9、数组的扩展

- 数组的遍历
- for...of循环的使用实例
- Array.from()
- Array.of()
- 数组实例的 find() 和 findIndex()
- 数组实例的 some() 和 every()
- 数组实例的 fill()

① 数组的遍历：

```
//定义数组
let a = ["aaa","bbb","ccc"];
a.name = "zhangsan"; //添加非数字属性

//1.使用for循环遍历数组(传统方式)
for(let i=0;i<a.length;i++){

    console.log(a[i]);
}

//2.使用for...in遍历(特点是会遍历出其他非数字属性信息)
for(let i in a){
    console.log(a[i])
}

//3.使用forEach遍历数组（不支持break和continue）
a.forEach(function(v){
    console.log(v);
});
//使用箭头函数
a.forEach(v =>{
    console.log(v);
});

//4.ES6我们提供了for...of循环
//特点不会遍历非数字属性，支持break和continue的使用
for(let v of a){
    console.log(v)
}
```

② for...of循环的使用实例

- for...of 语句创建一个循环来迭代可迭代的对象。
- 在 ES6 中引入的 for...of 循环，以替代 for...in 和 forEach(),并支持新的迭代协议。
- for...of 允许你遍历Arrays(数组),Strings(字符串),Maps(映射),Sets(集合)等可迭代的数据结构等。

```
//数组对象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)。
//1.使用for...of遍历数组：
let a = ['zhangsan','lisi','wangwu'];
for(let [k,v] of a.entries()){
    console.log(k,v);
}
//0 "zhangsan"
//1 "lisi"
//1 "lisi"

//2.使用for...of遍历Maps(映射)
const iterable1 = new Map([['one','zhangsan'],['two','lisi'],
['three','wangwu']]);
for (const [key, value] of iterable1) {
    console.log(`${key} -> ${value}`);
}
//one -> zhangsan
//two -> lisi
//three -> wangwu

//Set(集合) 对象允许你存储任何类型的唯一值，这些值可以是原始值或对象。
//3.使用for...of遍历Set(集合)
const iterable2 = new Set([10,30,20,10,30]);
for (const value of iterable2) {
    console.log(value);
}
//10
//30
//20

//字符串用于以文本形式存储数据
//4.使用for...of遍历字符串
const iterable3 = 'Hello';
for (const value of iterable3) {
    console.log(value);
}
//H
//e
//l
//l
//o

//5.使用for...of遍历arguments Object(参数对象)
function demo(){
    for(const arg of arguments) {
```

```
        console.log(arg);
    }
}
demo('a','b','c');
//a
//b
//c
```

③ Array.from()

- Array.from()方法就是将一个类数组对象或者可遍历对象转换成一个真正的数组。
- 格式：Array.from(arrayLike[, mapFn[, thisArg]])
  - arrayLike：想要转换成数组的伪数组对象或可迭代对象
  - mapFn：如果指定了该参数，新数组中的每个元素会执行该回调函数；
  - thisArg：可选参数，执行回调函数 mapFn 时 this 对象。
- 返回值：是一个新的数组实例（真正的数组）

```
 let array = {
    0: 'name',
    1: 'age',
    2: 'sex',
    3: ['user1','user2','user3'],
    'length': 4
};
let arr = Array.from(array);
console.log(arr); // ['name','age','sex',['user1','user2','user3']]
```

- Array.from()的案例

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Array.from()使用实例</title>
</head>
<body>
    <ul>
        <li>zhangsan</li>
        <li>lisi</li>
        <li>wangwu</li>
    </ul>
    <script type="text/javascript">
        //获取上面的li节点对象列表
        let nlists = document.querySelectorAll("li");
        console.log(nlists); //NodeList(3) [li, li, li]
        //将NodeList列表对象转成数组，并解析出每个元素之间的内容。
```

```
        const alist = Array.from(nlists,li => li.textContent);
        console.log(alist); // ["zhangsan", "lisi", "wangwu"]
    </script>
</body>
</html>
```

④ Array.of()

Array.of()方法用于将一组值转化为数组，即新建数组，而不考虑参数的数量或类型。

```
//使用Array.of()创建数组
console.log(Array.of());  //[] 创建一个空数组
console.log(Array.of(8)); //[8] 创建只有一个元素值为8的数组
console.log(Array.of(1, 2, 3)); //[1,2,3] 创建一个值为1，2，3的数组

//以前直接使用Array创建数组
console.log(Array()); //[] 创建一个空数组
console.log(Array(4)); // [ , , , ] 创建拥有4个元素空值的数组
console.log(Array(1, 2, 3)); //[1,2,3] 创建一个值为1，2，3的数组
```

⑤ 数组实例的 find() 和 findIndex()

- find()方法，用于找出第一个符合条件的数组成员。
  - 参数是一个回调函数，所有数组成员依次执行该回调函数。
  - 直到找出第一个返回值为true的成员，然后返回该成员。
  - 如果没有符合条件的成员，则返回undefined。
- findIndex()方法的用法与find方法非常类似，返回第一个符合条件的数组成员的位置。
  - 返回成员位置，如果所有成员都不符合条件，则返回-1。

```
const data =[
    {name:'zhangsan',age:22,sex:'man'},
    {name:'lisi',age:25,sex:'woman'},
    {name:'wangwu',age:23,sex:'man'},
];
//使用find获取name属性值为lisi的信息
let s1 = data.find(function(v){
    return v['name']=='lisi'
});
//同上效果使用箭头函数
let s2 = data.find(v => v['name']=='lisi');
console.log(s1); //{name: "lisi", age: 25, sex: "woman"}
console.log(s2); //{name: "lisi", age: 25, sex: "woman"}

//使用find获取name属性值为lisi的信息
let s3 = data.findIndex(function(v){
    return v['name']=='lisi'
});
```

```
//同上效果使用箭头函数
let s4 = data.findIndex(v => v['name']=='lisi');
console.log(s3); //1
console.log(s4); //1
```

⑥ 数组实例的 some() 和 every()

- every（）和 some（）目的：确定数组的所有成员是否满足指定的测试.
  - some()方法 只要其中一个为true 就会返回true。
  - every（）方法必须所有都返回true才会返回true，哪怕有一个false，就会返回false。
- 即：every:一假即假； some:一真即真。

```
const data =[
    {name:'zhangsan',age:22,sex:'man'},
    {name:'lisi',age:25,sex:'woman'},
    {name:'wangwu',age:23,sex:'man'},
];
//使用some判断data中是否含有一条name以"wang"开头的
let s1 = data.some(v => v['name'].startsWith("wang"));
console.log(s1); //true

//使用every判断data信息中是否都是age大于20的信息。
let s2 = data.every(v => v['age']>20);
console.log(s2); //true 若有一个不符合则返回false
```

7️⃣ 数组实例的 .fill()

- fill()函数，使用指定的元素替换原数组内容，会改变原来的数组。
- 语法结构：arr.fill(value[, start[, end]])
  - value：替换值。
  - start：替换起始位置（数组的下标），可以省略。
  - end：替换结束位置（数组的下标），如果省略不写就默认为数组结束。

```
//空数组则没有替换
console.log([].fill(6));     //[]

//将数组中的值都替换成指定值6
console.log([1,2,3].fill(6));//(3) [6, 6, 6]

//从数组索引位置2开始替换成指定值6，替换到数组结束位置。
console.log([1,2,3,4,5].fill(6,2));   //(5) [1, 2, 6, 6, 6]

//从数组索引位置2开始替换到索引位置4前结束。
console.log([1,2,3,4,5].fill(6,2,4)); //(5) [1, 2, 6, 6, 5]
```