版本 4.6.0

4.0.0后，co()返回一个Promise

if (!gen || typeof gen.next !== 'function') return resolve(gen);

参数是一个值或Promise对象，直接调用reslove()方法

if (typeof gen === 'function') gen = gen.apply(ctx, args);

ret = gen.next(res);

function next(ret) {

if (ret.done) return resolve(ret.value); //yield到最后，{value:undefind,done:true}

var value = toPromise.call(ctx, ret.value);

if (value && isPromise(value)) return value.then(onFulfilled, onRejected);

return onRejected(new TypeError('You may only yield a function, promise, generator, array, or object, '

+ 'but the following object was passed: "' + String(ret.value) + '"'));

}

参数是生成器，先执行方法，然后执行next()(yield后的函数)

toPromise()方法 转化yield的值到Promise

if (!obj) return obj;

if (isPromise(obj)) return obj;

if (isGeneratorFunction(obj) || isGenerator(obj)) return co.call(this, obj);

if ('function' == typeof obj) return thunkToPromise.call(this, obj);

if (Array.isArray(obj)) return arrayToPromise.call(this, obj); //promise.all()

if (isObject(obj)) return objectToPromise.call(this, obj); //得到所有键，调用toPromise函数，push进一个数组，promise.all()

return obj;

//调用实例

const co = require("co")

var foo = co(1)

Promise(1)

var foo = co(function (){

console.log(1)

})

1

Promise(undefied)

var foo = co(function \*(){

console.log(1)

})

1

function foo3(a,b){

console.log(a)

// console.log(b())

return [1] //写数字会报错 co的next方法

// return b()

}

function foo4(a){

return function(b){

// console.log(c)

console.log(b)

return foo3(a,b)

}

}

var foo5 = foo4(4)

// console.log(foo5(1))

var foo = co(function \*(){

console.log(1)

// var c = yield [foo1(),foo2()] //1,3,4,[3,4],2

var c = yield {a:foo1(),b:foo2()} //1,3,4,{a:3,b:4},2

// var c = yield foo5(5) //1,5,4,[1],2

console.log(c)

console.log(2)

})

function \*foo1(){

console.log(3)

return 3

}

function \*foo2(){

console.log(4)

return 4

}

var foo = co(new Promise(function(reslove,reject){

console.log(1)

reslove(new Promise(function(reslove,reject){

reslove(2)

}))

}).then(res=>{

console.log(res)

}))

1

2

var foo = co(new Promise(function(reslove,reject){

console.log(1)

reslove(new Promise(function(reslove,reject){

reslove(2)

}))

// return 2

}))

//co() 直接reslove()

foo.then(res=>{

console.log(res)

})

1

2

类似

var foo = new Promise(function(reslove,reject){

reslove(new Promise(function(reslove,reject){

console.log(1)

reslove(new Promise(function(reslove,reject){

// console.log(2)

reslove(2)

}))

// return 2

}))

})

foo.then(res=>{

console.log(res)

})

1

2