function\* 这种声明方式(function关键字后跟一个星号）会定义一个生成器函数 (generator function)，它返回一个 Generator 对象。

你也可以定义 生成器函数 使用构造函数 GeneratorFunction 和一个 function\* expression 。

语法

Edit

function\* name([param[, param[, ... param]]]) { statements }

name

函数名

param

要传递给函数的一个参数的名称，一个函数最多可以有255个参数。

statements

普通JS语句。

描述

Edit

生成器函数在执行时能中途退出，后面又能重新进入继续执行。而且在函数内定义的变量的状态都会保留，不受中途退出的影响。

调用一个生成器函数并不会马上执行它里面的语句，而是返回一个这个生成器的迭代器（iterator）对象。当这个迭代器的 next() 方法被首次（后续）调用时，其内的语句会执行到第一个（后续）出现yield表达式的位置为止，该表达式定义了迭代器要返回的值，或者被 yield\*委派至另一个生成器函数。next()方法返回一个对象，这个对象包含两个属性：value 和 done，value 属性表示本次 yield 表达式的返回值，done 属性为布尔类型，表示生成器是否已经产出了它最后的值，即生成器函数是否已经返回。

调用 next() 方法时，如果传入了参数，那么这个参数会取代生成器函数中对应执行位置的 yield 表达式（整个表达式被这个值替换）

当在生成器函数中显式 return 时，会导致生成器立即变为完成状态，即调用 next() 方法返回的对象的 done 为 true。如果 return 了一个值，那么这个值会作为下次调用 next() 方法返回的 value 值。

示例

Edit

简单示例

function\* idMaker(){

var index = 0;

while(index<3)

yield index++;

}

var gen = idMaker();

console.log(gen.next().value); // 0

console.log(gen.next().value); // 1

console.log(gen.next().value); // 2

console.log(gen.next().value); // undefined

// ...

yield\*的示例

function\* anotherGenerator(i) {

yield i + 1;

yield i + 2;

yield i + 3;

}

function\* generator(i){

yield i;

yield\* anotherGenerator(i);

yield i + 10;

}

var gen = generator(10);

console.log(gen.next().value); // 10

console.log(gen.next().value); // 11

console.log(gen.next().value); // 12

console.log(gen.next().value); // 13

console.log(gen.next().value); // 20

传递参数

function\* logGenerator() {

console.log(yield);//首次执行的中断处

console.log(yield);

console.log(yield);

}

var gen = logGenerator();

// 首次调用 next() 会执行到第一个 yield 语句处

gen.next();

gen.next('pretzel'); // pretzel

gen.next('california'); // california

gen.next('mayonnaise'); // mayonnaise

显式返回

function\* yieldAndReturn() {

yield "Y";

return "R";//显式返回处

yield "unreachable";

}

var gen = yieldAndReturn()

console.log(gen.next()); // { value: "Y", done: false }

console.log(gen.next()); // { value: "R", done: true }

console.log(gen.next()); // { value: undefined, done: true }

生成器函数不能当构造器使用

function\* f() {}

var obj = new f; // throws "TypeError: f is not a constructor"

Chrome、firefox、opera支持良好，safari和IE不支持