ES6

Promise

Promise承诺。这个承诺一旦从等待状态变为其他状态就永远不能再更改状态了。

等待中（pending） 完成了（resolved） 拒绝了（rejected）.

当我们在构造Promise的时候，构造函数内部的代码是立即执行的。

New Promise(( resolve,reject)=>{

Console.log(‘new promise’)

Resolve(‘success’)

})

Console.log(‘finish’) //先打印new Promise,在打印finish)

Promise实现了链式调用，也就是说每次调用then之后返回的都是一个Promise,并且是。

正因为

Generator生成器

Function \*foo(x){

Let y=2\*(yield(x+1))

Let z=yield(y/3)

Return (x+y+z)

}

Let it=foo(5)

Console.log(it.next()) //=> {value:6,done:false}

Console.log(it.next(12)) //=>{value:8,done:false}

Console.log(it.next(13)) //=>{value:42,done:true}

首先Generator函数调用和普通函数不同，它返回一个迭代器。

当执行第一次next时，传参会被忽略，并且函数暂停在yield(x+1)处，所以返回5+1=6

当执行第二次next时，传入的参数等于上一个yield的返回值，如果不传参，yield返回undefined,此时let y=2\*12,此时第二个yield等于2\*12/3=8

当执行第三次next时，传入的参数会传递给z,所以z=13,x=5,y=24,所以相加等于42.