正课:

1. ES6:

2. DOM:

1. ES6:

for of: 简化for循环遍历

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

var elem=arr[i]

arr.forEach(function(elem,i))

for(var elem of arr) 遍历数字下标的索引数组或类数组对象

vs for in 遍历关联数组和类数组对象

OOP:

封装encapsulate:

1. 对象直接量: 2个简化:

var obj={

year:year, //可简写为year,

intr:function(){ //可简写为: intr(){ ... }

}

}

2. 构造函数:

class Student {

constructor(sname,sage){

this.sname=sname;

this.sage=sage;

}

intr (){ //直接放在class中的方法，默认存在prototype中

... ...

}

}

//强调: class中不用逗号分隔

继承: 两种类型间的继承:

class Father{

...

}

class Child extends Father{

constructor( ... ){  
 super(...)  
 }

}

访问器属性:

class 类型名{

constructor( ... ){  
 需要添加一个受保护的数据属性  
 }

get 访问器属性(){ return this. 数据属性}

set 访问器属性(val){ 验证val, 并赋值}

}

静态方法:

class 类型名{

constructor( ... ){

... ...

}

static 方法名(){

}

}

静态 vs 非静态:

1. 存储位置:

非静态 存储在原型对象中

静态 存储在构造函数对象上

2. 调用方式:

非静态 子对象.方法()

静态 构造函数.方法()

3. 哪些对象可用:

非静态方法 只有该类型的子对象可用

静态 所有对象都可用

Promise: 代替异步调用的回调函数

为什么: 解决回调地狱(callback hell)

何时: 只要多个异步调用的函数，要求必须顺序先后执行

如何:

定义时: 前一个函数:

function fun(){

return new Promise(function(open,err){

//原fun函数要执行的操作

如果fun函数正确执行,

则调用open()打开开关，继续向后执行

否则

调用err()打开错误处理开关

})

}

调用时: fun().then(后一个函数名)

强调: 后一个函数不要加()，不是立刻调用！

.catch(function(){ ... 错误处理 ... })

等待多个异步调用都完成，才执行:

每个异步操作必须都是Promise

如何: Promise.all([

异步任务1(),

异步任务2(),

异步任务3()

]).then(function(){ ... 后续任务 ...})