我们在使用async await时如果要处理错误，如果有多个异步操作，需要每一次书写 try...catch。这样代码的简洁性较差，且业务代码需要包含在try...catch中。没办法把业务错误和代码错误分开；

为了让代码更整洁，不出现用法混乱的情况，我们抽了一个ToAsyncAwait方法；以下是方法代码跟使用示例

```
// 源码------------
/**
 * @param {*} promise 异步方法主体
 * @param {*} fromatResult 是否处理成统一格式，不处理则直接返回第一个参数。 true
 * @return {error,resutl}  有错误 resutl为null,error为错误体。没错误 error为nu
 */
const toAsyncAwait = (promise, fromatResult = true) => {
    if (!fromatResult) {
        return promise;
    } else {
        return promise.then((res) => ({ error: null, result: res })).catch((
    }
}


 //使用示例--------------
const fn = async () => {
    // 业务1 使用error判断
    const { error, result } = await toAsyncAwait(Axios.get('xxx1'));
    if (error) {
        alert(error);
    } else {
        console.log(result);
    }
    // 业务2 使用result判断  使用这种判断的时候注意你的返回结果是不是有为null的时候
    const { error: er, result: res } = await toAsyncAwait(Axios.get('xxx1'));
    if (res) {
        console.log(res);
```