for await of:

```
function gen(time){
    return new Promise((resolve,reject)=>{
        setTimeout(() => {
            resolve(time)
        }, time);
    })
}
async function test(){
    let arr = [gen(2000),gen(200),gen(1000)]
    for await (let i of arr){
        console.log(Date.now(),i);
    }
}
test()
```

遍历异步方法的值，它会等待所有异步方法执行完成之后才去执行，i为promise对象的resolve值

自定义数据结构 异步遍历：

```
const obj = {
    count:0,
    gen(time){
        return new Promise((resolve,reject)=>{
            setTimeout(() => {
                resolve({done:false,value:time})
            }, time);
        })
    },
    [Symbol.asyncIterator](){
        let that = this;
        return {
            next(){
                that.count++
```

```
                if(that.count<4){
                    return that.gen(Math.random()*1000)
                }else{
                    return Promise.resolve({
                        done:true,
                        value:''
                    })
                }
            }
        }
    }
}
async function test(){
    for await (let i of obj){
        console.log(i);
    }
}
test()
```