for await of ……

ES9

for await of

我们知道 for…of 是同步运行的，有时候一些任务集合是异步的，那这种遍历怎么办呢?

```
function Gen (time) {
  return new Promise(function (resolve, reject) {
    setTimeout(function () {
      resolve(time)
    }, time)
  })
}

async function test () {
  let arr = [Gen(2000), Gen(100), Gen(3000)]
  for (let item of arr) {
    console.log(Date.now(), item.then(console.log))
  }
}

test()
// 1560090138232 Promise {<pending>}
// 1560090138234 Promise {<pending>}
// 1560090138235 Promise {<pending>}
// 100
// 2000
// 3000
```

这里写了几个小任务，分别是 2000ms 、100ms、3000ms后任务结束。在上述遍历的过程中可以看到三个任务是同步启动的，然后输出上也不是任务的执行顺序输出的，这显然不太符合我们的要求。

聪明的同学一定能想起来 await 的作用，它可以中断程序的执行直到这个 Promise 对象的状态发生改变，我们修改上面的代码：

```
function Gen (time) {
  return new Promise(function (resolve, reject) {
    setTimeout(function () {
      resolve(time)
    }, time)
  })
}

async function test () {
  let arr = [Gen(2000), Gen(100), Gen(3000)]
  for (let item of arr) {
    console.log(Date.now(), await item.then(console.log))
  }
}

test()
// 2000
// 1560091834772 undefined
// 100
// 1560091836774 undefined
// 3000
// 1560091836775 undefined
```

从返回值看确实是按照任务的先后顺序进行的，其中原理也有说明是利用了 await 中断程序的功能。不过，在 ES9 中也可以用 for…await…of 的来操作：

```
function Gen (time) {
  return new Promise(function (resolve, reject) {
    setTimeout(function () {
      resolve(time)
```