对 11-13 和 11-14 的小思考和 更精简更好理解的实现

11-13 和 11-14 采用 setTimeout 这种实现要比常规写法更精简，更好理解，已经很好了，但如果 第 S100-S103 代码区中 的 setTimeout 的业务耗时超过了 我们在 processManyAsyncAndSync 中的 setTimout 设定的时间 ，会导致 processManyAsyncAndSync 的 setTimeout 会先执行，当然我们可加长 setTimout 时间也可以解决这个问题。但还有比这更好更精简的实现：

就是采用 async...await 来实现，这时可以去掉 processManyAsyncAndSync 中的 setTimeout

```
// test.ts 代码
promise.then((resolveData1) => {
  console.log("第一个then成功了:", resolveData1);
  return new Promise2((resolve, reject) => {
    // resolve("第二个异步操作...")
    // 模拟宏任务异步执行过程
    setTimeout(() => {// S100
      resolve("第二个异步操作...")
    }, 2000);
  })//S103
}, (rejectData1) => {
  console.log("第一个then失败了:", rejectData1);
  return "fail1"
}).then((resolveData2) => {
  console.log("第二个then成功了:", resolveData2);
  return "ok2"
}, (rejectData2) => {
  console.log("第二个then失败了:", rejectData2);
  return "fail2"
}).then((resolveData3) => {
  console.log("第三个then成功了:", resolveData3);
  return "ok2"
}, (rejectData3) => {
  console.log("第三个then失败了:", rejectData3);
  return "fail2"
})
```

采用 async...await 来实现，这时可以去掉 processManyAsyncAndSync 中的 setTimeout

比 11-16 要讲的常规写法简洁的多，而且好理解。

```
  processManyAsyncAndSync(resolveInthen: ResolveType, rejectInThen: RejectType,
    resolve: ResolveType, reject: RejectType) {
    let result: any
    this.resolve_then_callbacks.push(async () => {
      let result = await resolveInthen(this.resolve_executor_value)
      if (isPromise2(result)) {// 是异步的Promise2对象

        // setTimeout(() => {// 可以去掉了
        resolve(result.resolve_executor_value)
        //}, 5)
      } else {
        resolve(result)// 如果是普通的数据,不是异步的Promise2对象
      }
    })
    this.reject_then_callbacks.push(() => {
      result = rejectInThen(this.reject_executor_value)
      reject(result)
    })
  }
}
```