**Map在实际开发中的应用**

Map对象在JavaScript（ES5标准）中虽然没有这个对象信息，但是在实际项目开发中是我们非常常用的一个对象信息，例如

Var map={

id:’12’,

val:’Map对象’

}

我们从这段代码中能看到，这个对象其实是键值对的方式出现的也就是key-value模式，我们在代码中的使用也非常简单 map.id 就可以获取到对应的id信息，像这种定义的方式一般称为对象，id和val是这个对象的属性。也就是说在JavaScript（ES5标准）中没有真正的Map对象，我们在JavaScript中该如何使用Map来实现我们的代码开发呢？

其实比较简单就是，当我们在使用Map时可以给他定义为一个对象，属性和值就是键值对，我们在使用的时候可以直接使用“.”获取对应的数据也可以使用map[‘id’]的方式来获取到对应为的id数据，就类似Java中的Map对象一样。既然我们知道在JavaScript（ES5标准）中如何定义Map，那就让我们来看一下这种使用的模式能给我们解决怎样的问题。

## Map解决多层循环嵌套

解决多层循环嵌套问题，关于多层循环嵌套调用的例子大家都接触过，那如何解决多层循环嵌套问题呢？这种的问题是非常的麻烦的，那我们可以使用Map来解决对应的问题，比如说,下面的两个数组中的数据我们想让这两组数据合并为一组，两组中id是一一对应的：

var data=[

{id:1,val:”1”},

{id:2,val:”2”},

.....

{id:10000,val:”10000”},

{id:10001,val:”10001”},

];

var type=[

{id:1,type:”1”},

{id:2, type:”2”},

.....

{id:10000, type:”3”},

{id:10001, type:”4”},

]

面对这种情景我们该怎么办呢？我们最常用的解决方式是使用循环嵌套，但是不知道大家知不知道有这样的一个问题，当循环嵌套的数据越大时耗费的时间越长，我们常规的解决方案是这样子的：

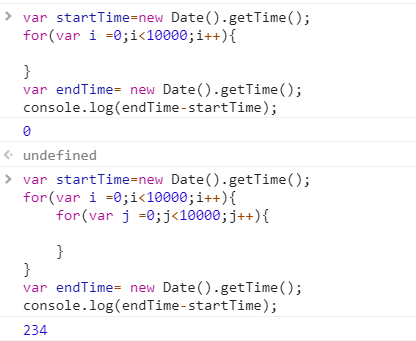


图1

从上图1的代码中我们能看出，单纯的for循环1W次耗费的时间接近于0ms，但是如果我们使用两次for循环遍历耗费的时间是234ms，有同学说，这也很快了。但是如果我们处理的数据远远大于这个数值那是怎么样的场景呢，我们就看这样的一组数据：

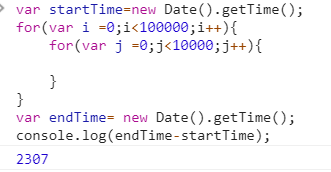
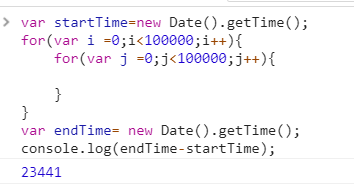
 

图2 图3

从图2和图3中我们能看出，当我们把数据值放大，循环遍历消耗的时间是跟我们的数据成正比的。面对这样的问题我们该如何处理呢？ 最简单的方式就是使用Map对象来解决此类的问题，我们来看这组示例：

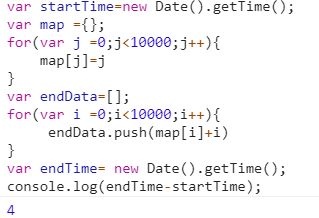
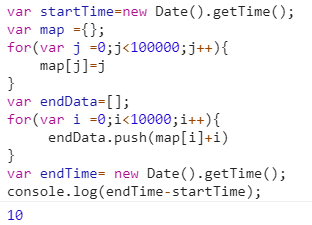
 

图4 图5

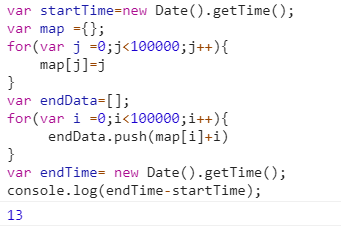


图6

通过这组数据我们能看出，当我们遍历1W条数据时我们的耗时是4ms，当我们把数值调大，每增加10倍的数据处理时间不会成10倍的时间值增长，这样我们在处理更多条数据的时候就比较方便，且效率更高；

## Map解决数据添加和删除

在日常的代码开发中我们还会遇到一种特殊的情况，就是动态的给数组中添加和删除数据，这种操作需要对数组进行添加和删除，操作数组的添加和删除比较繁琐，且不容易确定你要删除的数据到底是那条，需要每删除一次就要遍历一次数据，因此这种操作对于开发者来说是非常头疼的，那如何解决此类的问题？这里我们还是使用Map来解决此类的问题， 首先我们先来看这样的一个例子，我们想开发项目的时候，遇到一个选座问题如图7。

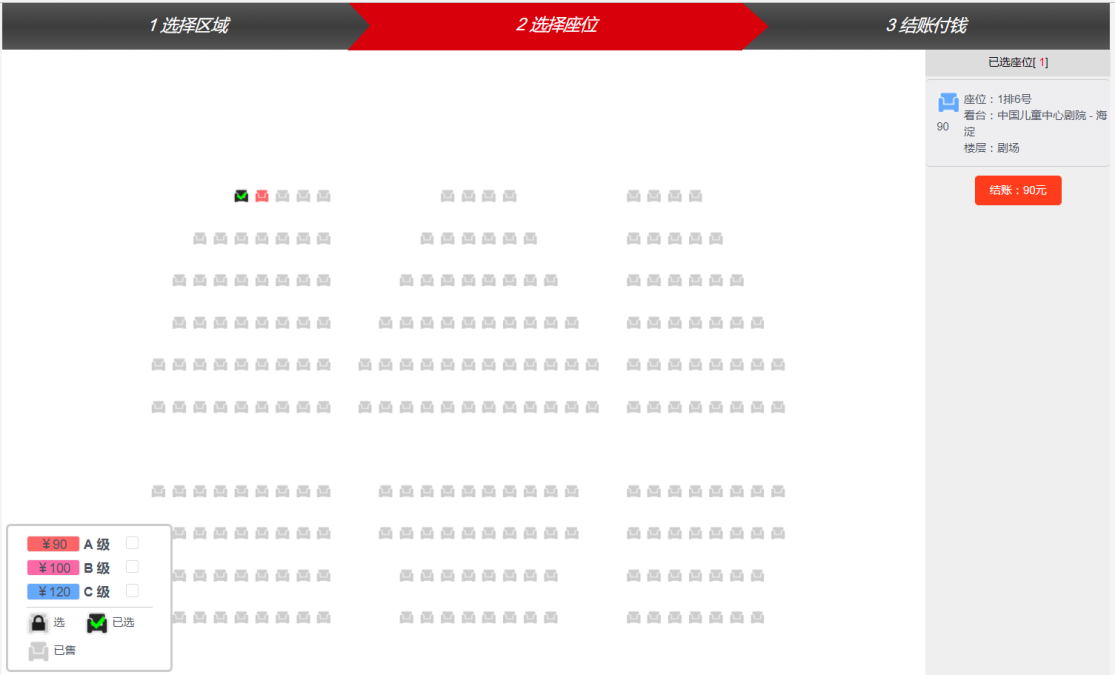


图7

从这张图中我们能看出，当我们选中一个座位后，右侧结算区会出现一个座位的信息，当我们点击的座位数越多，我们右侧选座区显示的数据的内容越多，那当我们取消其中的一个座位后，我们右侧结算区也要删除对应的一组数据，面对这样的问题我们将如何进行开发呢？这里我们就不能使用数组进行开发， 因为数组在这里虽然很方便添加数据，但是不方便删除对应的数据，我们在这里使用Map进行开发，可以灵活的进行增、删、改操作。且要注意的是使用Map进行座位信息的开发时，使用的Key值必须是唯一的，不然你的数据会被覆盖且最后获取的数据长度也会变短。

那我们来看一下如何使用Map进行选座代码的处理，首先我们看这样的代码，

var pojoMap={

1:{id:1,val:'01座'},2:{id:2,val:'02座'},3:{id:3,val:'03座'},4:{id:4,val:'04座'},

}

通过这个对象我们能看出来，pojoMap有四条数据信息，当我们想添加上数据的时候使用pojoMap[‘id’]=val 就可以添加和修改数据，使用delete pojoMap[‘id’]就可以删除数据，具体操作如图8.

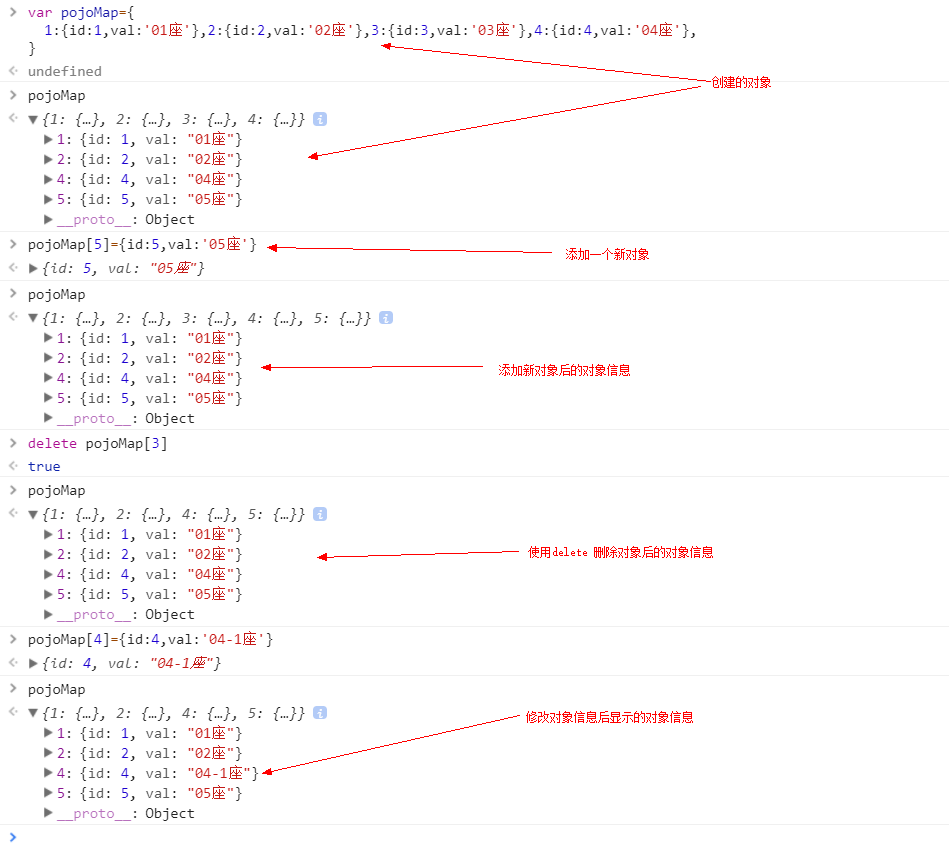


图8

## Map的遍历

最后我们点击结算按钮的时候就可以把这个Map对象传递或者发送给后台，现在又来了一个问题就是，如何对Map对象进行遍历呢？其实方法有很多种，比如：foreach方法，forIn等都是可以的，那我们来简单的了解一下在JavaScript中如何遍历Map对象吧。

我们还是使用上面的pojoMap对象来进行遍历，首先我们先来看一下foreach如何来遍历Map对象，其实使用foreach遍历map对象还是比较简单的

var pojoMap={

1:{id:1,val:'01座'},2:{id:2,val:'02座'},3:{id:3,val:'03座'},4:{id:4,val:'04座'},

};

Object.keys(pojoMap).forEach(function(key){

console.log(key)

})

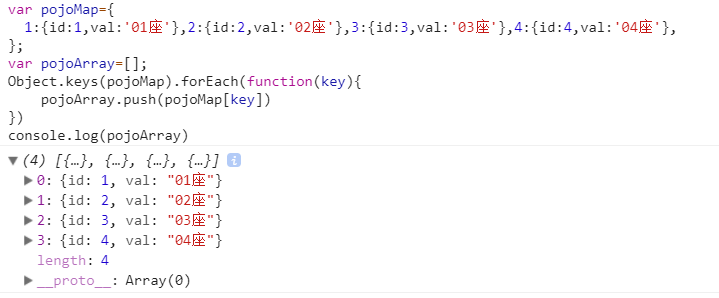
从上面的代码中我们可以看到，我们调研Object.keys这个方法将pojoMap对象所有的Key值都拿到了然后再新建一个数组使用push方法将数据添加到新的对象就可以了，详细代码如下图（图9）：  
 

图9

使用forIn进行数据打印的代码如下：

for(var key in Object.keys(pojoMap)){

....

}

综上可以看出我们在进行数据打印的时候实际上使用的是Object对象中的方法将对应的数据进行转化后得出的，其实我们在得到Map对象的数据的时候可以调用Object.values(pojoMap)这个方法就可以将我们想得到的数据拿到，具体打印的数据如下

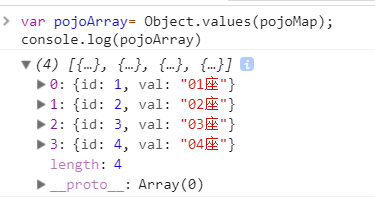


图10

如果一次性想得到Map对象的键值对的话可以使有Object.entries(pojoMap)来拿到对应的数据信息。

综上是关于Map在JavaScript（ES5标准）中的使用，接下来让我们看一下，在ES6中Map会有什么样式的改变。

## ES6中Map的应用

其实在ES6标准中，JavaScript是增加了Map对象这一概念的，极大的方便了我们的使用，ES6中的Map 跟我们在ES5中是用的Map是不一致的，我们在ES5中使用的Map实际上就是一个Object对象，而在ES6中是有单独的Map类型的，那我们看一下在ES6中Map的操作；

在ES6中Map的增加、删除、修改的代码如下：

let pojoMap=new Map();

pojoMap.set(1,{id:1,val:'01座'})

pojoMap.set(2,{id:2,val:'02座'})

pojoMap.set(3,{id:3,val:'03座'})

pojoMap.set(4,{id:4,val:'04座'})

console.log(pojoMap);

使用set添加数据，显示的效果如图11.

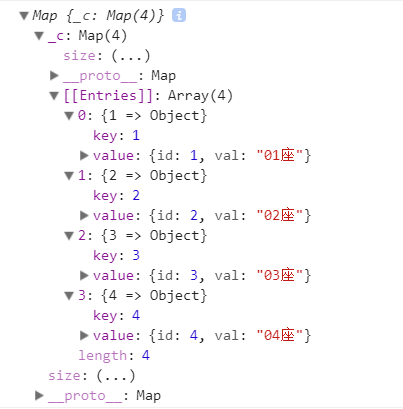


图11

获取数据使用get方法获取数据例如：console.log(pojoMap.get(1))，使用Map删除数据也是非常简单的，直接调用delete方法pojoMap.delete(‘key’)就可以将数据删除，图12

是调研删除方法后打印出的数据信息：

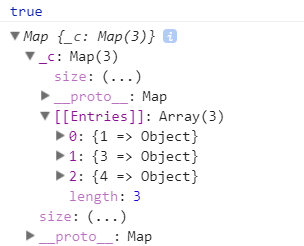


图12

Map 结构原生提供三个遍历器生成函数和一个遍历方法。

keys()：返回键名的遍历器。

values()：返回键值的遍历器。

entries()：返回所有成员的遍历器。

forEach()：遍历 Map 的所有成员。

主要使用以上四种方式，还是直接上代码吧，首先我们需要创建一个对象map，然后使用遍历方法进行数据打印如代码：

const map = new Map([

['id', '1'],

['val', '01座'],

]);

for (let key of map.keys()) {

console.log(key);

}

// id

// val

for (let value of map.values()) {

console.log(value);

}

// 1

// 01座

for (let item of map.entries()) {

console.log(item[0], item[1]);

}

// id 1

// val 01座

// 或者

for (let [key, value] of map.entries()) {

console.log(key, value);

}

// id 1

// val 01座

// 等同于使用map.entries()

for (let [key, value] of map) {

console.log(key, value);

}

// id 1

// val 01座

map.forEach(function(value, key, map) {

console.log("Key: %s, Value: %s", key, value);

});

//Key: id, Value: 1

//Key: val, Value: 01座

通过以上代码我们发现，使用Map要比使用我们自定义的Map要方便很多，有很多方法和属性是已经预置好的， 不需要我们进行转义后再获得，而且使用非常的灵活、高效。关于Map的其他属性和使用方法还需要各位同学在下面慢慢的学习。