将返回该元素而不使用分隔符
        let elements = ['Fire', 'Air', 'Water']
        console.log(elements.join('*'));//Fire*Air*Water
        console.log(elements.join(' '));//Fire Air Water
```

迭代方法

都不改变原数组

- some()
- every()
- forEach()
- filter()
- map()

```
  // 1. some() 检查某些数组是否通过了测试
        // some(function(value, index, arr){})
        let numbers = [45, 4, 9, 16, 25]
        let someOver18 = numbers.some(item => item > 18)
        console.log(someOver18); //true
```

```
        // 2. every()检查所有数组是否通过测试
        // every(function(value, index, arr){})
        console.log(numbers.every(item => item > 10)); //false
```

```
        // 3. forEach()对数组每一项都运行传入的函数 没有返回值
        // arr.forEach((item, index, arr){执行某些操作})
        numbers.forEach(item => console.log(item))
```

```
     // 4. filter()包含通过所提供函数实现的测试的所有元素。 函数返回true的项会组成数组后返回
        // filter()不会改变原数组
        let newNumbers = numbers.filter(item => item > 10)
        console.log(newNumbers); //[45, 16, 25]
```

```
        // 5. map()通过对每个数组元素执行函数来创建新数组
        // 方法不会对没有值的数组元素执行函数。 不会改变原数组
        let newNumbers2 = numbers.map(item => item * 2)
        console.log(newNumbers2); // [90, 8, 18, 32, 50]
```