async-await 和 promise 的区别与联系?

- async-await 是 promise 的语法糖，他们并不互斥。
- async 函数执行后，返回的是一个 promise。
- await 必须使用一个 async 包裹。
- await 相当于 promise 的 then。
- try-catch 相当于 promise 的 catch。

promise 只有三种状态，pending, fulfilled, rejected。
从 pending 变成 fulfilled （已成功）或者从 pending 变成 rejected（已失败）。
resolved 表示已完结，包含了 fulfilled 和 rejected。

```
// async 函数返回的是一个 promise
async function fn1() {
    return 100;
}
console.log(fn1());  // Promise {<fulfilled>: 100}
```

上面代码中，fn1 函数返回一个数值，和 promise 无关，但执行 fn1 函数时，返回的依然是 promise。

async 函数中，如果没有返回，则会使用 Promise.resolve(xx) 或 Promise.reject(xx) 来包裹。

```
// await 相当于 promise 的 then
!(async function fn1() {
    const p1 = await Promise.resolve(100);
    console.log(p1);
})()
// 100
```

上面代码说明，await 后面的 Promise 会被 then 执行一次，返回 then 执行后的结果。

```
// await 相当于 promise 的 then
!(async function fn1() {
    const p1 = await 200;
    console.log(p1);
})()
// 200
```

上面代码，即使 await 后面跟的不是一个 promise，也会转换成一个 promise，并执行 then 方法。等价于：

```
// await 相当于 promise 的 then
!(async function fn1() {
    const p1 = await Promise.resolve(200);
```

```
    console.log(p1);
})()
// 200
```

try-catch 相当于 promise 的 catch。

```
// try-catch 相当于 promise 的 catch
!(async function fn1() {
    const p1 = await Promise.reject('error');
    console.log("p1", p1);
})()
```

上面代码会报错，因为 await 相当于 promise 执行 then 方法。上面构造一个 rejected 状态的 promise ，不会执行 then 方法，所以就不会执行下面打印的这一步。

可以使用 try-catch 来解决。

```
!(async function fn1() {
    try {
        const p1 = await Promise.reject('error');
        console.log("p1", p1);
    } catch(err) {
        console.log("p1--err", err);
    }
})()
```

上面代码，打印 p1 这一步不会执行，catch 中将会捕获 try 块儿的异常（错误）。

await 那一行代码执行后，后面的代码不会马上执行，会被当作异步代码（微任务）放入 callback queue 中等待同步代码执行完再通过 event loop 机制放入 call stack 执行。