正课:

1. ES6

1. ES6:

解构:

1. 数组解构: 下标对下标

2. 对象解构: 属性对属性

var { 属性1: 变量1, 属性2: 变量2, ...}

={ 属性1: 值1, 属性2: 值2, ...}

结果: 变量1=值1, 变量2=值2

3. 参数解构: 其实是特殊的对象解构:

何时: 定义函数时，函数的参数个数和顺序都不确定时

如何:

1. 定义函数时，将所有参数定义在一个对象结构中

function fun({参数1, 参数2, ... ...})

2. 调用时:

fun({ 变量1, 变量2,...})

其中, 变量名要和参数名保持一致，

但是，顺序和个数不必一致！

优点: 调用函数时，参数的个数和顺序与函数定义时无关！

for...of...

简化for循环:

总结: 遍历数组: 3种方式

for(var i=0;i<arr.length;i++)

var elem=arr[i]

...

arr.forEach((elem,i,arr)=>{ ... elem ...})

for(var elem of arr){ ... elem ...}

for of vs for in

for of专门遍历索引数组或类数组对象——下标都是数字

for in 专门遍历关联数组或对象——下标是无规则的自定义名称

class:

1. 封装: class

什么是: 包裹构造函数和原型对象方法的一个程序结构

为什么: 传统构造函数和原型对象方法是分别独立定义的，不符合封装的要求

何时: 今后，只要创建一种类型都要包在class中

如何:

1. 用class{}包裹构造函数和原型对象方法

2. 构造函数名提升为class名，构造函数统一更名为constructor

3. 放入class中的原型对象方法，不再需要加prototype前缀，默认就已经保存在原型对象

简写: 放在class中的原型对象方法，不需要加function

2. 继承: extends

什么是: 让一个子类型继承父类型

如何:

1. class 子类型 extends 父类型

代替了setPrototypeOf()——不再需要！

2. 用super代替父类型.call(this,...)

super是专门指代extends后的父类型的关键词

super(参数值1, 参数值2,...)——不必再考虑this

3. 静态方法: static

定义时

class 类型{

...

static 方法名(){ ... }

}

调用时: 类型.方法名()

4. 访问器属性:

class 类型{

constructor( ... ){

... this.\_xxx 定义隐藏受保护的数据属性...

}

get 访问器属性名(){ return this.\_xxx }

set 访问器属性名(val){

只有验证val通过，才this.\_xxx=val

否则报错

}

}

Promise:

专门解决回调地狱: callback hell

什么是回调地狱: 因为回调函数嵌套层级太深导致的代码可读性差

何时: 只要要求两个异步调用的函数，必须先后执行时

如何:

1. 定义函数时:

function ajax(参数){ //去掉回调函数参数

return new Promise(function(open){

...办完自己的事儿后

open(要传给下一步的参数)

})

}

2. 调用时:

ajax(...) //return Promise

.then(function(){

return ajax(...) //return Promise

})

... ...

.then(function(){

...最后一步...

})

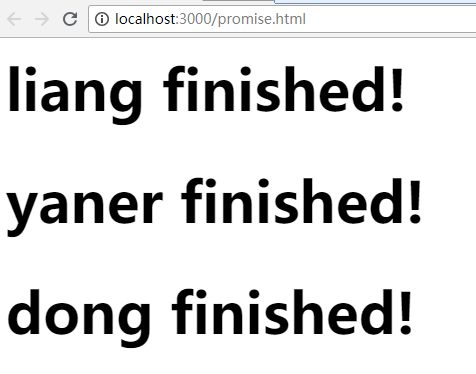
课后任务:

(1)复习: 复习ES6新特性

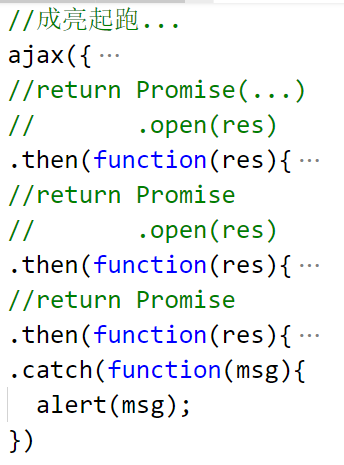
(2)作业: 完成课后练习:  
 题目要求:

使用Promise技术控制异步方法顺序执行，并支持错误处理

运行效果:



提示:



(3)项目:

题目要求:

运行效果: