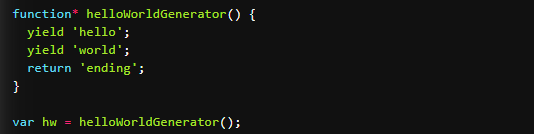
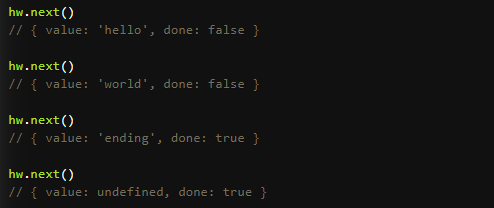
参考摘抄于：<https://es6.ruanyifeng.com/#docs/generator>

1、基本概念

Generator函数是ES6提出的异步编程解决方案，从语法上，Generator是一个状态机，封装了多个内部状态。返回一个遍历器对象，可以依次遍历除Generator中的所有状态。

形式上：一是function关键字与函数名之间有一个星号(由于ES6语法并未规定星号在什么位置，所以只要星号出现在function关键字和函数名之间就都可以通过，但是一般紧跟在function关键字后)，二是函数体内部使用yield关键字定义状态。



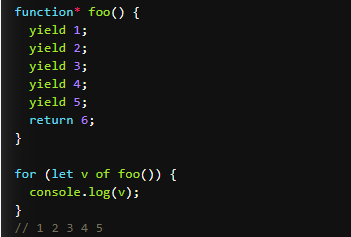
注意：使用()调用Generator函数后，该函数并不会执行，返回的也不是函数运行结果，而是一个指向函数内部状态的指针对象，也就是遍历器对象（interator Object）。必须调用函数的next()方法，使得指针移向下一个状态。也就是说，每次调用next()方法，内部指针会从函数顶部或上一次执行停止的地方开始执行，直到遇到下一个yield表达式或return语句结束。换言之，Generator函数是分段执行函数，yield可以暂停函数执行，next可恢复。

注意：返回对象里，done属性表示遍历是否执行完毕。

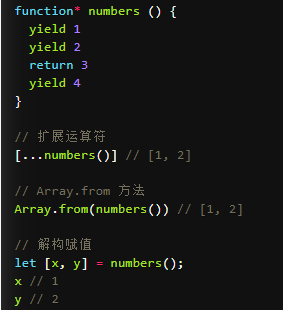
next方法的参数，yield表达式本身没有返回值，或者说返回值为undefined。next方法可传入一个参数，该参数的值被当做上一个yield表达式的返回值。

2、for…of循环

for…of循环会自动遍历由Generator函数生成的Interator对象，且不需要执行next()方法。

注意：一旦next返回对象的done属性为true时，for…of循环就会终止，且不包含该返回对象。故return 语句返回的6并不在返回值中。

除了for…of循环外，扩展运算符(…)、解构赋值，Array.from方法内部调用的，都是遍历器接口。这意味着，他们都可以把Generator函数返回的遍历器对象作为参数。



3、Generator.prototype.throw()方法

Generator函数返回的遍历器对象都有一个throw方法，可以在函数外部抛出错误，在函数内部捕获。

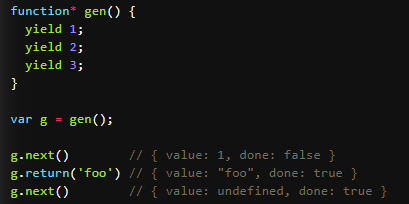
throw方法接收一个参数，该参数会被catch语句接收，建议抛出Error对象实例。throw方法被捕获后，会附带执行一次yield表达式，即执行一次next()方法。且一旦函数内部部署了try…catch代码，那么遍历器throw方法抛出的错误，并不影响下一次遍历。

遍历器对象的throw方法可以在Generator函数内部捕获，但是使用throw命令全局抛出的异常，只能在函数外部使用catch语句进行捕获。



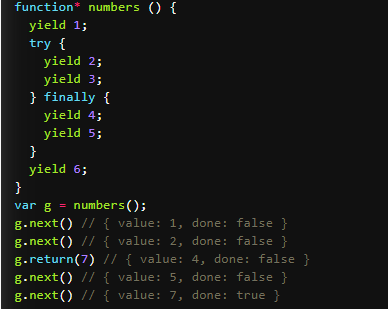
4、Generator.prototype.return()

该函数可以返回指定的值，并且终结遍历Generator函数。



如果return方法调时，不传入参数，则返回值的value属性为undefined。

若Generator函数内部存在try…finally代码块，且正在执行try代码块，那么return()方法会立刻进入finally代码块，直到finally代码块执行结束。

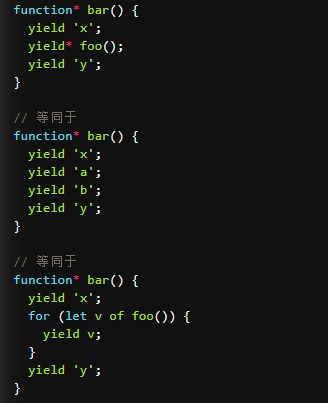


注意：g.return(7)的返回结果value值并不是7，而是finally代码块的第一次返回值；且在finally代码块执行结束后，才返回了return()方法指定的返回值（value为7的结果）。

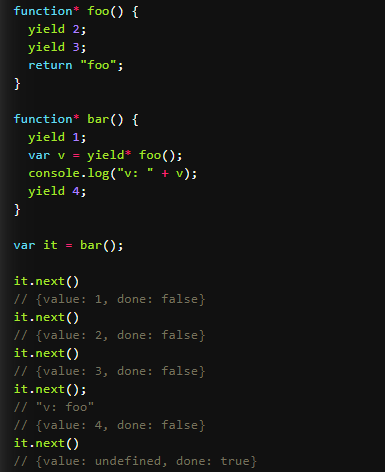
5、yield\* 表达式

如果在一个Generator函数内部调用另一个Generator函数，需要在前者的函数体内部手动遍历后者。yield\* 为此场景提供了解决方案。

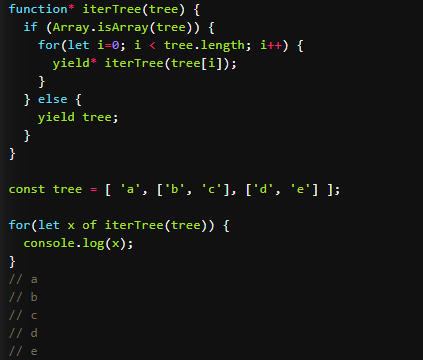




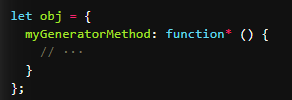
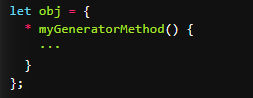
如果被代理的Generator函数中有return语句，那么就可以向代理他的Generator函数返回数据。



使用场景，yield\* 可以方便的取出嵌套数组中的每一个元素。



6、作为对象属性的Generator函数



二者等价。