**promise对象的理解引申**

**（一）同步异步-线程进程**

1. **同步异步的理解**

概念：用于形容方法的调用

**同步**：一旦开始要等到方法调用返回后，才能继续后续的行为（等待）

**异步：**一旦开始方法调用就会立即返回，调用者可以继续执行后续的操作，**提高执行效率，互不干扰**（无需等待）

**JS单线程 -> 任务队列 -> 异步任务**

1. **同步异步适用场景**

**异步：**所有需要等待的场景，但等待过程中又不能阻塞程序

浏览器默认认为下面都是耗时程序，会放到任务队列中，等待执行不耗时程序执行结束后，再依次执行该任务队列中。

1. 定时任务：setTimeout(一次执行) setInverval(反复执行)
2. Ajax网络请求；img图片加载
3. 事件绑定（不知道事件用户何时执行需要等待）

**同步：**ajax请求比如应用程序往下执行时以来从服务器请求的数据，那么必须等这个数据返回才行。如权限问题

1. **同步异步优缺点**

**同步：**可以避免死锁（两个或以上进程再执行过程中，因同一个资源相互等待，无外力作用下都无法推进下去，这种系统状态）和读脏数据（一个任务把数据修改后还没有存入数据库，另外一个任务使用到了这个数据，这就是一个脏数据，是无效的），可以理解为一个共享的资源，每个人都有权限修改时，会出现多个人同时进行操作的问题，就容易出错，而同步就会按执行顺序修改，但会**阻塞任务执行**

**异步：**提高效率，避免阻塞，缺点就是要处理好高并发问题。异步编程造成回调地狱，多层嵌套，(promise对象解决)

1. **进程和线程**

**进程：**程序一旦被载入到系统中执行，这就是一段一个进程，一个或者多个进程组成系统中的运行的应用程序，如某浏览器。系统为每一个进行分配独立的CPU、内存等资源。**是系统分配资源的最小单位。**

**线程：**进程执行过程中的每一个任务就是线程，各个线程共享进程拥有的内存资源。**是进程执行的最小单位**

**进程与线程的关系：**（1）一个程序至少有一个进程，一个进程至少有一个线程 （2）线程不能脱离进程 （3）当一个进程中只有一个线程时，被称为单线程 （4）单线程或者多线程是指同一个进程中的单或者多

1. **异步和多线程的关系**

不是同等的关系，异步是最终目的，而多线程是实现一步的一种手段，实现异步可以采用多线程的技术或者交给另外的进程处理。

**（二）Promise对象**

1. **Promise对象**

ES6中提出的，异步编程的解决方案，避免多层嵌套，简单说Promise就是一个容器，里面保存着某个异步操作的结果

1. Promise对象的特点：代表一个异步操作，有三种状态pending（进行中）、fulfilled（已成功）、rejected（已失败）,只有异步操作的结果可以决定是那种状态，一旦改变就会再变
2. 缺点：一旦开始无法取消；如果内部不设置回调函数，Promise内部抛出的错误不会反映到外部；在pending状态是无法确定发展到那个阶段。
3. 用法：两个参数 resolve reject

Var p = new Promise( function (resolve,reject) {

$.ajax({

.....

success:function(res){

Resolve(res)

},

error:function(err){

Reject(err)

}

})

})

.then((res)=>{

//成功

}).catch((err)=>{

//失败

})

1. **async await**

ES7中的技术，简化promise的操作，提高 promise代码的阅读性和理解性。async、await可以使异步调用不返回promise对象，直接把then回掉函数的第一个形参的结果返回

async function xxx () {

let result = await yyy()

}

Eg:

function xx(){

return new Promise( function (resolve,reject) {

$.ajax({

.....

success:function(res){

Resolve(res)

},

error:function(err){

Reject(err)

}

})

})

}

async function yy(){

let result = await xx()

}

yy()

**注意：**

1. async需要设置到promise对象外面最近的function前面。
2. Await必须结合async一起使用，可一对多，一个async对多个await，各个await顺序执行。