promise 知识

迭代器是什么

遍历和迭代有什么区别?

迭代就是从目标源依次按逐个抽取的方式来提取数据，其中目标源满足：1、有序的 2、连续的而遍历就没有这些要求，对于不同数据类型会有不同遍历方式。

迭代引入

可以通过 for..of 来迭代出来

Array、Map、Set、String、TypeArray、arguments、NodeList

```
let arr = [1, 2, 3, 4]

function makeIterator(arr){
  let index=0;
  return{
    next(){
      if(index<arr.length){
        return{
          value: arr[index++], done: false
        }
      }
      return {value: undefined, done: true};
    }
  }
}
let iter = makeIterator(arr);
console.log(iter.next()); // {value: 1, done: false}
console.log(iter.next()); // {value: 2, done: false}
console.log(iter.next()); // {value: 3, done: false}
console.log(iter.next()); // {value: 4, done: false}
console.log(iter.next()); // {value: undefined, done: true}
```

自己动手实现 Symbol.iterator

```
var obj = {
  a: 1,
  b: 2,
  c: 3,
  [Symbol.iterator](){
    let index = 0;
    let map = new Map([['a',1],['b',2],['c',3]]);
    return{
      next(){
        // 获取键值对
        let mapEntries = [...map.entries()];
        if(index < map.size){
          return{
            value: mapEntries[index++], done: false
          }
        }
        return {value: undefined, done: true};
      }
    }
  }
};
let iter = obj[Symbol.iterator]();
console.log(iter.next()); // {value: (2) ["a", 1], done: false}
console.log(iter.next()); // {value: (2) ["b", 2], done: false}
console.log(iter.next()); // {value: (2) ["c", 3], done: false}
console.log(iter.next()); // {value: undefined, done: true}
```

实现对象的自定义迭代器对象

生成器的使用

生成器初识

探讨生成器

探究生成器传参

```
function* test(){
  yield 1;
  yield 2;
  yield 3;
  yield 4;
}

let iter = test();
console.log(iter.next()); // {value: 1, done: false}
console.log(iter.next()); // {value: 2, done: false}
console.log(iter.next()); // {value: 3, done: false}
console.log(iter.next()); // {value: 4, done: false}
console.log(iter.next()); // {value: undefined, done: true}
```

Promise 自测面试题（由易到难 | 满分100分）

async | await

async await和generator的写法很像，就是将 Generator 函数的星号（*）替换成 async，将 yield 替换成await

但async 函数对 Generator 函数做了改进：

1、内置执行器：Generator函数的执行必须靠执行器，所以才有了 co 函数库，而 async 函数自带执行器.也就是说，async 函数的执行，与普通函数一模一样。

2、更好的语义：async 和 await，比起星号和 yield，语义更清楚了。async 表示函数里有异步操作，await 表示紧跟在后面的表达式需要等待结果。

3、更广的适用性： co 函数库约定，yield 命令后面只能是 Thunk 函数或 Promise 对象，而 async 函数的 await 命令后面，可以跟 Promise 对象和原始类型的值（数值、字符串和布尔值，但这时等同于同步操作）

Promise

Promise的特性

1、promise 状态不受外界影响

2、promise 的固化（一旦 promise 状态发生变化后就不能再更改）

thenable 对象

promise 的链式调用

promise 的固化 | 多层嵌套

promise 固化即一旦状态发生变化后就不能再更改

promise 状态依赖

当 promise 状态存在依赖时，它的状态与自身无关了，由依赖来决定

Promise.all | Promise.race