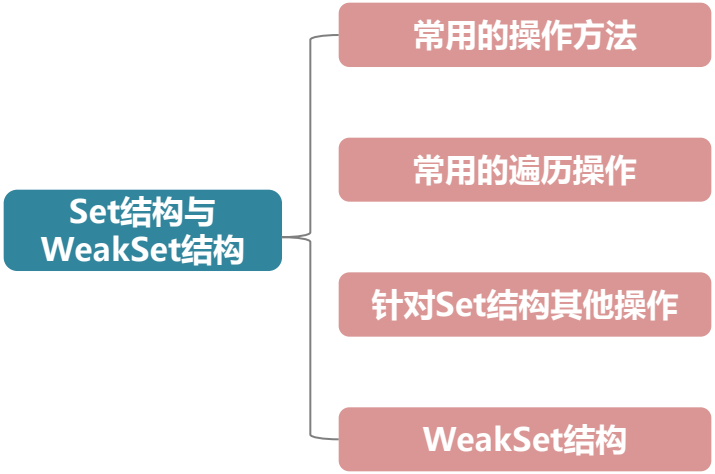
博学谷
www.boxuegu.com

第十章 Set和Map数据结构

一样的在线教育，不一样的教学品质

目录 Contents

Set数据结构



1.常用操作方法

4种常用的操作方法：

add(value)：添加某个值，返回Set结构本身；

delete(value)：删除某个值，返回一个布尔值，表示删除是否成功；

has(value) ：返回一个布尔值，表示参数是否为Set的成员；

clear( )：清除所有成员，没有返回值；

2.常用遍历操作

对阵Set结构的遍历，提供了4种方法：

keys( ) ：返回一个键名的遍历器；

values( ) ：返回一个键值的遍历器；

entries( ) ：返回一个键值对的遍历器；

forEach( ) ：使用回调函数遍历每一个成员；

注意：

Set结构没有键名，只有值；可认为键名和值都是同一个；所以，keys方法和values方法的结果一样。

3.针对Set结构其他操作

◆ **扩展运算符**

```
let s = new Set(['zs', 'ls', 'wangwu']);

console.log(...s);
```

◆ **数组去重**

```
let arr = [1, 2, 3, 3, 5, 6];

let set = [...new Set(arr)];

console.log(set);
```

4.WeakSet结构

WeakSet结构与Set类似，也是不重复的值的集合。

WeakSet结构与Set有2大区别：

1）WeakSet的**成员只能是对象**，而不能是其它类型的值；

2）WeakSet中的对象都是**弱引用**，**即垃圾回收机制不考虑WeakSet对该对象的引用**。

如果其它的对象不再使用WeakSet中的对象，垃圾回收机制会自动回收WeakSet中对象所占用的内存。

不考虑对象是否还在WeakSet中。

4.WeakSet结构

常用的操作方法：

1）add(value)：向WeakSet中添加 一个新的成员；

2）delete(value)：清除WeakSet中的指定成员；

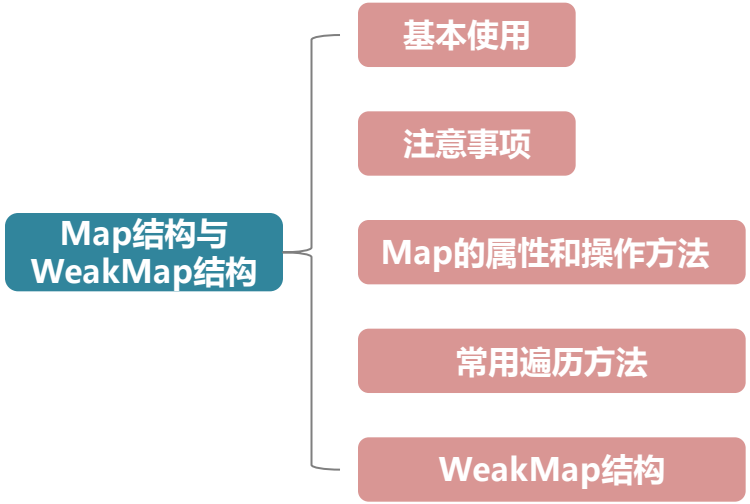
3）has(value)：返回一个布尔值，表示某个值是否在WeakSet中；

注意：在WeakSet中没有size属性和forEach方法；

因为WeakSet不能遍历，成员都是弱引用，随时可能消失，遍历机制无法保证成员的存在；可能刚刚遍历结束，成员就无效了。

目录 Contents

Map结构



1.基本使用

Map类似于对象，也是键值对的集合，“键”的类型可以是各种类型；

Map适用于需要"键值对"的数据结构。

```
let m = new Map();

m.set('age', 20);

console.log(m.get('age'));
```

说明：键的类型是字符串，键的类型也可以是其它的类型。

```
let m = new Map();
        let obj = {
            age: 20
        }
        m.set(obj, 'zhangsan');
        console.log(m.get(obj));
```

上述案例中，是以对象作为Map的key。

2.注意事项

1）如果对同一个键多次赋值，后面的值会覆盖前面的值；如果读取了一个未知键，则返回undefined；

2）只有对同一个对象的引用，Map结构才认为是同一个键。

3.Map的属性和操作方法

size属性：该属性返回Map结构的成员总数；

操作方法：

set( )方法：该方法根据对应的key设置Map结构的值，返回整个Map结构，可以使用链式写法；

get( )方法：读取key对应的值，如果找不到对应的key，则返回undefined；

delete( )方法：删除某个键，如果成功返回true，否则返回false；

clear( )方法：该方法清除所有成员，没有返回值；

4.常用遍历方法

- keys( ) : 返回键名；
- values( ): 返回值；
- entries( ): 返回所有成员；
- forEach( ): 遍历Map的所有成员；

5.WeakMap结构

WeakMap结构与Map结构类似，**唯一的区别**：WeakMap只接受对象作为键名(null除外)，不接受其他的类型的值作为键名；且键名所指向的对象不计入垃圾回收机制。

如果采用如下的方式，会出错：

```
let map = new WeakMap()
map.set('num1', 10);
console.log(map.get('num1'));
```

修改成如下形式：

```
let map = new WeakMap()
       let obj = {
           "num1": 12
       };
       map.set(obj, "成绩");
       console.log(map.get(obj));
```

WeakMap与Map在API上的区别：

1）没有keys()，values()和'entries()'这些遍历的方法，也没有size 属性。

2）不支持clear方法；WeakMap只有4个方法 get()、 set()、 has()、delete()。

6.WeakMap结构应用场景

WeakMap结构中的键名所指向的对象不计入垃圾回收机制；即键名是对象的弱引用，其所对应的对象可能会被自动回收；当对象被回收后，WeakMap自动移除对应的键值。

典型应用：

一个对应DOM元素的WeakMap结构，当某个DOM元素被清除，其所对应的WeakMap存储的内容也会自动被移除。

基本上WeakMap的**专用场合**是：

WeakMap的键所对应的对象可能会在将来消失，**WeakMap结构有助于防止内存泄漏。**

一样的在线教育，不一样的教学品质