es6的理解1

目录：

let——和var的区别

const

特性：暂时性死区

解构赋值

[]

{}

**属性简洁**

**函数简洁**

**属性表达式可以作为对象的属性名**

字符串扩展

数组扩展

对象扩展

Object.getPrototypeOf(实例对象)

Object.setPrototype()

new Proxy(实例，处理器)——new Proxy(obj，{get(){}})

**Reflect**.ownKeys(***obj***)

函数扩展

**默认值**

**剩余参数——fn(arr,...data){}**

**扩展运算符 ...arr**

箭头函数

参数

函数体

返回值

特性：this，arguments，不能是构造函数

数据结构

set

map

主体：

**let**——和var的区别

特性：

1.不允许重复申明，否则会报错

2.暂存死区（先申明后使用），没有预解析机制，在申明之前就是死区

3.支持快级作用域，比如if,for,switch...

clipboard.png

{}：因为支持块级作用域，所以类似闭包

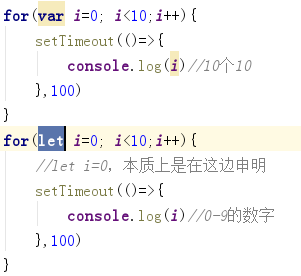
闭包：函数套函数，并且外部可以访问内部变量。作用：内部函数可以访问外部函数的变量数据；当内部作为外部函数返回值返回，并被外部数据引用，那么函数内容数据会长期驻留内容，不被垃圾回收。

主要原因：（下面的例子）

从两方面来解释：作用域和异步处理。

用var，setTimeOut是异步处理，if处理完之后，setTimeOut会等时间到了再执行，因为setTimeOut里面没有i，会根据作用域查找规则作用域链找i值，找到外面的i值，现在i值是10，所有打印了10个10。

用let，setTimeOut是异步处理，if处理完之后，setTimeOut会等时间到了再执行，因为setTimeOut里面没有i，会根据作用域查找规则作用域链找i值，找到外面的i值，因为let可以支持块级作用域，所以在里面会申明一个let。根据作用域链找了这个let。



clipboard.png

**const**

特性：

1.不允许重复申明，否则会报错

2.暂存死区（先申明后使用），没有预解析机制，在申明之前就是死区

3.支持快级作用域，比如if,for,switch...

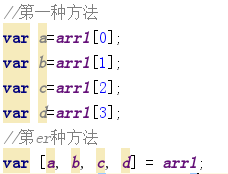
4.初始化必须赋值，要不然报错

clipboard.png

**解构赋值**

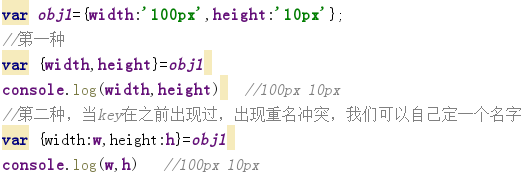
**[]**——内部其实走了两步：第一步申明变量，第二步赋值

特征：数组的解构赋值，位置需要一一对应



**{}**——

特征：对非数组的对象进行结构赋值，那么左侧变量的位置不在是一一对应的关系，而是key值对应



涉及的知识：

**属性简洁**表示法/**对象简洁**表示法

当key与值（变量）名称相同的时候，可以简写

**函数简洁**表示法

obj={

abc(){

}

}

以前：

**obj**={

abc:**function** () {

}

}

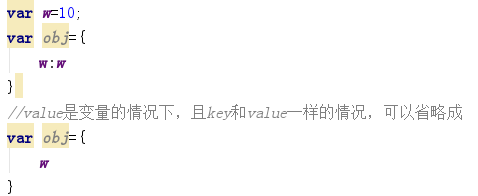
**属性表达式可以作为对象的属性名**——因为：[]表达式，函数解析

var a='miao'

var obj={

[a]:1

}



字符串扩展

数组扩展

对象扩展

**Object.getPrototypeOf(实例对象)**

**Object.setPrototype()**

分析：

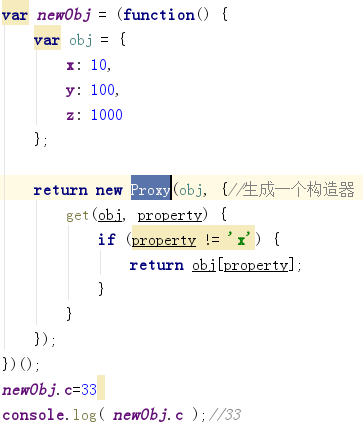
之前我们可以用，实例.\_proto\_取原型，但是浏览器不建议这样写，因为考虑到会出现赋值覆盖，原先的就找不到了，还考虑到数据结构统一性。

现在可以通过Object.getPrototypeOf(实例对象)获取原型。

**new Proxy(实例，处理器)**——相当于产生一个助理

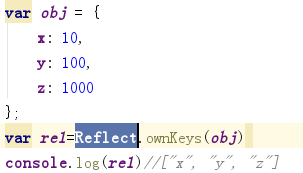
作用：一个构造函数，通过Proxy构造创建一个对象的代理对象

意义：生成一个构造器，就不用直接操作原对象，想传什么属性，就可以传什么属性出来。在外面也可以修改



**Reflect**.ownKeys(***obj***)——返回值是[key]组成的数组

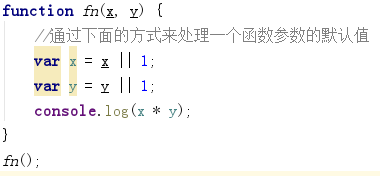
和Object.keys类似

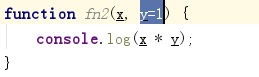


函数扩展

**默认值**：es6可以通过形参来设置默认值

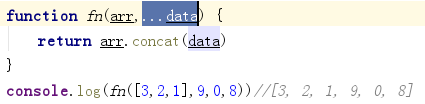
注意：有默认值的参数最好写在参数列表的最后面





**剩余参数**——fn(arr,...data){}

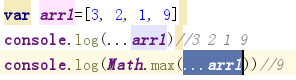
解释：函数第一个传入的是arr数组，后面是不定个数的参数，可以把不定的参数存在data数组里面，函数内部就可以方便调用了。



**扩展运算符 ...**

作用：和剩余参数功能相反——把数组变成参数列表，就是打散数组数据成单独单元。

应用：取数组里面的最大值



clipboard.png

箭头函数

**形式**——省了function

参数列表=>函数体

**var** *fn*=(a)=>{

**console**.log(a)

}

**参数**

没有参数**（）**，必须要用括号

一个参数**（a）**或 **a**，可以省略括号

二个几以上参数**（a,b）**，必须要用括号

**函数体**

一句代码，可以**省略{}**

**var** *fn*=(a)=>**console**.log(a)

**返回值**

一句代码：可以**省略return**，执行结果就是返回值

其他形式：如果没有return，返回值就是undefined

注意：

如果仅有的一条语句是一个字面量对象,而且需要返回一个值，那么就必须加return，因为js不知道 {}是函数体的大括号，还是字面量对象的{}。

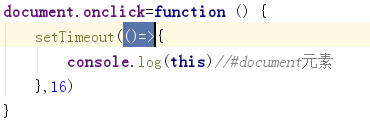
**特性：**

**this**，取决于申明期。普通函数this取决于执行期

**arguments**，没有，解决：用剩余参数...data

**不能是构造函数**，不能被new

注意：事件函数不要用箭头函数，this指向就不一定指向调用该事件的对象



数据结构

set

作用：集合，是js中新增一种数据结构，类似数组，但是Set中每一个数据是唯一的，不重复的

特征：不重复的，不能获取

创建：new Set( | [])

属性：

size。类似length

方法：

add()添加集合

clear()清空集合

delete()删除集合指定值

has()是否包含

forEach()

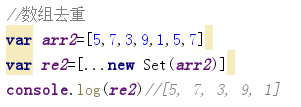
keys()

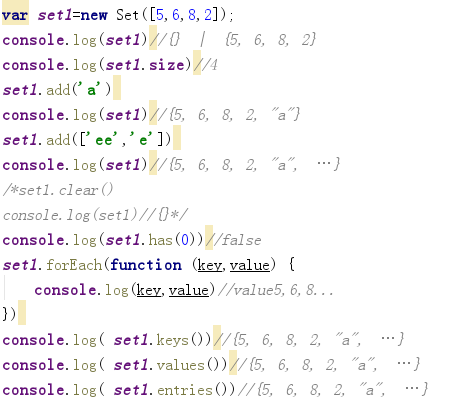
values()

entries()

应用：数组去重

解释：利用set数据不能重复的特性，将重复的数据剔除，并用扩展符...将set数据打散，最后将打散的数据用【】括起来。





map（{}|[],值）

对象的key只能是字符串

key可以是任何,可以是对象

属性：

size : 返回成员总数。

方法：

set(key, value) : 设置一个键值对。

get(key) : 读取一个键。

has(key) : 返回一个布尔值，表示某个键是否在Map结构中。

delete(key) : 删除某个键。

clear() : 清除所有成员。

