这节课我们学习如何将一段异步代码包装成同步代码

**异步代码**

function GetMess(){

GM\_xmlhttpRequest({

url:"https://bbs.tampermonkey.net.cn/",

method :"GET",

headers: {

},

onload:function(xhr){

console.log(xhr.responseText);

}

});

}

GetMess()

copy

这是一段十分简单的代码，在函数内使用了一个get获取网页内容，如果我们想在获取内容之后进行一些处理需要写在onload函数内，这种代码就叫做异步代码，当执行完请求后再回调处理的函数，有时候这样非常麻烦，那有没有什么办法改为同步代码呢？解决方案就是使用Promise

let p=new Promise((resolve, reject) => {

})

copy

我们首先创建一个Promise，当Promise不调用resolve和reject结束状态，那将一直是pengding状态，只有调用resolve或reject的时候，才可以结束状态并返回resolve或reject函数内的值

function GetMess(){

let p=new Promise((resolve, reject) => {

GM\_xmlhttpRequest({

url:"https://bbs.tampermonkey.net.cn/",

method :"GET",

headers: {

},

onload:function(xhr){

resolve(xhr.responseText)

}

});

})

}

GetMess()

copy

现在我们要求在onload后使用resolve结束这个Promise状态，并返回xhr.responseText的值

我们再结合async和await的语法糖就可以了

await需要一个Promise，所以我们需要将我们构建的这个Promise返回

function GetMess(){

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

GM\_xmlhttpRequest({

url:"https://bbs.tampermonkey.net.cn/",

method :"GET",

headers: {

},

onload:function(xhr){

resolve(xhr.responseText)

}

});

})

}

async function GetPromiseAndWait(){

let text=await GetMess()

}

GetPromiseAndWait()

copy

添加await GetMess()后，GetMess函数会返回一个Promise，Async和Await语法糖会使Promise阻塞直至完成状态后，将resolve内填入的值赋值给text变量上。

需要注意的是我们不可以在全局作用域内使用await，目前并未实现该特性，只有在async的函数内才可以使用await。

由于GetPromiseAndWait使用了Async语法糖，所以调用他现在也会返回一个Promise变量，如果我们想要同步获取他返回的值，也需要使用Async与Await

function GetMess(){

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

GM\_xmlhttpRequest({

url:"https://bbs.tampermonkey.net.cn/",

method :"GET",

headers: {

},

onload:function(xhr){

resolve(xhr.responseText)

}

});

})

}

async function GetPromiseAndWait(){

let text=await GetMess()

return '我处理完了'

}

function WantGetFinallyText(){

let data=GetPromiseAndWait()

console.log('WantGetFinallyText',data)

}

WantGetFinallyText()

copy

图片.png

我们可以看到console.log输出了一个promise，这个值是pengding，因为我们调用了GetPromiseAndWait，他在没有执行到return的时候是一个执行中的状态。

如果我们想要获取到GetPromiseAndWait的值，需要让这个函数执行到return，也就是fulfilled状态。这个时候我们需要对调用这个函数返回的Promise再进行一个Async和Await，来让这个Promise形成阻塞

function GetMess(){

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

GM\_xmlhttpRequest({

url:"https://bbs.tampermonkey.net.cn/",

method :"GET",

headers: {

},

onload:function(xhr){

resolve(xhr.responseText)

}

});

})

}

async function GetPromiseAndWait(){

let text=await GetMess()

return '我处理完了'

}

async function WantGetFinallyText(){

let data=await GetPromiseAndWait()

console.log('WantGetFinallyText',data)

}

WantGetFinallyText()

copy

图片.png

这个时候await就会阻塞住GetPromiseAndWait一直到履行状态再将其值赋值给data，因为return会使函数变为履行状态，也就是说await会使函数一直阻塞到return返回内容为止。

注意，还有一个知识点

就是GetPromiseAndWait如果return一个Promise，那这个函数返回的将是这个Promise，await阻塞的也是这个Promise的值。

这个概念我们之前也讲过，当async函数return一个值时，会将其包装成一个Promise，当返回一个Promise时，将直接返回这个Promise

那么到这里，你就学会了Promise以及多层Async函数的同步！

**结语**