1. 为什么要是用ES6/7/8/9/10?

问题：ES3/5

1. 没有类class 没有extends继承

--构造器函数当做类，混合模式

1. 代码冗余

--function

--将数组元素赋值给变量

--将对象成员赋值给变量

1. 可读性差、可维护性差

$.ajax({参数非常多}) 地狱回调

导致：JavaScript做大型企业级应用开发存在很多问题

**解决：ES6/7/8 重点重点**

1. 类class extends继承
2. 块作用域—变量冲突范围更小
3. 解构赋值

--将数组元素赋值给变量

--将对象成员赋值给变量

4. 隐式迭代、链式写法

--隐式迭代—iterator

5. 通过promise/generator/async await 改写异步增强可读性

6. 增加proxy对象拦截—代替ES3对象劫持

7. 数组Array和String增加了大量新API，增加功能

8. 支持模块化导入和到出

--导入import

--到出export /export default

9.增加一个JavaScript原始数据类型Symbol

10.增加新数据结构/对象 Set Map WeakSet WeakMap

说明：ES6大部分语法都是ES3/5语法糖，ES6大部分语法可以转换为ES5代码

ES6 ES5

let test=(x,y)=>x+y; var test=function(x,y){return x+y;}

1. **ECMASCRIPT和JavaScript之间关系：重点**

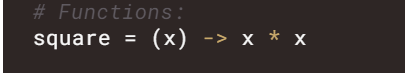
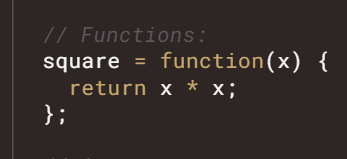
ECMASCRIPT是JavaScript语言的标准

JavaScript是ECMASCRIPT标准的具体实现

说明：JavaScript和ActionScript都属于ECMASCRIPT标准

说明：Coffeejs是ES3/5语法糖，Coffee浏览器不支持，所有运行前需要将Coffeejs转换为JavaScript才可以运行

Coffeejs ES3/5

说明：less/scss是css语法糖

**ES6基本语法：**

1. **新增块作用域：重点重点**

**ECMASCRIPT6中的块作用域:**

**1. if 块**

**if(){}**

**2. while 块**

**while(){}**

**3. 函数块**

**function foo(){}**

**4. for 循环块**

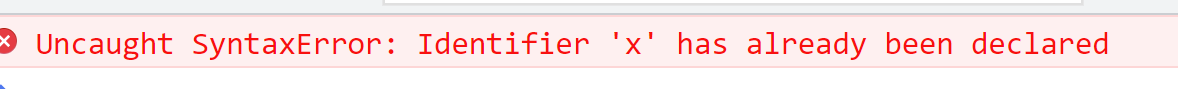
**for(let i = 1; i<100; i++){}**

**5. 单独一个块**

**{}**

**块作用域使用:let 或 const**

**说明:同一个块作用域，let不允许声明多个同名变量**

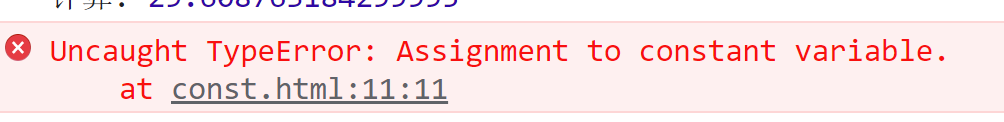
****

**说明：块作用域主要减少代码变量命名冲突**

1. **常量const：**不会变化的变量，只能在创建处修改，创建后变量值不能被修改

如果修改报错！

开发场景：开发中有些变量值非常重要，防止在后面的代码中意外修改，我们将其定义为常量



问题：ES3/5中，开发的时候，使用匿名函数会出现大量的function关键词，写起来特别冗余；如果让函数内this，指向函数外对象，还得加bind(this)

解决：使用ES6提供的箭头函数—匿名函数

1. **箭头函数:Es6匿名函数—重点重点重点重点**
2. 没有形参写法:

let 函数名=()=>只有返回值不用加{}

let 函数名=()=>多条语句需要加{}

let 函数名=()=>只有返回值并且是一个对象需要加()

1. 只有一个形参的写法

let 函数名=形参=>同上

1. 一个以上的参数

let 函数名=(形参1,形参2,……)=>同上

1. 可以设置默认值 重点重点重点重点重点重点重点重点

let 函数名=(形参1=值,形参2,……)=>同上

**说明：调用函数的时候如果传递了实参，参数就是实参值，如果没有传递实参，那么就是参数默认值**

**面试题：箭头函数和ES5普通函数有什么区别：**

1. 箭头函数是匿名函数，不能作为构造函数，不能使用new
2. 箭头函数不绑定arguments，取而代之用不定参数…arg
3. 箭头函数中this，指向函数外对象，相当于自带bind(this)
4. 箭头函数没有原型属性
5. 箭头函数不能当做Generator函数,不能使用yield关键字

说明：babeljs JavaScript编译器—JavaScript转换

ES6/7/8-🡪转成ES5

React jsx🡪转成javascript

Vue jsx->转成javascript

<https://babeljs.io/>

ES5没有类的概念，只好拿构造器函数当做类

*类和子类将来就是React 类组件*

1. **类class 继承 extends 重点重点重点重点重点重点重点**
2. 创建一个类：父类

class 类名{//类名必须用Pascal命名法

//constructor是构造器函数

//功能:给新建的对象添加属性

//当new 类名(),js引擎**自动**调用构造器函数，为新建对象添加属性

constructor(形参1,形参2,形参3,……){

this.属性=形参1;

this.属性=形参2;

this.属性=形参3;

}

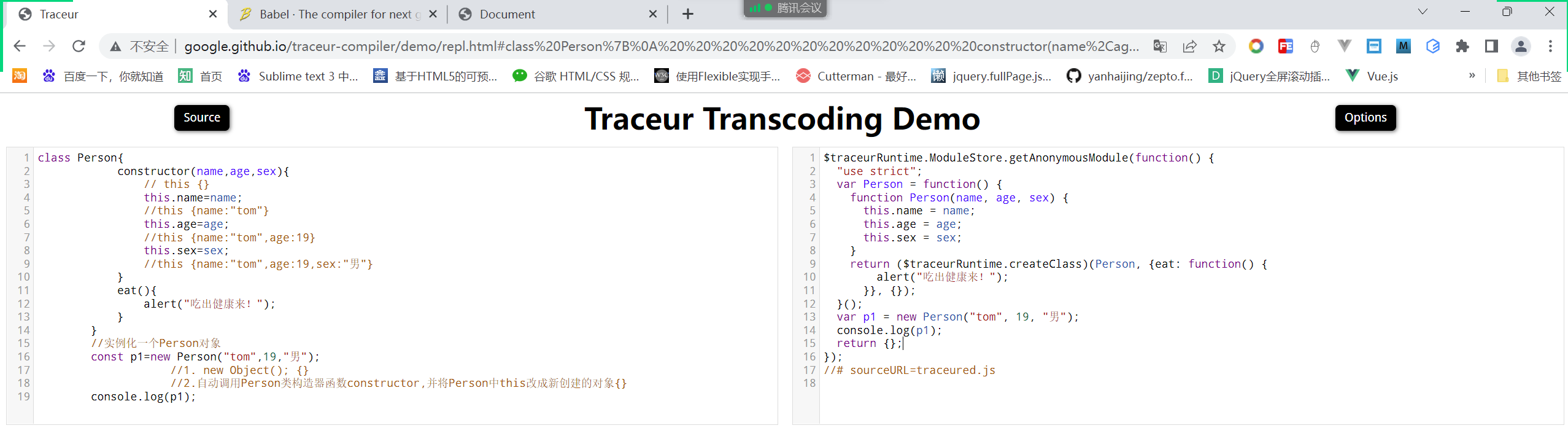
//原型上的方法

//ES6类或对象中方法，可以省略function

方法名1(){}

方法名2(){}

}



//子类继承父类的属性和方法—实现代码高度复用

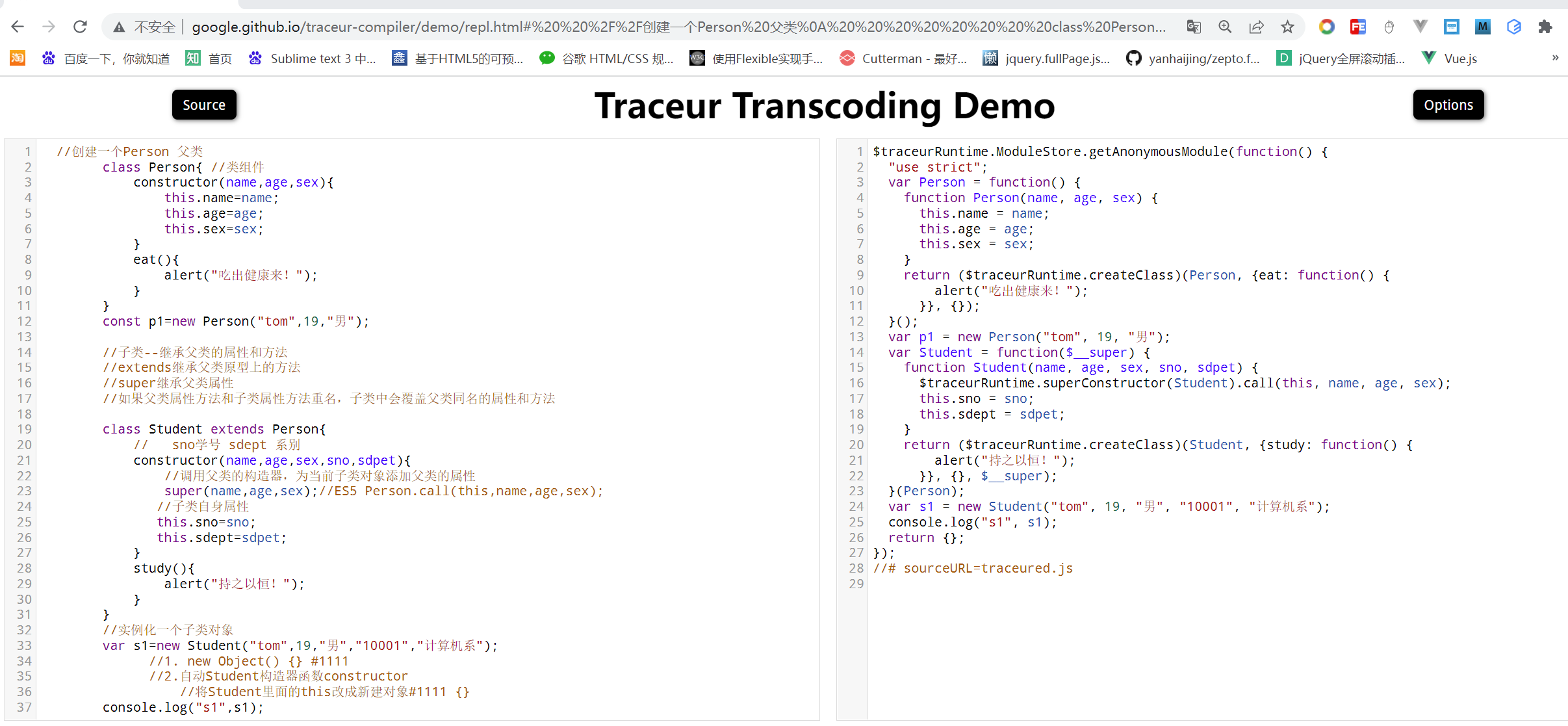
extends 继承原型上的方法---相当于原型拷贝

super 继承父类的属性---相当于 父类构造器.call(this,实参,实参)

super调用父类构造器，给当前对象添加父类的属性—实现属性继承

super超类—超级方法

*特别注意：super必须是constructor第一条语句*

**

**五、\_\_proto\_\_ 增强字面量：功能实现对象的继承**

回顾：ES5 \_\_proto\_\_ 原型对象，实现同类型对象方法或属性共享

class类的继承:extends super

对象的继承\_\_proto\_\_

1. **字符串模板—重点重点重点重点重点**

ES5字符串拼接存在的问题：+

“字符串”+变量+3+变量+“字符串” 太麻烦

解决：ES6字符串模板/模板字符串

`` 里面内容都是字符串，也可以回车

如果要嵌入变量或表达式 ${ 变量或表达式 }

**模板叫法来自于后端后端模板：**

HP和html混合到一起的--前后端无法分离开发：

解决：

1.ajax

2.后端模板 thinkphp smarty

开发的时候将前端代码和后台代码分离

前端只写:.**html 模板页**

--如何嵌入PHP变量或对象 {{ 变量或表达式}} {{userName}}

后端只写.php

运行的时候，后端模板再将前端代码和后端代码混合到一起

{{userName}} 替换成 tom

react 模板 { }

angular vue模板 {{ }}

1. **解构赋值**：重点重点重点重点

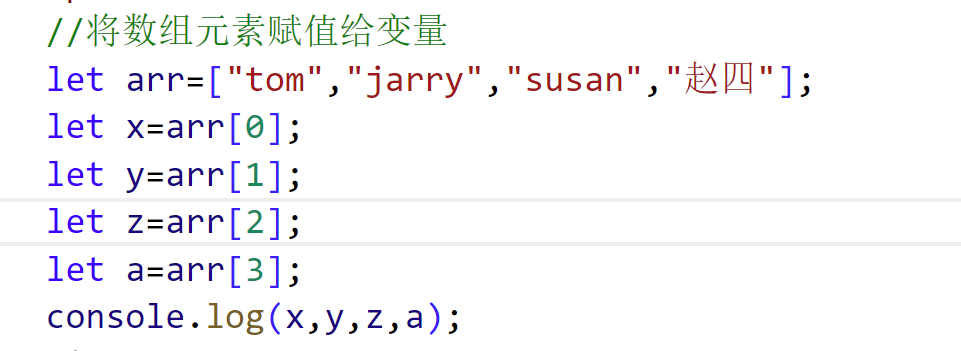
将数组元素赋值给变量

将对象指定属性的值赋值给变量

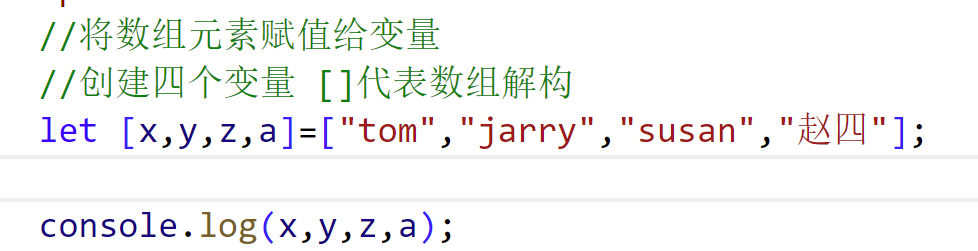
说明：使用了ES6基于iterator遍历器对象实现的**隐式迭代**

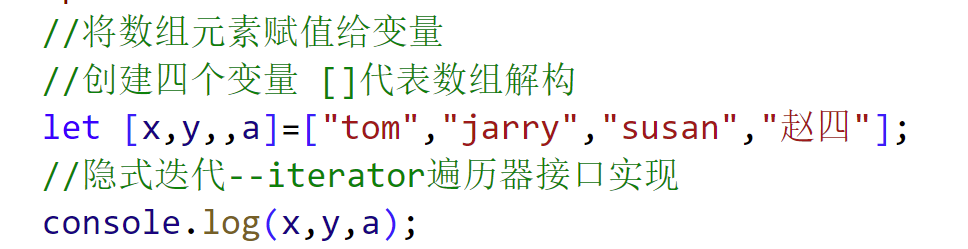
1. 数组不解构赋值，将数组元素赋值给变量

缺点：代码冗余

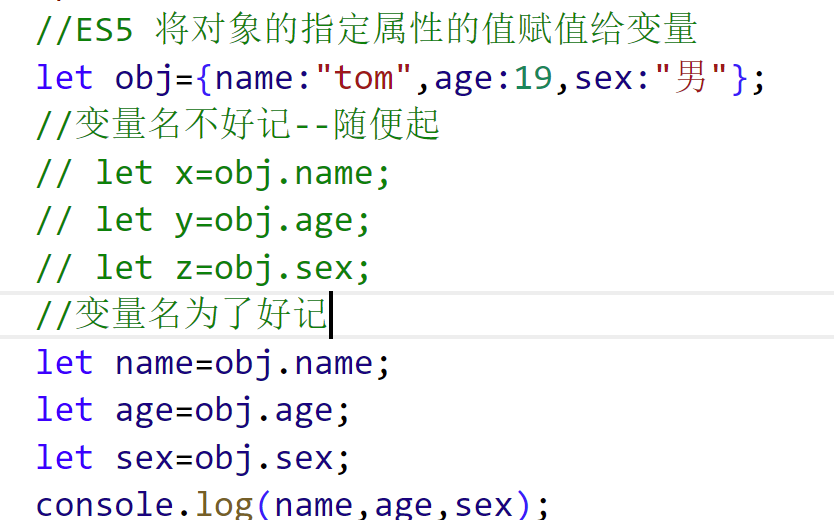


1. **数组解构赋值：**

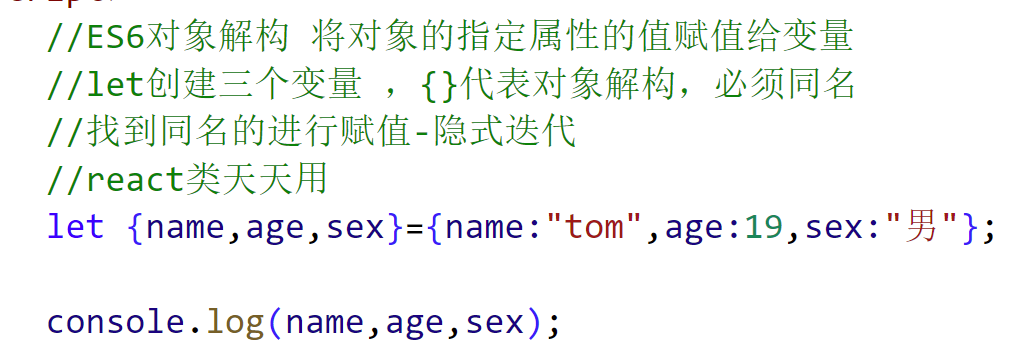
****

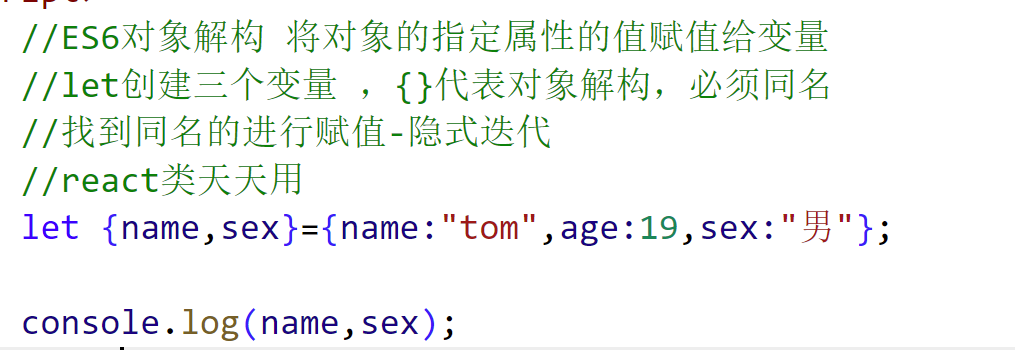
****

1. **对象不使用解构赋值，将对象指定属性赋值给变量**

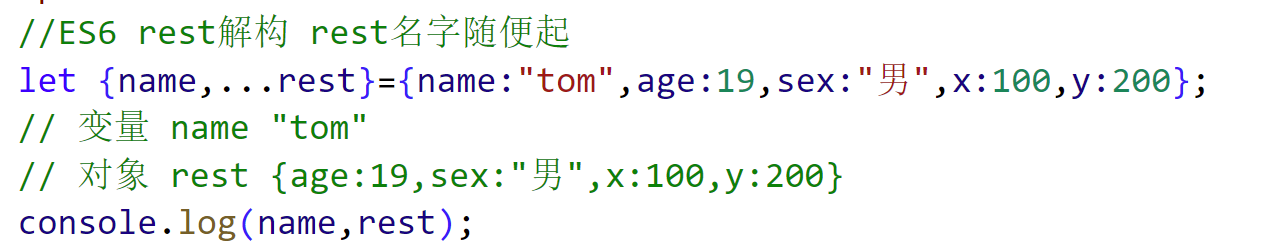
****

1. **对象使用解构赋值，将对象指定属性赋值给指定变量**

****

****

**对象rest解构：**

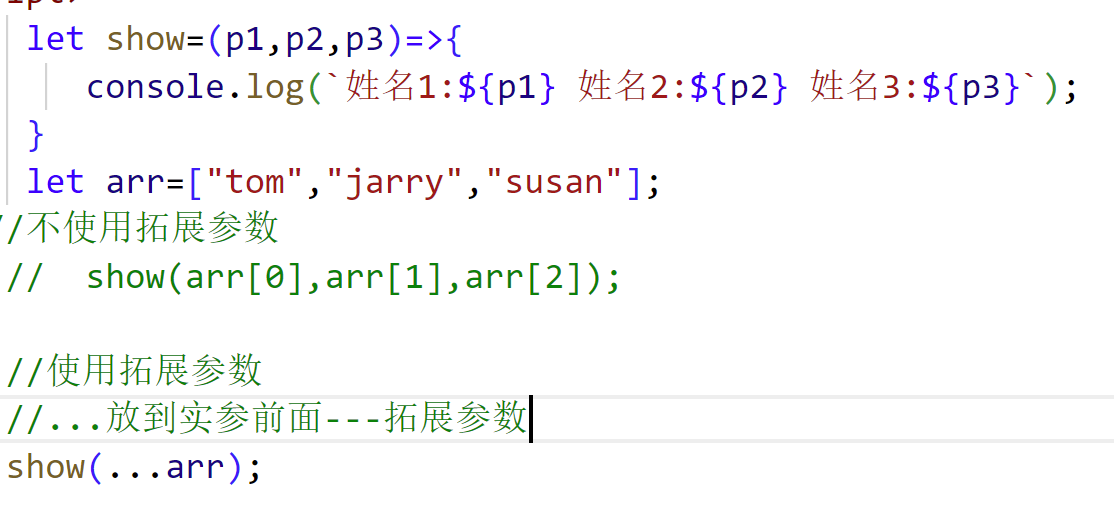
****

**七、…拓展运算符：原理实现隐式迭代，基于ES6 iterator接口 重点重点重点重点重点**

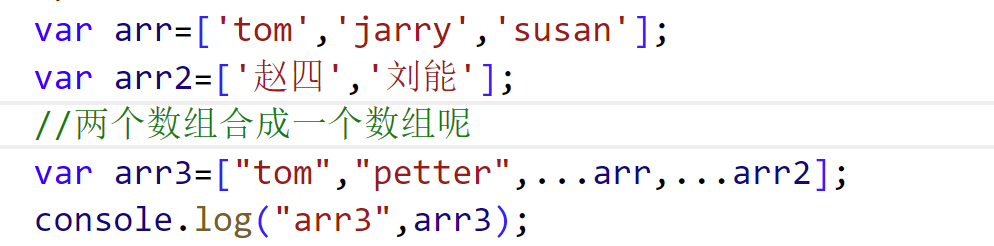
1. …形参 --不定参数、



1. …实参 --拓展参数

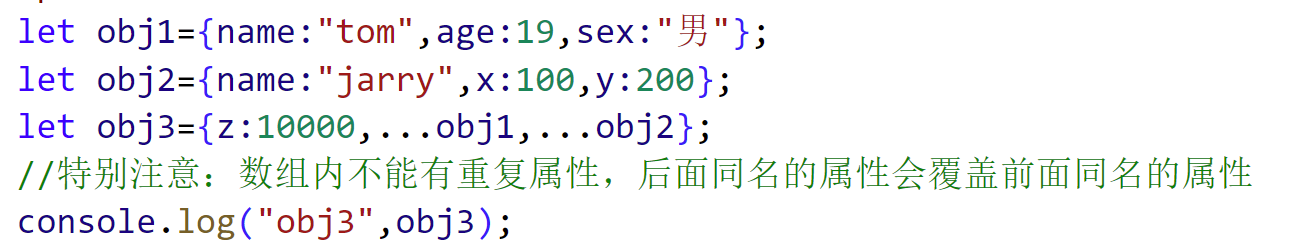


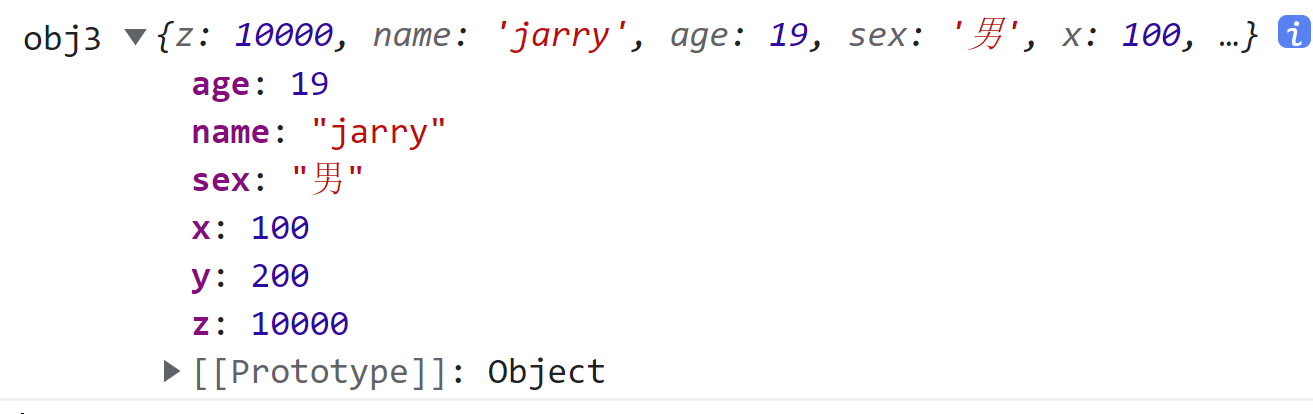
1. …数组 --数组拓展-展开





4. …对象 --对象拓展-展开





**八．ES6新增的数据结构：Set WeakSet**

Map WeakMap

JavaScript中数据结构：Array/Object/Set/Map

1. Set结构：唯一性—数组去重

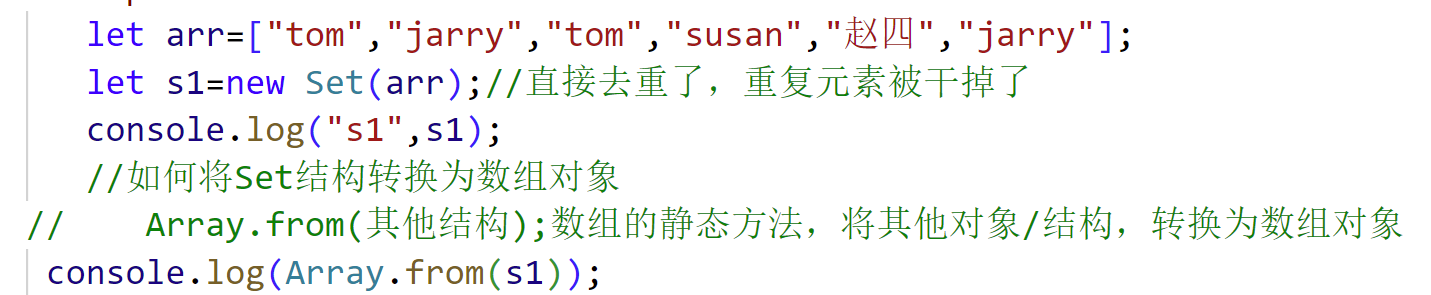
特点：唯一性—元素没有重复的

只有值没有键



说明：Set结构类似于数据

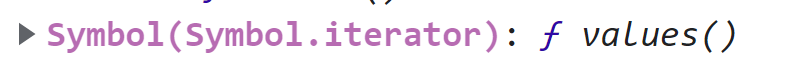
1. 数组去重



1. 遍历Set对象：只有值没有键

Set对默认部署的是 values遍历器

ES6 for of 结构遍历任何部署了遍历器对象接口的对象



for of遍历的是遍历器对象/接口

        只要部署了 Symbol.iterator接口的对象

        都可以使用for of进行遍历

        部署的是values值遍历器，取得就是值

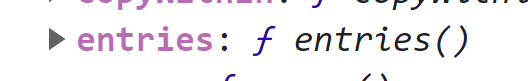
        部署的是keys键遍历器，取得就是键

        部署的是entries成员遍历器，取得就是键值对组成的数组

数组举例：数组默认部署的是values遍历器接口



也提供keys、entries遍历器接口





**九、WeakSet 弱引用Set结构：重点**

WeakSet和Set结构异同：

1. 相同点:

唯一性，去重

1. 不同点
2. WeakSet成员只能是对象
3. 垃圾回收机制不考虑，WeakSet的引用，成员随时会被垃圾回收机制回收

WeakSet成员—别的对象无法引用，无法遍历

说明：WeakSet引用对象/DOM对象，不会引发内存泄露

**十、Map结构：值值对：重点**

-值可以做为键名

说明：Object对象哈希结构: “字符串”值对，键都是字符串

**十一、iterator 遍历器：ES6隐式迭代 重点**

ES6提供iterator就为数据结构（Array,Set,Map），提供统一的遍历方式，凡是部署了**Symbol.iterator**属性的对象，都可以使用for of进行遍历，也可以使用…拓展运算符

for of

…

都是基于ES6 iterator遍历器

遍历器对象.next() 用来遍历对象Array Set Map的成员

next方法返回值： {value:成员值,done:标识遍历是否结束}

**十二、generator函数：**解决地狱回调问题 重点，重点重点—直接决定你能否学会react 核心技术redux-saga

**Generator函数**是一个状态机，封装了多个内部状态。

照相机 多张胶片

重要说明：调用generator函数，返回一个遍历器对象，然后通过遍历器对象.next()

可以逐个遍历generator内部的每个状态yield

**generator函数两个特征:**

1. 关键字与函数名之间有一个星号;

2. 函数体内部有yield语句，定义不同的内部状态



说明：axios.get(“url”) 返回一个promise对象

只有promise对象，才有.then 方法

遍历是否结束 done属性

done:false 遍历未完成

done:true 遍历完成—return/函数调用结束

递归函数：函数内部有自己调用自己的语句

1. 永动
2. 遍历generator的每一个状态

run(generator函数); 自动遍历每个状态

将来run方法，就是redux-saga中间件提供的用来遍历generator函数每个yield状态的递归方法---

**遍历器对象.next(“实参”); 给当前这条yield的返回值赋值**

**地狱回调：后面的异步操作，依赖于前面执行的结果，层层嵌套**

可读性差，可维护差，降低开发和维护的效率

$.ajax({

url:””,

type:””,

dataType:””,

success:function(){},

complete:function(){

$.ajax({

url:””,

type:””,

dataType:””,

success:function(){},

complete:fucntion(){

$.ajax({

url:””,

type:””,

dataType:””,

success:function(){},

})

}

})

}

})

**面试题：如何解决地狱回调：**

1. generator

2. async await

**十三、ES5严格模式：重点重点重点—面试题**

问题： 为什么使用ES6/7/8进行项目开发，为啥要学ES5严格模式？

答：1. 为了兼容更多浏览器，打包的时候会使用babel将ES6/7/8转换为ES5代码的严格模式

2. ES6自动就是严格模式

3. 严格模式下，减少很多bug

说明：最终在浏览器中跑的JavaScript是ES5严格模式代码

ES5中，设立"严格模式"的目的，主要有以下几个：

-- 消除代码运行的一些不安全之处，保证代码运行的安全；

-- 提高编译器效率，增加运行速度；

-- 为未来新版本的Javascript做好铺垫。