链滴

JavaScript 中的几种集合

作者：xiluotop
原文链接：https://ld246.com/article/1571225106665
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

JavaScript 中的数据集合

集合：集合是指具有某种特定性质的具体的或抽象的对象汇总而成的集体。其中，构成集合的这些对则称为该集合的; JavaScript 中也有多种不同的集合，他们也有着不同场景的使用区别，本文列举一经常使用和比较重要的一些集合数据。JS 中的集合分有三类：数组，伪数组，其他'集合对象'（也称器），本文只是粗略的将一些数据集合分类做简要说明，具体的使用可以查阅相关文档进行使用

- 数组：JS 中的 Array 实例化对象，有系列的增删改查方法可以对数据集合进行操作
- 伪数组：其结构与数组相似，可以遍历，可以看长度（length）但不能使用 Array 的各种方法对数集合进行操作，故而称为伪数组（类数组）
- 其他集合对象（容器）：其他专门用于某些场合下的对数据进行的集合操作。如：map set

本文介绍的内容

- Array -> 数组
- arguments -> 伪数组
- HTMLCollection -> DOM 节点集合 （伪数组）
- NodeList -> 节点列表 （伪数组）
- set -> 集合对象（容器）
- map -> 集合对象（容器）
- 三种类型基本遍历方法
  - for
  - for in
  - for of

原文链接：JavaScript 中的几种集合

Array 数组

JS 中，数组是一个经常使用的集合对象，JS 为我们提供了非常丰富的数组增删改查操作。诸如 push, op,shift,unshift,sort,each,indexOf,join,slice,splice,length等

- 声明一个数组
  - var array = []
  - var array = new Array(args)
- 对数组进行一些操作
  - array.push(obj) 添加一个数组到最后
  - array.pop() 弹出最后一个数组并返回弹出的结果
  - array.sort(callback) 按照一个回调函数的指定规则进行数组排序
  - array.each(function(ele,index,array){}) each 遍历一个数组
  - array.join(str) 以某个 str 将数组拼接成字符串
- 使用场景 (用于对数据集合的增删改查)：
  - 会员列表
  - 购物车列表
  - 诸如数据集合一直要发生变化的操作场景

arguments 伪数组

JS 中每个函数（Function）的内部，JS 会自动生成一个叫 arguments 的对象，

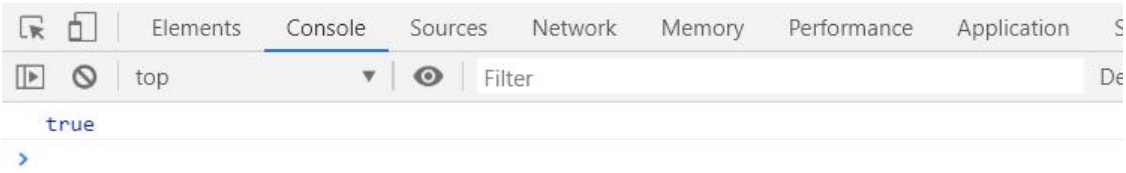
此对象的类数组结构存储的接受的各种参数，它本身也是一个对象，存有各种常用的属性，如 length, allee

- arguments 的使用及场景

  - 参数：js 允许函数调用的时候传入的参数可以不予函数声明时的参数个数相同，有时可能需要入多个参数，那么在不确定参数个数就要使用 arguments 来进行参数的获取，此时将 arguments 成一个类数组，其参数存入里面，访问时可以用数组的访问元素方式进行获取，如 arguments[0] arguments[1] arguments[arguments.length-1] 分别获取 第一个、第二个以及最后一个传入的参数，用 for + length 可是使用便利获取所有的参数。

  - callee：arguments.callee 保存了当前函数的内存地址，在不改变函数名的时候其等价于函数，均指向函数的存储空间的那个地址，验证如下：

```
function CalleeTest() {
   // 打印函数名和callee是否指向同一个地址
   console.log(CalleeTest === arguments.callee);
}
CalleeTest();
```



原文链接：JavaScript 中的几种集合

● 典型使用案例：使用 setTimerout 实现 setInterval 的循环效果

```
// 用来计数使用
var count = 1;
// 用来保存计时器，将来用于关闭
var timerId = setTimeout(function () {
console.log('第' + count++ + '次执行任务');
// 执行五次任务后关闭计时器
if (count === 6) {
    clearTimeout(timerId);
} else {
    // 利用 callee 继续设置计时器执行任务，有点类似递归的感觉~
    timerId = setTimeout(arguments.callee, 1000);
    }
}, 1000)
```

HTMLCollection DOM 节点集合（伪数组）

HTMLCollection 是 DOM 节点集合，也是 JS 一个内置对象（方法），不过他比较特殊，它不允许用这主动实例化 HTMLCollection 对象，使用 new HTMLCollection() 就会报错，但是这也是我们常会使用的一种集合。

在 DOM 操作中，我们会经常使用 document.getElementsByClassName、document.getElement ByTagName 获取一个存放 DOM 节点对象的伪数组，如果输出就会发现 getElements 方法就给我返回一个 HTMLCollection 类型的对象，此对象的数据访问结构也类似于数组，因此也可以成为类数，可以使用基本的数组访问和遍历，但是不能做其他数组的其他增删改操作。

```
> document.getElementsByTagName('script');
<- HTMLCollection [script]
    ▶ 0: script
      length: 1
    ▶ __proto__: HTMLCollection
>
```

● 特点：通过 getElements 获取的 HTMLCollection DOM元素集合可以会自动动态的随着网页 DOM 节点的变化而自动更新，之前有博主的文章提到过这个

NodeList -> 节点列表 （伪数组）

NodeList 与 HTMLCollection 非常相似，是 JS 内置对象，不能主动实例化，主动使用的情况也不多常用的场景也是通过 DOM 操作中的 document.querySelectorAll() 返回的结果，除了与 HTMLColl ction 均具有类数组的特性外，其与 HTMLCollection 的区别就是获得 DOM 节点集合是个静态快照不会随 DOM 页面动态变化而即使更新集合内容

```
> document.querySelectorAll('script');
<- ▼ NodeList [script]
    ▶ 0: script
      length: 1
    ▶ __proto__: NodeList
>
```

set 集合对象 （容器）

Set 是 JS 的一个内置集合对象，其允许你有序的存储任意类型且不重复的数据集合，如果传入有重的数据则会自动过滤掉重复的内容，这里的有序只的是**存取/获取**有序，而不是按照值有序存储，不自动排列大小顺序。

- Set 集合：
  - 初始化一个 Set 集合容器： var myset = new Set([2, 1, 3, 1, 1, '1', 4]) // 不传参则为空，返一个空的 Set 容器
  - 向 Set 添加一个值： myset.add({name:'lily'})
  - 判断 Set 是否含有某个值： myset.has(1)
  - 迭代获取 set 的所有值

```
for(var item of myset) {
   console.log(item);
}
```

  - 获取 set 容器的长度： myset.size
  - 删除 set 的一个值： myset.delete(1)
  - 清空 set 容器： myset.clear()

```
var myset = new Set([2, 1, 3, 1, 1, '1', 4]);
      console.log(myset);
   console.log(myset.add({
      name: 'lily'
   }));
   console.log(myset.has(1));
   for (var item of myset) {
      console.log(item)
   }
   console.log(myset.size);
   console.log(myset.delete('1'));
   myset.clear()
   console.log(myset);
```

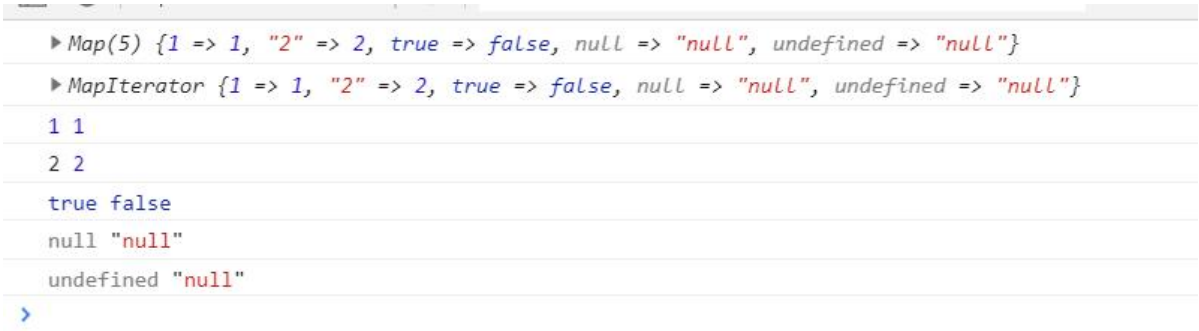
```
▶ Set(5) {2, 1, 3, "1", 4}
▶ Set(6) {2, 1, 3, "1", 4, …}
true
2
1
3
1
4
▶ {name: "lily"}
6
true
▶ Set(0) {}
```

map 集合对象 （容器）

map 是个映射集合，是一个 key/value 的集合对象，它的主要操作就是建立一张表，可以存取任意

据类型的键值对，通过 map.get(key),map.set(kev,value) 快速的进行键值的存取，主要属性和方法有 get,set,clear,entries,delete,size

```
var myMap = new Map();
myMap.set(1, 1);
myMap.set('2', 2);
myMap.set(true, false);
myMap.set(null, 'null');
myMap.set(undefined, 'null');
console.log(myMap);
console.log(myMap.entries())
for (var [key, value] of myMap.entries()) {
    console.log(key, value);
}
```



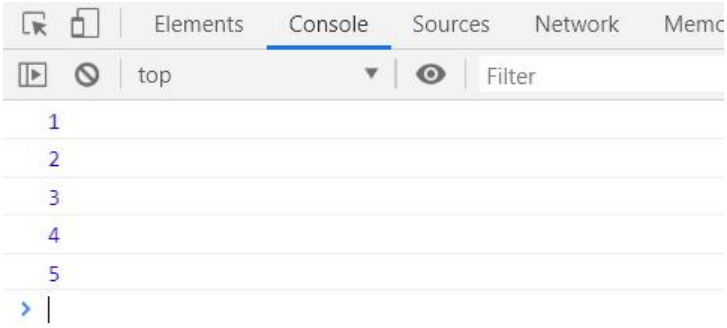
三种类型基本遍历方法

for 普通遍历

for 普通遍历主要使用场景为数组、字符串、伪数组等可以通过下标顺序访问

实例代码：

```
var arr = [1, 2, 3, 4, 5];
for (var i = 0; i < arr.length; i++) {
    console.log(arr[i]);
}
```

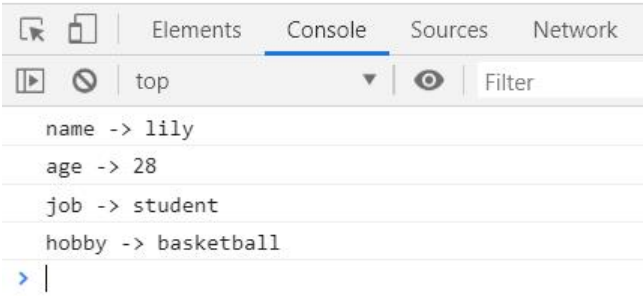


for in 遍历

for in 遍历多用于对象的属性遍历

示例代码如下：

```
var obj = {
    name: 'lily',
    age: 28,
    job: 'student',
    hobby: 'basketball',
}
for (var propname in obj) {
    console.log(propname + ' -> ' + obj[propname]);
}
```



for of 遍历

es6 中引入了 for of 迭代遍历，它可以代替普通遍历和for in 遍历，并且还能遍历诸如 map，set 等合

实例代码：

```
var arr = [1, 2, 3, 4, 5];
for (var item of arr) {
    console.log(item);
}
// entries 可以为数据集合创建一个迭代器，此时可以使用 [index, value]用来迭代遍历获取 key/valu

for (var [index, value] of arr.entries()) {
    console.log(index + ' -> ' + value);
}
var map = new Map();
map.set('name', 'lily');
map.set('age', 28);
map.set('job', 'student');
map.set('hobby', 'game');
// map 本身是一种集合，当使用 for (var item of map) 时取到的是 map 表中的每一项([key,value])
// 所以可以直接使用 [key,value] 直接迭代遍历 map 表取得所有的键值
for (var [key, value] of map) {
    console.log(key + ' -> ' + value);
}
```

```
1
2
3
4
5
0 -> 1
1 -> 2
2 -> 3
3 -> 4
4 -> 5
name -> lily
age -> 28
job -> student
hobby -> game
>
```

原文链接：JavaScript 中的几种集合