ES6 PPT1604221: Set 与 Map 对象

Thursday, November 5, 2015 10:15 AM

Set/WeakSet 对象：

1. 生成类似于数组的无重复值集合： new Set(alterable), new WeakSet(alterable)
   new Set([2, 2, 2, 3, 3, 5]) // Set {2, 3, 5}
   new Set([5, '5']) // Set {5, "5"}
   new Set([parseInt(window), parseInt(document)]) // Set {NaN} -> 此为特例，虽然NaN并不等于自己
2. Set 实例生成的是可迭代对象：
   for (let itm of new Set([6, 7, 7, 8])) console.log(itm); // 6 // 7 // 8
3. Set 实例的属性与方法：
   new Set([3, 5, 5, 6]).size // 3 -> size 返回Set实例的成员总数
   new Set([]).add(6).add(6).add(7) // Set {6, 7} -> add 添加某个值，返回Set结构本身
   new Set([3, 5, 5, 6]).delete('5') // false -> delete 删除某个值，返回一个布尔值，表示删除是否成功
   new Set([3, 5, 5, 6]).has('6') // false -> has 返回一个布尔值，表示该值是否为Set的成员
   new Set([3, 5, 5, 6]).clear() // undefined -> clear 清除所有成员，没有返回值
   [...new Set([7, 8, 8, 9]).keys()] // [7, 8, 9] -> keys 返回一个键名的遍历器
   [...new Set([7, 8, 8, 9]).values()] // [7, 8, 9] -> values 返回一个键值的遍历器
   [...new Set([7, 8, 8, 9]).entries()] // [Array[2], Array[2], Array[2]] -> entries 返回一个键值对的遍历器
   new Set([7, 8, 8, 9]).forEach((itm, key, map) => {;}) // itm-键值, key-键名, map-实例 -> forEach 遍历各成员
4. WeakSet 类似于Set，但其实例的成员为弱引用，并且成员的值只能是对象/数组(或function)：
   new WeakSet([{a: 1}, {a: 1}, ...(obj => [obj, obj])({a: 1})]) // WeakSet {Object {a: 1}, Object {a: 1}, Object {a: 1}}
   new WeakSet([2, '2', Symbol(), undefined, null]) // 报错TypeError: Invalid value used in weak set
5. WeakSet 实例生成的是不可迭代对象：
   for (let i of new WeakSet([{}, {}])); // 报错TypeError: (intermediate value)[Symbol.iterator] is not a function
6. WeakSet 实例的属性与方法：
   (t => new WeakSet([t]).add(t).add({b: 2}))({b: 2}) // WeakSet {Object {b: 2}, Object {b: 2}} -> add 向实例添加成员
   (t => new WeakSet([t, t, {b: 2}]).delete(t))({b: 2}) // true -> delete 清除指定成员
   new WeakSet([{b: 2}]).has({b: 2}) // false -> has 返回一个布尔值，查看某值是否存在其中
7. 应用：
   a. 使用 Set 去除数组重复元素：
      [...new Set([2, 2, 2, 3, 3, 5])] // [2,3,5]
   b. 使用 WeakSet 操作 DOM 元素避免内存泄露：
      (DOMs => new WeakSet(DOMs).delete(DOMs[1]))([document.body, document.head])

Map/WeakMap 对象：

1. 扩展了Object属性/键名定义类型(Object下仅可为 string 类型, Map下可为任意类型)：new Map([...alterable])
   new Map([[6, 'six'], [6, '六'], [{a:1}, 1], [{a:1}, 'a->1']]); // Map {6 => "六", Object {a: 1} => 1, Object {a: 1} => "a->1"}
   new Map([[undefined, 'undefined1'], [undefined, 'undefined2']]); // Map {undefined => "undefined2"}
   new Map([[null, 'null1'], [null, 'null2']]); // Map {null => "null2"}
   new Map([[Symbol(), 'symbol1'], [Symbol(), 'symbol2']]); // Map {Symbol() => "symbol1", Symbol() =>

"symbol2"}
new Map([[NaN, 'NaN1'], [NaN, 'NaN2']]); // Map {NaN => "NaN2"} -> 此为特例，虽然NaN并不等于自己

2. Map 实例生成的是可迭代对象：

for (let itm of new Map([[6, 'six'], [7, 'seven'], [7, '七']])) console.log(itm); // [6, "six"] // [7, "七"]

3. Map 实例的属性与方法：

new Map([[5, 5], [5, '五'], [{}, 0]]).size // 2 -> 返回Map实例的成员总数

(t => new Map([]).set({}, 0).set({}, '空').set(t, 1).set(t, 2))({}) // Map {0, "空", 2} -> set 设定某成员，返回实例本身

(m => [m.get(5), m.get({})])(new Map([[5, 5], [5, '五'], [{}, 0]])) // ["五", undefined] -> get 获取某成员的值

new Map([[5, 5], [5, '五'], [{}, 0]]).delete({}) // false -> delete 删除某成员，返回一个布尔值，表示删除是否成功

new Map([[5, 5], [5, '五'], [{}, 0]]).has({}) // false -> has 返回一个布尔值，表示该成员是否存在

new Map([[5, 5], [5, '五'], [{}, 0]]).clear() // undefined -> clear 清除所有成员，没有返回值

[...new Map([[8, 8], [8, '八'], [{}, 0], [{}, 1]]).keys()] // [8, Object, Object] -> keys 返回一个键名的遍历器

[...new Map([[8, 8], [8, '八'], [{}, 0], [{}, 1]]).values()] // ["八", 0, 1] -> values 返回一个键值的遍历器

[...new Map([[8, 8], [{}, 0], [{}, 1]]).entries()] // [Array[2], Array[2], Array[2]] -> entries 返回一个键值对的遍历器

new Map([[{}, 0], [{}, 1]]).forEach((itm, key, map) => {;}) // itm-键值, key-键名, map-实例 -> forEach 遍历各成员

4. WeakMap 类似于Map，但其实例的成员为弱引用，并且成员的属性/键名只能是对象/数组(或function)：

new WeakMap([[{}, 0], ...(obj => [[obj, null], [obj, '空']])({})]); // WeakMap {Object {} => 0, Object {} => "空"}

new WeakMap([[2, 1], ['2', 1], [Symbol(), 1], [null, 1]]); // 报错TypeError: Invalid value used as weak map key

5. WeakMap 实例生成的是不可迭代对象：

for (let i of new WeakMap([[{}, 0]])); // 报错TypeError: (intermediate value)[Symbol.iterator] is not a function

6. WeakMap 实例的属性与方法：

(t => new WeakMap([]).set({}, 0).set({}, '0').set(t, '').set(t, '{}'))({}) // WeakMap {0, "0", "{}"} -> set 设定某成员

(t => new WeakMap([[t, '0']]).get(t))({}) // "0" -> get 获取某成员的值

(t => new WeakMap([[t, '0']]).delete(t))({}) // true -> delete 清除指定成员

new WeakMap([[{}]]).has({}) // false -> has 返回一个布尔值，表示该成员是否存在

7. 应用：

   a. 使用 Map 将一般对象 Object 生成一个可迭代对象：

```
function createMapFromObj(myObj) {
  let myMap = new Map();
  Object.keys(myObj).forEach(i => {
    myMap.set(i, myObj[i]);
  })
  return myMap;
}
createMapFromObj({a: 1, b: 2});
```

   b. 使用 WeakMap 关联 DOM 元素，DOM被删除后无需设定null而避免内存泄漏：

```
let myElement = document.getElementById('logo');
let myWeakmap = new WeakMap();
myWeakmap.set(myElement, {timesClicked: 0});
myElement.addEventListener('click', function() {
  let logoData = myWeakmap.get(myElement);
  logoData.timesClicked++;
  myWeakmap.set(myElement, logoData);
}, false);
```