JavaScript原有的表示“集合”的数据结构，主要是数组（Array）和对象（Object），ES6又添加了Map和Set。这样就有了四种数据集合，用户还可以组合使用它们，定义自己的数据结构，比如数组的成员是Map，Map的成员是对象。这样就需要一种统一的接口机制，来处理所有不同的数据结构。

遍历器（Iterator）就是这样一种机制。它是一种接口，为各种不同的数据结构提供统一的访问机制。任何数据结构只要部署Iterator接口，就可以完成遍历操作（即依次处理该数据结构的所有成员）。

Iterator的作用有三个：**一是**为各种数据结构，提供一个统一的、简便的访问接口；**二是**使得数据结构的成员能够按某种次序排列；**三是**ES6创造了一种新的遍历命令`for...of`循环，Iterator接口主要供`for...of`消费。

Iterator的遍历过程是这样的。

（1）创建一个指针对象，指向当前数据结构的起始位置。也就是说，遍历器对象本质上，就是一个指针对象。

（2）第一次调用指针对象的`next`方法，可以将指针指向数据结构的第一个成员。

（3）第二次调用指针对象的`next`方法，指针就指向数据结构的第二个成员。

（4）不断调用指针对象的`next`方法，直到它指向数据结构的结束位置。

每一次调用`next`方法，都会返回数据结构的当前成员的信息。具体来说，就是返回一个包含`value`和`done`两个属性的对象。其中，`value`属性是当前成员的值，`done`属性是一个布尔值，表示遍历是否结束。