async 关键字返回一个promise对象 所以在其中需要await关键字来进行等待操作

```
async function testAsync() {
    return "hello async";
}

const result = testAsync();
console.log(result);
```

看到输出就恍然大悟了——输出的是一个 Promise 对象。

```
c:\var\test> node --harmony_async_await .
Promise { 'hello async' }
```

普通promise的写法

```
 * 传入参数 n，表示这个函数执行的时间（毫秒）
 * 执行的结果是 n + 200，这个值将用于下一步骤
 */
function takeLongTime(n) {
    return new Promise(resolve => {
        setTimeout(() => resolve(n + 200), n);
    });
}

function step1(n) {
    console.log(`step1 with ${n}`);
    return takeLongTime(n);
}

function step2(n) {
    console.log(`step2 with ${n}`);
    return takeLongTime(n);
}

function step3(n) {
    console.log(`step3 with ${n}`);
    return takeLongTime(n);
}
```

现在用 Promise 方式来实现这三个步骤的处理

```
function doIt() {
    console.time("doIt");
    const time1 = 300;
    step1(time1)
        .then(time2 => step2(time2))
        .then(time3 => step3(time3))
        .then(result => {
            console.log(`result is ${result}`);
            console.timeEnd("doIt");
        });
}

doIt();

// c:\var\test>node --harmony_async_await .
// step1 with 300
// step2 with 500
// step3 with 700
// result is 900
// doIt: 1507.251ms
```

使用async await关键字的写法

```
async function doIt() {
    console.time("doIt");
    const time1 = 300;
    const time2 = await step1(time1);
    const time3 = await step2(time2);
    const result = await step3(time3);
    console.log(`result is ${result}`);
    console.timeEnd("doIt");
}

doIt();
```

看起来会更加的清楚 代码更加简洁
使用场景多用于多个函数同时依赖于上一个函数的执行返回结果的场景 （在接受多个参数的场景下更实用！！！）
eg：

```
function step1(n) {
    console.log(`step1 with ${n}`);
    return takeLongTime(n);
}

function step2(m, n) {
    console.log(`step2 with ${m} and ${n}`);
    return takeLongTime(m + n);
}

function step3(k, m, n) {
    console.log(`step3 with ${k}, ${m} and ${n}`);
    return takeLongTime(k + m + n);
}
```

这回先用 async/await 来写：

```
async function doIt() {
    console.time("doIt");
    const time1 = 300;
    const time2 = await step1(time1);
    const time3 = await step2(time1, time2);
    const result = await step3(time1, time2, time3);
    console.log(`result is ${result}`);
    console.timeEnd("doIt");
}

doIt();

// c:\var\test>node --harmony_async_await .
// step1 with 300
// step2 with 800 = 300 + 500
// step3 with 1800 = 300 + 500 + 1000
// result is 2000
// doIt: 2907.387ms
```

```
function doIt() {
    console.time("doIt");
    const time1 = 300;
    step1(time1)
        .then(time2 => {
            return step2(time1, time2)
                .then(time3 => [time1, time2, time3]);
        })
        .then(times => {
            const [time1, time2, time3] = times;
            return step3(time1, time2, time3);
        })
        .then(result => {
            console.log(`result is ${result}`);
            console.timeEnd("doIt");
        });
}

doIt();
```