1. 在ES5中，如何实现多个形参不确定，还要求参数值与形参变量一一对应?

答: function order(menu){

//menu: { zs:xxx, tang:xxx, td:xxx, rc:xxx, sc:xxx}

menu.zs...

menu.tang...

}

order({ zs:xxx, tang:xxx, td:xxx, rc:xxx, sc:xxx})

2. function Fun(){

var i=0;

return function(){

console.log(i++);

}

}

var f1=Fun();

var f2=Fun();  
 f1(); //0

f1(); //1

f2(); //0 不是2?

f2(); //1 不是 3?

答: 妈妈分两次分别生了两个孩子，每次生孩子都给孩子包一个专属的红包。所以，f2和f1用的不是一个红包。他们分别有自己专属的红包。

3. .then()中的函数为什么不加()也能执行？谁调用的？

答：liang().then(ran)，其实还是回调函数的方式，先把ran放入.then中留着备用。只有当前一个liang中调用了door()打开门后，才自动找到.then()中提前放好的ran，此时才自动加()调用。

4. liang().then(ran).then(dong).then(liang)为什么不会形成死循环?

答: 每次调用一个函数，都是创建并返回一个新的格子间。新的格子间，如果后边没有接then,就不会有下一步后续任务。

这里liang() ... .then(liang)等于先后调用了两次liang，返回过两个格子间。而第二次调用liang返回的格子间绝对不等于第一个格子间，且第二个格子间后边没有.then。所以，第二次调用.then(liang)后，就没有后续步骤了。

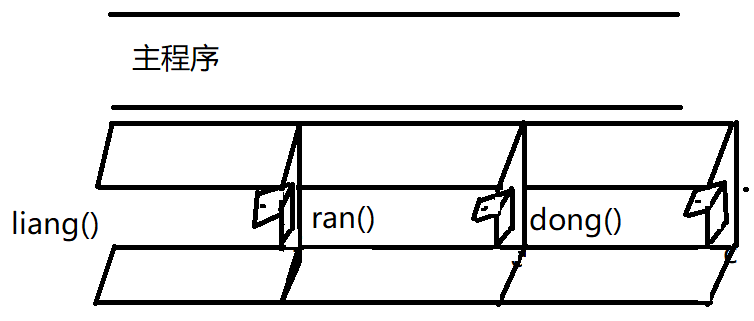
5. ran不加()也能调用，是.then()的功劳吗？

答？不只是then。

所有.then,其实起到的是收集函数并排队的作用。比如: liang().then(ran).then(dong)，等效于: [liang, ran, dong]顺序存储好。然后，依次自动调用每一个函数，如果前一个函数调用了door，才i++，自动调用第二个。如果第二个调用了door，才i++，自动调动第三个。所以ran的执行，是因为被.then()收齐起来，再由上一步调用door()共同触发执行的。

6. console.log(“比赛结束”)，不放在then中，依然会在所有人到达终点前就执行？

答：是的！看图



Promise格子间本身都是异步的。第一个格子间是异步的，在主程序之外执行的，所以，就算后续格子间与第一个格子间是串联的关系，无非也是在主程序之外，串联起一个更长的异步执行过程而已。所以，由promise串起来的多个异步函数，组成的是一个大的异步任务，但与主程序相比，依然是异步的。

7. var bang=await liang(); 可不可以不加await

答: 不加虽然不会报错，但是后续程序不会等待当前函数执行完，就提前执行了。执行顺序是不是颠倒了！

8. return new Promise(

function(resolve, reject){

}

)中，resolve和reject，必须叫这两个名字吗？

答: 只要是形参，永远没有固定的名称

但是，顺序是固定的！

将来只要调用第1个开关，都表示当前函数执行成功，都走.then()

将来只要调用第2个开关，都表示当前函数执行错误，进入.catch()

9. Uncaught (in promise) 说明是人家Promise函数中用reject()抛出了错误！但是你没接住:

答: ES6: .catch(e=>console.log(e))

ES7: try{

...

}catch(e){ console.log(e) }

10. 异步是不是脱离开~~全局~~主程序，没有和主程序在一起? 如果加上await，是不是就等待了呢？

答: 问题一: 异步的确是不和主程序一起执行的

问题二: await不能随便用:

1. 这个异步函数，必须支持Promise  
 2. await外围必须套上(async function(){...})();

11. 到目前为止有哪些函数是异步的:

答: setTimeout() setInterval() 可以自己控制结束的时间

xhr.onreadystatechange=function(responseText){

}

将来:

ajax({url, type, data, dataType}).then(res=>{...})

jquery:

$.ajax({url, type, data, dataType}).then(res=>{...})

vue:

this.axios.get(url,{ params: ... }).then(res=>{ ... })

nodejs:

pool.query(sql,[...],(err,result)=>{//异步

... ...

pool.query(sql,[...],(err,result)=>{//异步

... ...

pool.query(sql,[...],(err,result)=>{//异步

... ...

res.send(result)

})

})

})

12. 如果等待多个异步任务都完成后，才执行后续操作，用ES7怎么写？如果Promise.all中抛出异常，如何处理?

答: var arr=await Promise.all([liang(), ran(), dong() ])

console.log("比赛结束")

其中: arr中: [101,102,103]

如果Promise.all中抛出异常，则依然使用try{ ... }catch(e){ ... }处理:

try{

var arr=await Promise.all([liang(),ran(),dong()])

}catch(e){

//错误处理代码

}

await代替了.then,

try{ ... }catch(e){ ... }代替了.catch()

13. 三种遍历，那种效率最高？

答：一样高！因为其实只有一种遍历，就是for循环。其它都是封装的for循环而已。

所以，ES5的其它遍历函数,indexOf, every, some, forEach, map, filter和reduce，都没有效率的提升，因为内部都是for循环而已。只是语法简单而已。

14. arguments可不可以通过认数组的原型对象当干爹来使用数组家的函数

答: 只要是数字下标的对象，都可以通过继承数组的原型对象来使用数组家的函数。

其实，还可以用.call()抢。

比如:

Array.prototype.reduce.call(arguments,(prev,elem)=>prev+elem,0)

15. reduce()去掉0，也能正常执行？

答: 标准写法都要带第2个参数

只是在极特殊情况下，如果数组中要统计的元素都是数字，则捐款箱的默认值为0——了解。

16. 是不是有了ES6，就不用ES5了？

答: 要看浏览器是否兼容ES6.

如果使用框架和微信开发，基本上可以使用最新的语法。因为框架中和微信开发中都自带翻译工具（Babel），在项目打包时，会将ES6代码翻译成ES5代码，保证浏览器兼容性。

如果没使用框架，使用原生或jquery开发，则尽量不要使用ES6。

17. ...不是打散的意思吗？这里怎么成了收集？

答: 定义函数时，形参列表中的...是收集不确定个数的实参值的意思

调用函数时，实参值列表中的...是打散数组实参为多个实参值，分别传入的意思。

18. ...作为打散数组放在实参列表中时，不一定放在结尾？

答: 是的！可以可其他单独的实参值混搭！

19. function liang(callback){ ... }

liang(function(){ ran() })

liang()调用时，实参可不可以使用函数名？

答: 可以，但是只能接一个函数，下一个函数就接不了了

比如: liang(ran) 此时callback=ran callback()=ran()

但是，因为ran不是立刻调用，不能加()，所以ran无法接下一个dong函数了。

20. liang(

function(){

ran();

}

)为什么必须套一层function()，为什么不能写成liang(ran())

答: liang(ran())，程序先执行ran()，将ran()的返回值，给liang()作为参数。但是，因为function ran(){} 是一件事儿，没有return。意味着ran()，返回undefined。就说明liang()得到的参数是undefined。而liang()要的是一个可执行的函数对象。

liang(ran)，是对的。ran没有加()，说明不是立刻调用执行。而是将ran这个函数对象本身交给liang()作为参数。这符合liang()的要求。但是，这样做的缺陷是，ran不能加()，也就无法接第三个函数。

解决: liang(

function(){

下一步要执行的js语句

ran(); //好处: ran()中可接下一个函数

}

)

21. 都已经百米赛跑了，为什么每个人还需要door()开门呢？

答: 整个比赛结束的标准是每个人都比赛结束了

而其中仅任何一个人开门，都无法代表别人也结束比赛了。所以，只有每个人都打开了自己的门，才说明整个比赛结束了。

22. door(bang)能不能用return代替

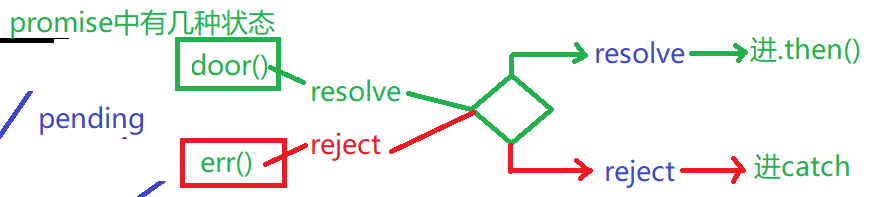
答: 不行！因为door()不仅有返回值的作用，而且，还有修改promise状态的作用。而return只有返回值的作用，无法改变promise对象的状态触发.then或.catch

其实一个promise对象有三个状态:

1. pending: 挂起: 异步任务正在执行，但还没执行完。

2. resolve: 同意: 当异步任务执行完，且执行成功，主动调用door()函数，会将Promise对象的状态，改为resolve。当promise对象的状态被改为resolve后，就会自动执行.then()中的函数

3. reject: 拒绝: 当异步任务执行完，且执行失败了，主动调用err()函数，会将Promise对象的状态，改为reject。当promise对象的状态一旦被改为reject后，就会自动调用catch()中的函数。



23. Promise.all如何处理错误？

如果Promise.all中一个异步函数出错，结果是怎样？

答: Promise.all()也是用.catch()处理函数某一个异步任务错误。如果其中一个异步任务出错，则其它异步任务继续执行完。但是.then()不再执行，而是转到.catch()中执行错误处理代码。因为all中其它异步任务不知道哪个同伴出错了，所以依然继续执行。而.then因为没有收到all中所有异步函数都转为resolve状态的通知，认为不是所有异步任务都成功，就不能继续执行.then了。