async 和 await 的顺序问题

```
async function async1() {
    console.log("async1 start");  // 2
    await async2();  // 立即执行 async2 函数

    // await 后面的代码会被当作微任务放入 microTask 等待同步代码执行完再执行
    console.log("async1 end");  // 6
}

async function async2 () {
    console.log("async2");  // 3
}

console.log("script start");  // 1,  先执行同步代码

setTimeout(function () {  // 宏任务
    console.log("setTimeout");  // 8 最后执行宏任务
}, 0)

async1(); // 调用一个微任务函数，立即执行里面的内容

// 实例化一个 Promise，会立即执行里面的内容
new Promise(function (resolve) {
    console.log("promise1");  // 4
    resolve();
}).then (function () {  // 微任务
    console.log("promise2");  // 7
})

console.log("script end");  // 5
// 同步代码执行完，就按照放入 microTask queue 的顺序来执行微任务


// 猜测，未验证
// script start
// async1 start
// async2
// async1 end
// promise1
// promise2
// script end
// setTimeout

// 经过验证，下面为正确答案
// script start
// async1 start
// async2
// promise1
// script end
```

```
// async1 end
// promise2
// setTimeout
```

上面代码分析

- 先执行同步代码。
- 再执行微任务代码，微任务代码也分先后顺序，先放入 microTask queue 的先执行。
- 在尝试 DOM 渲染，这里不涉及 DOM 渲染。
- 最后执行宏任务。

```
// 场景2
(async function () {
    console.log("start");
    const a = await 100;
    console.log("a", a);
    const b = await Promise.resolve(200);
    console.log("b", b);
    const c = await Promise.reject(300);
    console.log("c", c);
    console.log("end");
})()

// start
// a 100
// b 200
// 接着就报错了
```

上面代码，在 `await Promise.reject(300)` 这一步中，await 相当于 promise 的 then，而 这里返回的是一个 reject 状态的 promise，用 then 来接收会报错，应该用 catch 接收。所以会报错。

```
Promise.reject(300).then(() => console.log("a"))  // 报错
Promise.reject(300).catch(() => console.log("a"))  // a
```