准备: 微信小程序: WEB问题速查 小恐龙图标

1. 历届师兄师姐在今天课程中曾经问过的高频的经典的问题清单和解答: 小程序->首页->JSCORE->day08

自己可能没有发现问题，但是想知道别的同学都问过哪些问题

2. 今日课程对应的重点案例的视频以及鄙视面试所需的扩展知识视频：小程序->在线->JSCORE-> day08

如果因为某些原因错过直播讲解，不用着急！直接看小程序中视频的回看即可。

如果tmooc不能用了，立刻看小程序视频，小程序视频比tmooc要全，甚至有扩展的新知识。

如果学有余力，将来有能力拿更高薪资，小程序中所有视频必须都要看。但是不必现在都看完。只要在笔试面试前都看完即可！

问题: 个人版小程序有功能限制：禁止在线播放视频

解决: 长按视频连接地址，完整复制视频连接地址和验证码到浏览器打开。通过百度网盘下载后观看。

3. 今日课程中包含的英语单词的翻译：小程序->单词->JSCORE->day08

英文不太好的同学，可以提前看一下第二天需要用到哪些英文单词。

约定: 2件事:

1. 第三阶段禁止上课跟着敲笔记和敲代码！谁抄笔记抄的勤，谁抄代码抄的勤，谁学的越烂！

2. 每个完整功能的案例/视频，至少写三遍: (如果需要画图，必须亲手画图)

(1). 连代码+注释抄一遍 —— 明白这个例子是要做什么事儿

(2). 保留注释，自己试着看着注释，将代码填回来 —— 复习函数和对象

(3). 删除所有代码和注释，自己试着用自己的话，把注释写回来，再翻译为代码！—— 只有自己可以写出人话的注释，这个例子才算自己会！

如果没有问题的同学: （享受vip一对一答疑的时间还剩13天）

1. 上课时那句话没听懂，笔记中那句话看不懂，案例中那句话看不懂，立刻问！

2. 小程序问题清单中哪个问题或解答看不懂，立刻问！

3. 小程序视频中哪个视频看不懂，立刻问

4. 个人项目中哪个功能不会做，立刻问

5. 平时刷笔试题时，遇到不会的笔试题，立刻问

谁问的问题越多！谁的培训费交的越值！

正课:

1. Promise

一. Promise:

对应小程序视频: 小程序->在线->JSCORE->day08 3. 回调地狱 callback hell 和promise...

1. 什么是: 专门实现多个异步任务顺序执行的技术，专门用来避免回调地狱问题

callback hell

2. 问题: 旧js中如何使用回调函数保证多个异步任务顺序执行

(1). 错误: 单纯顺序调用多个异步函数，无法保证多个异步任务必须顺序执行。

(2). 正确: 可以用回调函数: 2步:

a. 定义前一项任务时:

function 前一项任务(小推车){

...异步任务

异步任务的最后一句话之后

自动调用 小推车()

}

b. 调用前一项任务时:

前一项任务(

function(){

下一项任务的执行语句

}

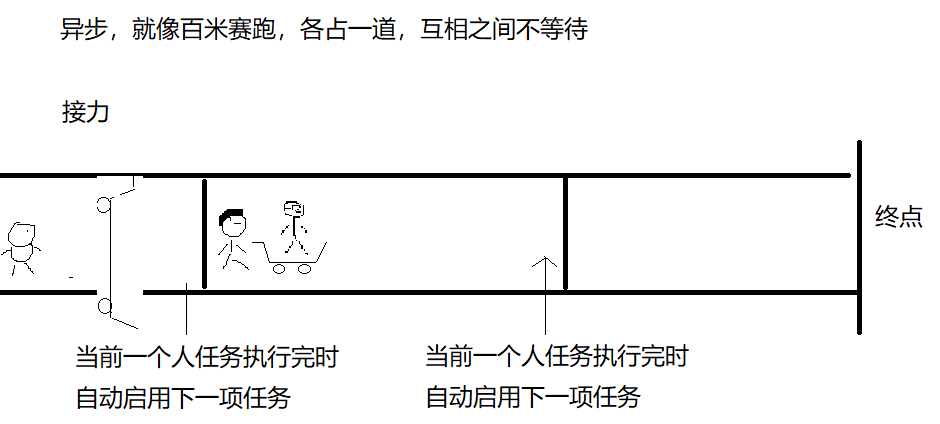
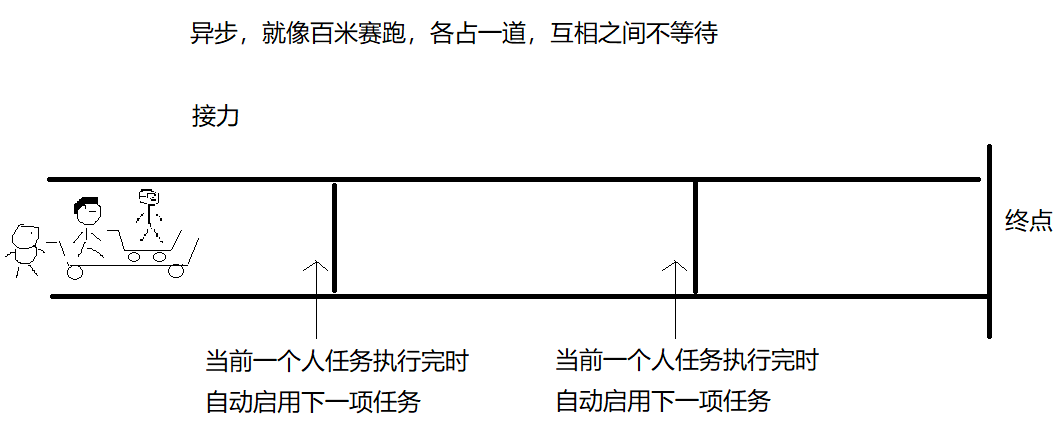
)

c. 结果:

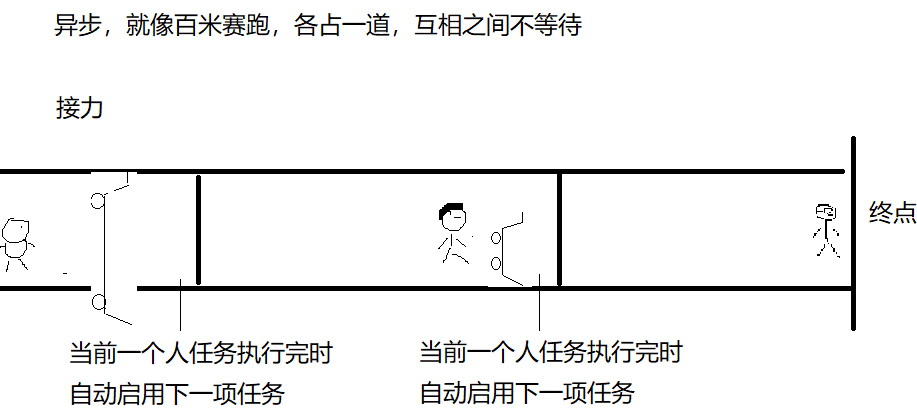
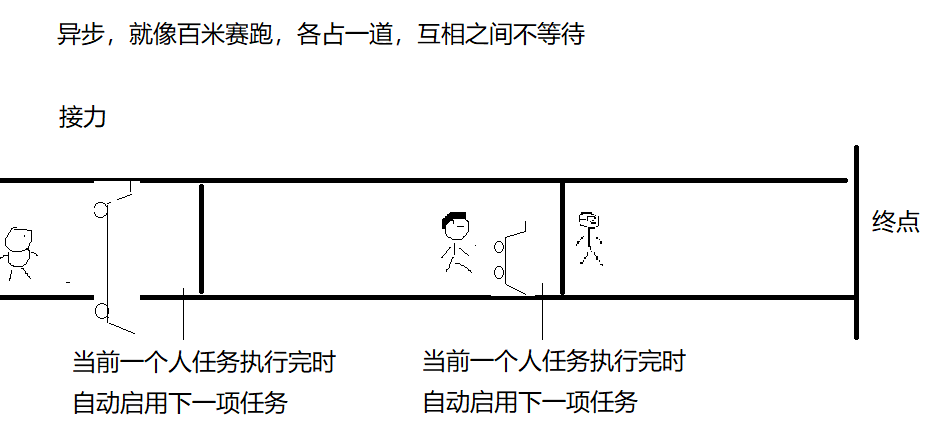
1). 前一项任务的“小推车”在开始执行时，就提前保存住了包含下一项任务的匿名函数

2). 下一项任务只有在前一项任务最后一句话执行完，自动调用“小推车”时，才能开始执行。此时已经是几s之后了！

1 2



3 4

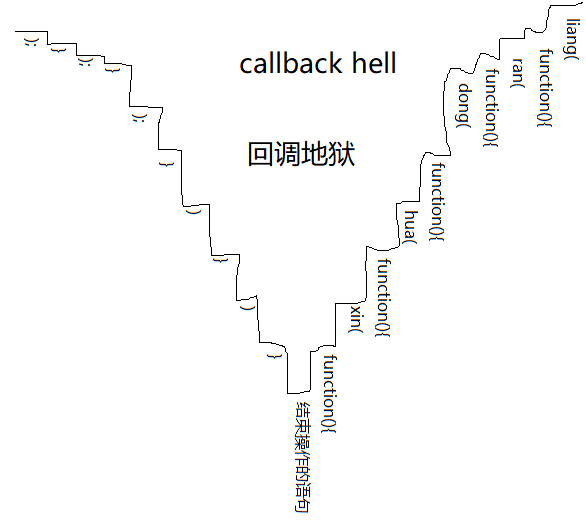
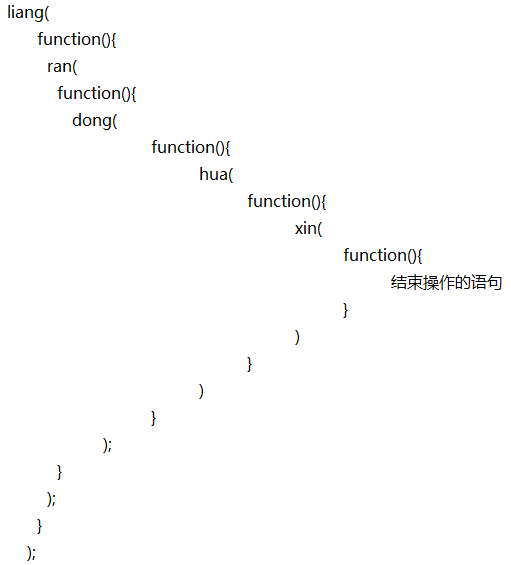


d. 示例: 使用回调函数方式，保证多个异步函数必须顺序执行

1\_callback.html

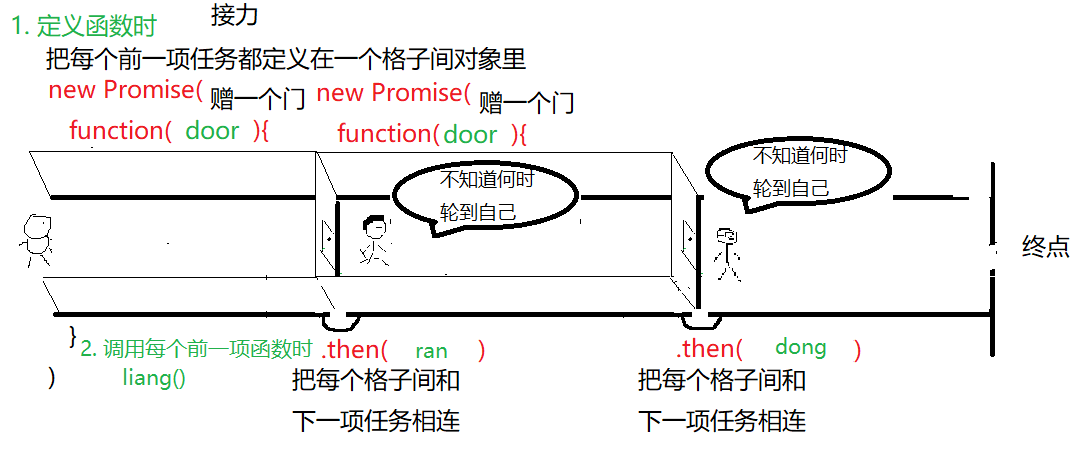
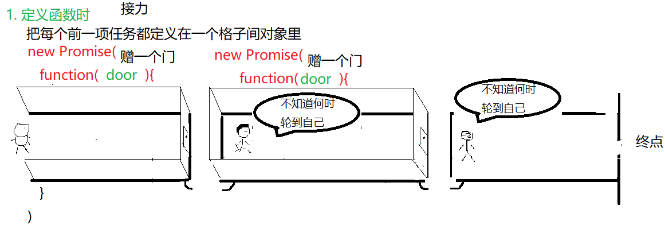
|  |
| --- |
| <script>  //亮可以跑步，但是要提前准备一个小推车，装上下一项要执行的任务  function liang(小推车){  console.log(`亮起跑...`);  setTimeout(function(){ //异步  console.log(`亮到达终点!`);  //当亮的最后一步执行完之后，自动倒出小推车中的东西开始执行  小推车();  },6000)//亮需要6s跑完全程  }  //然可以跑步  function ran(小推车){  console.log(`然起跑...`);  setTimeout(function(){ //异步  console.log(`然到达终点!`);  //当然的最后一步执行完时，然才调用小推车里的东西  小推车();  },4000)//然需要4s跑完全程  }  //东可以跑步  function dong(){  console.log(`东起跑...`);  setTimeout(function(){ //异步  console.log(`东到达终点!`)  },2000)//东需要2s跑完全程  }  //必须让亮、然、东按顺序依次起跑，跑接力  //错误  // liang();//亮起跑...  // ran();//然起跑...  // dong();//东起跑...  //正确:  liang(  //只创建一个新函数，暂不执行！  function(){//用匿名函数包裹下一步要做的事儿  ran(  function(){//只创建一个新匿名函数，包含然的下一项任务，但暂不执行  dong();  }  );//然的小推车=function(){ dong() }  }  );//亮的小推车=function(){ ran(); }  //亮内部，当亮最后一步执行完，自动调用小推车()，匿名函数才开始执行，那已经是6s只有了！  //然内部，当然最后一步执行完，自动调用然的小推车()，最内层匿名函数才开始执行，已经又是4s后  //所以轮到内层函数执行时，已经等了6s+4s=10s;  // 亮 然  </script>  运行结果:  亮起跑...  亮到达终点!  然起跑...  然到达终点!  东起跑...  东到达终点! |

3. 新问题: 有时企业中需要5个异步函数顺序执行！就会产生非常深的嵌套！酷似传说中的18层地狱，所以这个新问题，这种特别难看的深层嵌套的书写格式，被称为回调地狱问题

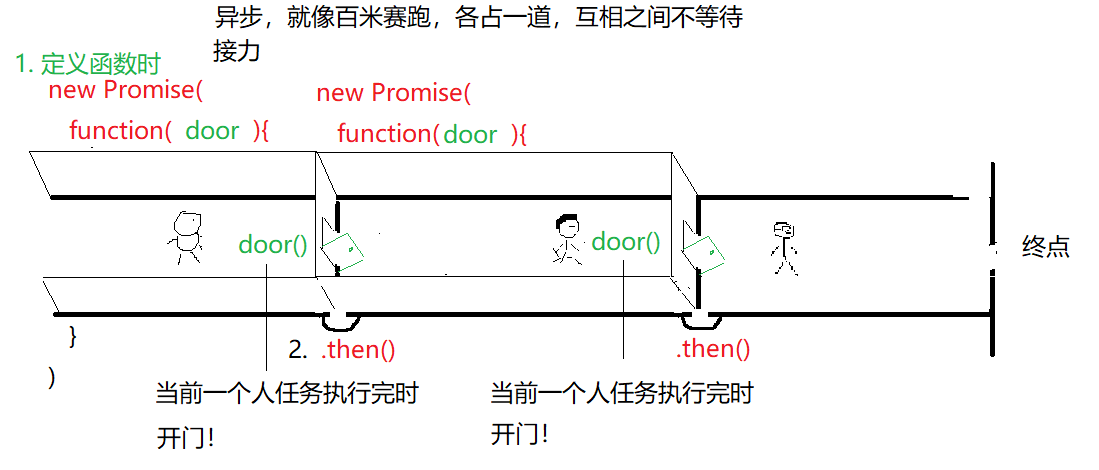
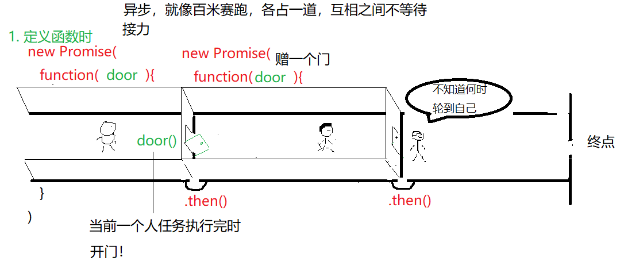


4. 新解决: Promise，2步:

1 2



3 4



(1). 定义前一项任务时:

function 前一项任务(){

//要让外界知道这个函数中有一个格子间

return new Promise( //创建一个格子间对象

//形参永远可以随意改名

function(door){//附赠了一个门door

原异步任务

原异步任务最后一步执行完，一定记得开门:

door() //只有开门，下一项任务才能开始执行

}

)

}

(2). 调用前一项任务时:

//.then()否则连接前后两项任务

前一项任务().then(下一项任务)//千万不要加()，因为不是立刻调用执行！是稍后才执行

(3). 结果: 当调用前一项任务结束后，前一项任务调用door()开门，通知下一项任务。.then()中的下一项任务才开始执行！

5. 优点: 即使有很多任务必须顺序执行，也不会出现深层嵌套的回调地狱

任务1.then(任务2).then(任务3).then(任务4).then(...)...

6. 示例: 使用Promise优化多个异步任务顺序执行

2\_promise.html

|  |
| --- |
| <script>  //亮可以跑步  function liang(){  //让外部知道liang中有一个格子间  return new Promise(  // 赠了一个门  function(door){  //原来的异步任务！该怎么写就怎么写  console.log(`亮起跑...`);  setTimeout(function(){ //异步  console.log(`亮到达终点!`);  //亮执行完之后，要开门！下一项任务才知道该轮到自己执行了  door();  },6000)//亮需要6s跑完全程  }  )  }  //然可以跑步  function ran(){  //让外界知道ran里有一个格子间  return new Promise(  function(door){  console.log(`然起跑...`);  setTimeout(function(){ //异步  console.log(`然到达终点!`);  //然最后一步执行完，一定记着开门！  door();  },4000)//然需要4s跑完全程  }  )  }  //东可以跑步  function dong(){  console.log(`东起跑...`);  setTimeout(function(){ //异步  console.log(`东到达终点!`)  },2000)//东需要2s跑完全程  }  // //让亮先执行，但是亮之后接着然继续执行  // liang()  // //return一个new Promise对象（格子间）  // //所有格子间都有.then()方法用来连接下一项任务或下一个格子间  // .then(ran)//千万不要加()，因为ran不是立刻调用，仅仅是和前一个任务连接起来而已  // //当亮中调用door()后.then中的ran()开始执行  // //ran结束后，又return一个new Promise对象(格子间)  // //所有格子间都可以继续.then()  // .then(dong)//千万不要加()，因为dong不是立刻调用，仅仅是和前一个任务连接起来而已  //所以:  liang().then(ran).then(dong);  </script>  运行结果:  亮起跑...  亮到达终点!  然起跑...  然到达终点!  东起跑...  东到达终点! |

总结:

\*\*\*\*ES6\*\*\*\*

1. 今后，拼接字符串都用模板字符串代替+号拼接

2. 今后，凡是声明变量都用let代替var。(不会被声明提前，不会超出代码块{})

3. 今后，几乎所有function都可用=>简写

特例: 只有不希望函数内this与函数外this保持一致时，才不能用箭头函数简写

4. 今后，凡是从头到尾遍历索引数组全部元素时，且只关心元素内容不关心元素位置的遍历都可用for of简写普通for循环

5. 今后，只要希望调用者没有传入实参值时，形参也有默认值可用就用：形参=参数默认值

6. 今后，只要希望定义函数时，可收集多个不确定个数的实参值时，都用...数组名，代替arguments。

7. 今后，只要希望调用函数时，先打散数组为多个值，再分别传参给函数时，都用...数组名

8. 今后，只要希望提取出数组或对象中个别成员值放在变量中单独简化使用时，就用解构。

var [变量1, 变量2, ...]=数组

var {属性名1, 属性名2, ...}=对象

9. 今后，只要多个实参值不确定有没有，但是又要求实参值必须对应的传给指定的形参变量时，就用参数解构:

(1). 定义函数时:

function 函数名({

形参名1=默认值1,

形参名2=默认值2,

... = ...

}){

... 函数体 ...

}

(2). 调用函数时:

函数名({

形参名1: 实参值1,

形参名2: 实参值2,

... : ...

})

10. 今后，只要希望定义一种类型，包含构造函数和原型对象方法，反复创建多个相同结果的对象，都要用class包裹

class 类型名{

constructor(形参列表){

this.属性名=形参;

... = ...

}

共有方法(){

... ...

}

}

11. 今后，只要两种class中包含部分相同的属性和方法定义，都可用两种类型间的继承;

class 父类型{

constructor(形参列表){

this.相同部分的属性=形参

... = ...

}

相同部分的方法(){ ... }

}

class 子类型 extends 父类型{

constructor(形参列表){

super(形参列表);

this.子类型独有属性=形参

... = ...

}

子类型独有方法(){ ... }

}

12. 今后，只要希望一项任务必须在一个异步任务之后执行，或者多个异步任务必须顺序执行，都必须用Promise技术，2步

(1). 定义前一项任务时:

function 前一项任务(){

//要让外界知道这个函数中有一个格子间

return new Promise( //创建一个格子间对象

//形参永远可以随意改名

function(door){//附赠了一个门door

原异步任务

原异步任务最后一步执行完，一定记得开门:

door() //只有开门，下一项任务才能开始执行

}

)

}

(2). 调用前一项任务时:

//.then()否则连接前后两项任务

前一项任务().then(下一项任务)

总结: this 判断this一定不要看定义在哪儿！只看调用时.前是谁或有没有new

4种:

1. obj.fun() this->.前的obj

2. new Fun() this->new正在创建的新对象

3. 构造函数.prototype.共有方法=function(){ ... } this->将来调用这个共有方法的.前的某个子对象

4. fun() 或 (function(){ ... })() 或 回调函数 this->window

今日对应小程序视频列表:

小程序->在线->JSCORE->day08 3. 回调地狱 callback hell 和promise...

作业:

1. 复习今日问题清单(小程序->首页->JSCORE->day08)，预习明日问题清单(小程序->首页->DOM->day01)

2. 看小程序视频学习封装发送ajax请求的函数

小程序->在线->JSCORE->day08 作业: 封装发送ajax请求的函数

强调:

(1). 不用开任何服务器端！！不需要nodejs和xampp！！直接用东哥新浪云服务器

(2). 视频中ajax请求的服务器端接口地址如果是http://localhsot:3000或localhost:5050，一律都改为"http://xzserver.applinzi.com/接口名" 即可