## 1、正常情况下，async中的await命令是一个Promise对象，返回该对象的结果。但如果不是Promise对象的话，就会直接返回对应的值，相当于Promise.resolve()

async function fn () {

// return await 1234

// 等同于

return 123

}

fn().then(res => console.log(res))

结果：123

## 2、在async1中await后面的Promise是没有返回值的，也就是它的状态始终是pending状态，因此相当于一直在await，await，await却始终没有响应...

async function async1 () {

console.log('async1 start');

await new Promise(resolve => {

console.log('promise1')

})

console.log('async1 success');

return 'async1 end'

}

console.log('srcipt start')

async1().then(res => console.log(res))

console.log('srcipt end')

结果：

'script start'

'async1 start'

'promise1'

'script end'

## 3、

async function testAsync() {

console.log("执行testAsync");

return Promise.resolve("hello async");// 相当于 return “hello async”

}

## 错题集：

### 5.6 题目六

async function async1 () {

console.log('async1 start');

await new Promise(resolve => {

console.log('promise1')

resolve('promise1 resolve')

}).then(res => console.log(res))

console.log('async1 success');

return 'async1 end'

}

console.log('srcipt start')

async1().then(res => console.log(res))

console.log('srcipt end')

现在Promise有了返回值了，因此await后面的内容将会被执行：

'script start'

'async1 start'

'promise1'

'script end'

'promise1 resolve'

'async1 success'

'async1 end'

### 5.7 题目七

async function async1 () {

console.log('async1 start');

await new Promise(resolve => {

console.log('promise1')

resolve('promise resolve')

})

console.log('async1 success');

return 'async1 end'

}

console.log('srcipt start')

async1().then(res => {

console.log(res)

})

new Promise(resolve => {

console.log('promise2')

setTimeout(() => {

console.log('timer')

})

})

这道题应该也不难，不过有一点需要注意的，在async1中的new Promise它的resovle的值和async1().then()里的值是没有关系的，很多小伙伴可能看到resovle('promise resolve')就会误以为是async1().then()中的返回值。

因此这里的执行结果为：

'script start'

'async1 start'

'promise1'

'promise2'

'async1 success'

'async1 end'

'timer'

### 6.1 题目一

在async中，如果 await后面的内容是一个异常或者错误的话，会怎样呢？

async function async1 () {

await async2();

console.log('async1');

return 'async1 success'

}

async function async2 () {

return new Promise((resolve, reject) => {

console.log('async2')

reject('error')

})

}

async1().then(res => console.log(res))

例如这道题中，await后面跟着的是一个状态为rejected的promise。

#### 如果在async函数中抛出了错误，则终止错误结果，不会继续向下执行。

所以答案为：

'async2'

Uncaught (in promise) error

#### 如果改为throw new Error也是一样的：

async function async1 () {

console.log('async1');

throw new Error('error!!!')

return 'async1 success'

}

async1().then(res => console.log(res))

结果为：

'async1'

Uncaught (in promise) Error: error!!!

### 6.2 题目二

#### 如果想要使得错误的地方不影响async函数后续的执行的话，可以使用try catch

async function async1 () {

try {

await Promise.reject('error!!!')

} catch(e) {

console.log(e)

}

console.log('async1');

return Promise.resolve('async1 success')

}

async1().then(res => console.log(res))

console.log('script start')

这里的结果为：

'script start'

'error!!!'

'async1'

'async1 success'

#### 或者你可以直接在Promise.reject后面跟着一个catch()方法：

async function async1 () {

// try {

// await Promise.reject('error!!!')

// } catch(e) {

// console.log(e)

// }

await Promise.reject('error!!!')

.catch(e => console.log(e))

console.log('async1');

return Promise.resolve('async1 success')

}

async1().then(res => console.log(res))

console.log('script start')

运行结果是一样的。