**介绍**

迭代器模式(Iterator)：提供一种方法顺序一个聚合对象中各个元素，而又不暴露该对象内部表示。

迭代器的几个特点是：

1. 访问一个聚合对象的内容而无需暴露它的内部表示。
2. 为遍历不同的集合结构提供一个统一的接口，从而支持同样的算法在不同的集合结构上进行操作。
3. 遍历的同时更改迭代器所在的集合结构可能会导致问题（比如C#的foreach里不允许修改item）。

**正文**

一般的迭代，我们至少要有2个方法，hasNext()和Next()，这样才做做到遍历所有对象，我们先给出一个例子：

var agg = (function () {  
 var index = 0,  
 data = [1, 2, 3, 4, 5],  
 length = data.length;  
  
 return {  
 next: function () {  
 var element;  
 if (!this.hasNext()) {  
 return null;  
 }  
 element = data[index];  
 index = index + 2;  
 return element;  
 },  
  
 hasNext: function () {  
 return index < length;  
 },  
  
 rewind: function () {  
 index = 0;  
 },  
  
 current: function () {  
 return data[index];  
 }  
  
 };  
} ());

使用方法和平时C#里的方式是一样的：

// 迭代的结果是：1,3,5  
while (agg.hasNext()) {  
 console.log(agg.next());  
}

当然，你也可以通过额外的方法来重置数据，然后再继续其它操作：

// 重置  
agg.rewind();  
console.log(agg.current()); // 1

**jQuery应用例子**

jQuery里一个非常有名的迭代器就是$.each方法，通过each我们可以传入额外的function，然后来对所有的item项进行迭代操作，例如：

$.each(['dudu', 'dudu', '酸奶小妹', '那个MM'], function (index, value) {  
 console.log(index + ': ' + value);  
});  
//或者  
$('li').each(function (index) {  
 console.log(index + ': ' + $(this).text());  
});

**总结**

迭代器的使用场景是：对于集合内部结果常常变化各异，我们不想暴露其内部结构的话，但又响让客户代码透明底访问其中的元素，这种情况下我们可以使用迭代器模式。