【前端面试】一文搞懂Map与Set的用法和...

前言

作为前端开发人员，我们最常用的一些数据结构就是 Object、Array 之类的，毕竟它们使用起来非常的方便。往往有些刚入门的同学都会忽视 Set 和 Map 这两种数据结构的存在，因为能用 set 和 map 实现的，基本上也可以使用对象或数组实现，而且还更简单。

但是，存在必然合理，当你真正了解 Map 和 Set 之后，你就会发现它们原来时如此美好！

1.基本概念

我们先来了解以下 Map 和 Set 的基本概念，这样才能帮助我们更好的使用。虽然我们通常把这两种数据结构混合着来讲，但事实上它们它们还是有挺大区别的！

1.1 Map（字典）

想要迅速了解一个新的数据结构或 API 是，查看官网是一个不错的选择。Map 在官网上也有解释，我们一起来看下。

官网解释：

> Map 对象保存键值对，并且能够记住键的原始插入顺序。任何值（对象或者原始值）都可以作为一个键或一个值。

官网的这句话非常精炼，我们从上面这句话中总结如下几个关键词：

- 键值对
- 记住插入顺序
- 任意值作为键

一看到键值对，难免会想到对象。事实确实如此，Map 与我们平常所用的对象非常类似，它是一种**类对象**的数据结构，所以我们通常称它为 **Map 对象**。

但是我们可以把它说得更为官方一点：**Map 字典**。关于程序中字典的概念大家可以下去了解一下。

特点总结：

- Map 对象这种数据结构和和对象类型，都已键值对的形式存储数据，即 key-vlue 形式。

- Map 对象存储的数据是有序的，而我们平常使用的对象是无序的，所以通常当我们需要使用对象形式（键值对）存储数据且需要有序时，采用 Map 对象进行存储。
- Map 对象的键值可以是任意类型，我们平时使用的对象只能使用字符串作为键。

1.2 Set（集合）

和 Map 类似，我们同样先来看一看官网是怎么解释 Set 这个数据结构的。

官网的解释：

> Set 对象允许你存储任何类型的唯一值，无论是原始值或者是对象引用。

Set 的解释比 Map 的解释还要精炼，我们从中提取出几个关键词：

- 任何类型
- 唯一值

上面关键词中我们需要重点关注“唯一值”，这说明使用 Set 存储的数据是不会重复的，除此之外，Set 也是一个对象，但是它是一个**类数组**对象，也就是说它长得像数组，我们通常直接称它为 Set 对象。

当然也可以官方一点的称它：**Set 集合**。

特点总结：

- Set 对象是一个类数组对象，它长得就很像数组。
- Set 对象存储的值是不重复的，所以我们通常使用它来实现数组去重。
- Set 对象存储的数据不是键值对的形式，而且它可以存储任何类型的数据。

2.基本使用

我们平常使用 Array 或者 Object 的时候，都是直接采用[变量] = []、[变量] = {}的形式来进行初始化。而这里我们所讲的 Map 和 Set 数据结构它们都是以构造函数的形式出现的，所以我们通常使用 new Set()或者 new Map()的形式初始化的。

2.1 Map 基本使用

初始化 map 对象：

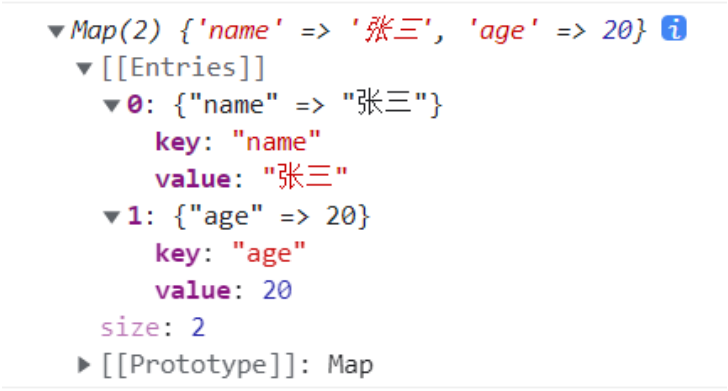
```
let myMap = new Map();
```

初始化 map 时传入数据：

由于 Map 对象是一个构造函数，所以我们在初始化的时候可以传入默认数据的，只不过我们需要注意传入默认数据的格式，它默认接收一个二维数组。

```
let defaultMap = new Map([['name', '张三'], ['age', 20]]);
```

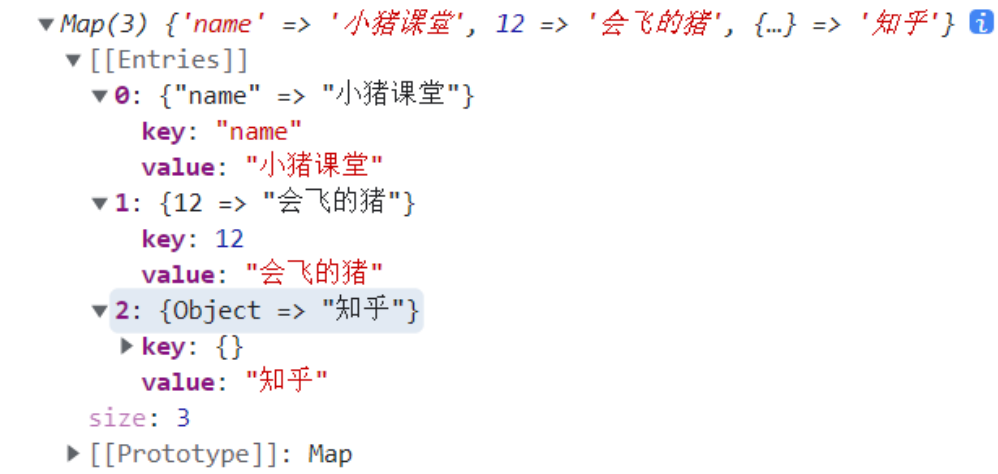
打印出来看看结果：



插入数据：

```
myMap.set('name', '小猪课堂'); // 字符串作为键
myMap.set(12, '会飞的猪'); // number 类型作为键
myMap.set({}, '知乎'); // 对象类型作为键
```

我们先打印出来看看结果：



获取长度：

我们传统的对象可以通过 Object.key().length 来获取对象长度，而 map 对象自带 size 属性获取对象长度。

```
let myMapSize = myMap.size;
```

获取值：

```
let objKey = {};
myMap.set('name', '小猪课堂'); // 字符串作为键
myMap.set(12, '会飞的猪'); // number 类型作为键
myMap.set(objKey, '知乎'); // 对象类型作为键

let name = myMap.get('name');
let age = myMap.get(12);
let any = myMap.get(objKey);

console.log(name, age, any); // 小猪课堂 会飞的猪 知乎
```

上段代码中需要注意的是不能使用 myMap.get({})的形式获取数据，因为 objKey!=={}。

删除某个值：

```
myMap.delete('name');
```

判断某个值是否存在：

```
myMap.has('name'); // 返回 bool 值
```

2.2 Set 基本使用

Set对象的使用方式和Map对象的使用方式非常的类似，只不过存储的数据格式不一样罢了。这里需要注意的Set对象存储的不是键值对形式，它只存储了值，没有键，就和数组类似。

初始化Set对象：

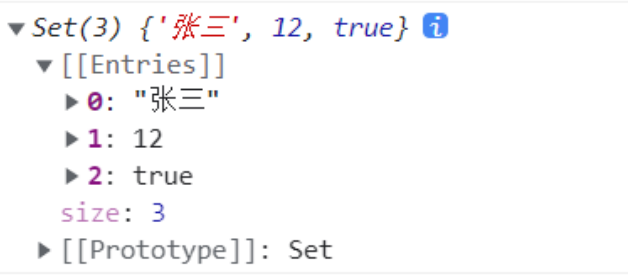
```
let mySet = new Set();
```

初始化Set对象带有默认值：

和Map类似，Set初始化时也可以初始化默认数据。

```
let defaultSet = new Set(['张三', 12, true]);
```

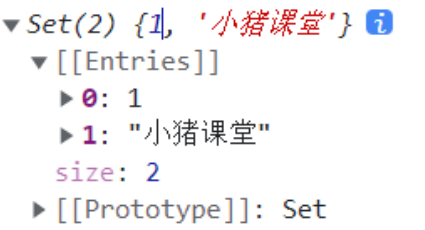
一起来看看输出结果：



插入数据：

```
mySet.add(1);
mySet.add('小猪课堂');
```

打印结果：



获取长度：

```
let mySetSize = mySet.size;
```

获取值：

由于Set对象存储的不是键值对形式，所以未提供get方法获取值，我们通常遍历它获取值：

```
mySet.forEach((item) => {
```

```
  console.log(item)
})
```

删除某个值：

```
mySet.delete(1);
```

判断某个值是否存在：

```
mySet.has(1); // 返回Boolean值
```

3.Map和Set区别

如果我们学会了它们两者如何使用，或多或少都知道它们的区别在哪里，我们这里为大家总结一下它们的区别要点：

- Map和Set查找速度都非常快，时间复杂度为O(1)，而数组查找的时间复杂度为O(n)。
- Map对象初始化的值为一个二维数组，Set对象初始化的值为一维数组。
- Map对象和Set对象都不允许键重复（可以将Set对象的键想象成值）。
- Map对象的键是不能改的，但是值能改，Set对象只能通过迭代器来更改值。

4.使用场景介绍

4.1 Set对象使用场景

数组去重

这是大家很熟悉的一种场景，使用Set对象的唯一性值特性方便的给我们数组去重。

代码如下：

```
let arr = [1, 2, 3, 4, 5, 6, 3, 2, 5, 3, 2];
console.log([...new Set(arr)]); // [1, 2, 3, 4, 5, 6]
```

需要注意的是Set对象是一个类数组，我们使用...扩展运算符将一个类数组转化为了一个真正的数组。

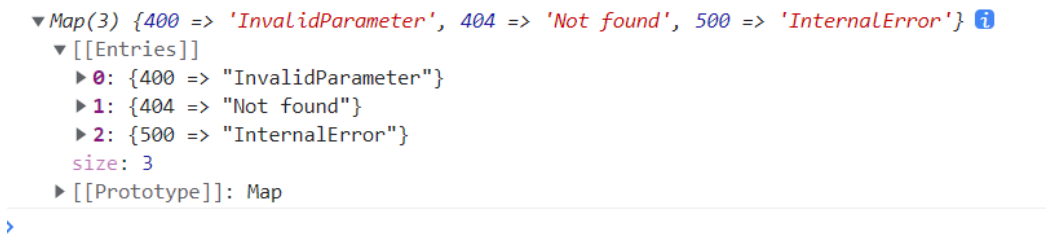
4.2 Map对象使用场景

数字类型充当键

代码如下：

```
let errors = new Map([
  [400, 'InvalidParameter'],
  [404, 'Not found'],
  [500, 'InternalError']
]);
console.log(errors);
```

输出结果：



我们可以使用Map对象建立一个请求状态码对象字典，因为状态码是数字类型，所以使用Map对象很合适。

除了该场景外，如果需要保证对象的顺序，那么也是可以使用Map对象的。

5.思考点

前面我们说Set和Map的插入删除效率为什么很高呢?

这里简单讲一下，更加深入需要大家自己下去好好学习一下数据结构了。

简述原因：

map和set存储的所有元素都是以节点的方式来进行存储的，这种节点结构和链表有点类似。我们都知道链表的特点是插入和删除都非常快，时间复杂度为O(1)，两个节点通过指针相连，删除或者增加元素时，我们只是重新更改了指针的指向，不想数组那样，掺入或删除之后需要重新排序。

总结

Set对象和Map对象有很多优点的，比如说性能比较好等等，我们需要一一去体会它们的优缺点。你不如在每次创建数据结构之前想一想：**使用Object更好还是Map更好呢？使用Array更好还是Set更好呢？**当然，如果深究Set和Map底层原理之后，你会发现它们的实现原理就是**红黑树。**