**十三 、javascript-ECMA6**

**一、解构**

传统的声明变量：var x = 10, y = 20, z = 30;

**1.使用解构书写：**

let [x, y, z] = [10, 20, 30];//结果为x = 10, y = 20, z = 30；

let [x, y, z] = [10, 20];//结果为：10,20,undefined;

【好处1】交换两个变量的值

let [x, y] = [10, 20];

alert(x + ", " + y); //x=10,y=20;

[x, y] = [y, x];

alert(x + ", " + y); //x=20,y=10;

【好处2】函数返回多个值。

function exec(){

return ["结果a", "结果b", "结果c"];

}

let [a, b, c] = exec();

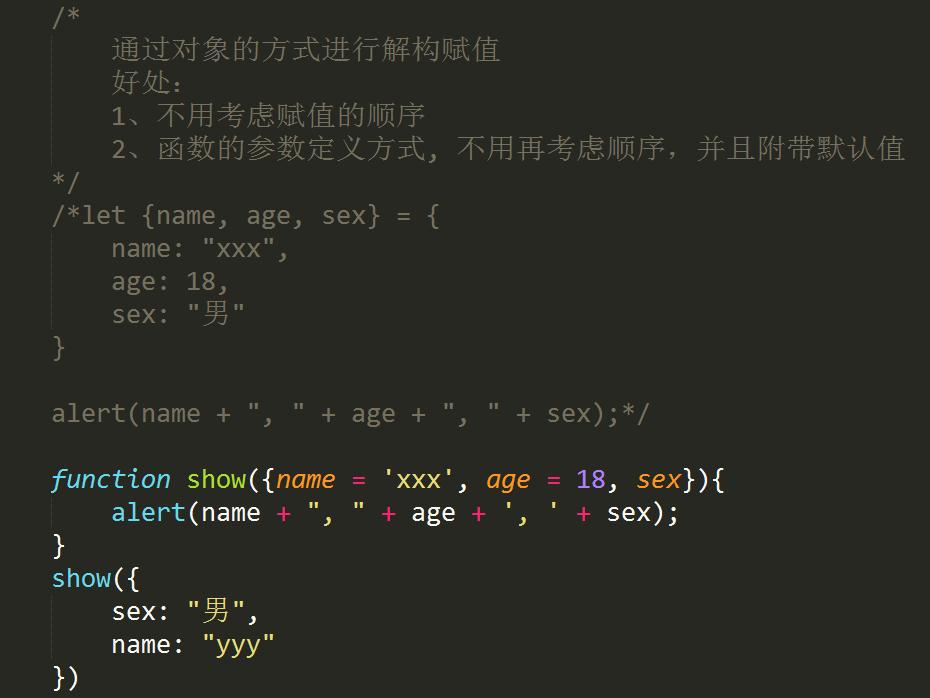
alert(a + ", " + b + ', ' + c); //结果a, 结果b, 结果c;

**2.通过对象的方式进行解构赋值：**

好处：

(1)不用考虑赋值的顺序

(2)函数的参数定义方式, 不用再考虑顺序，并且附带默认值

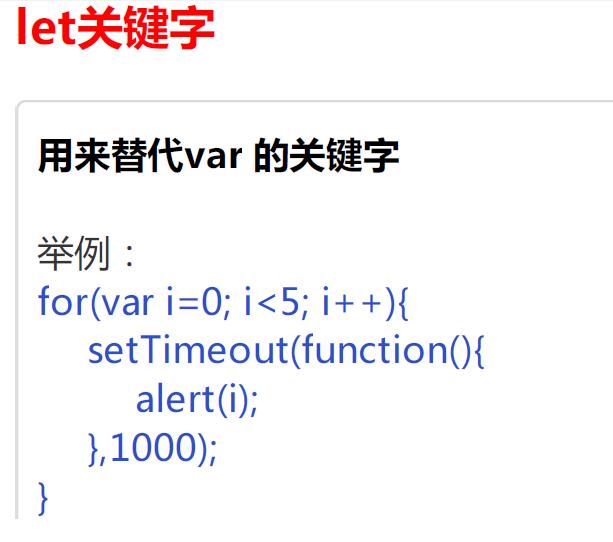


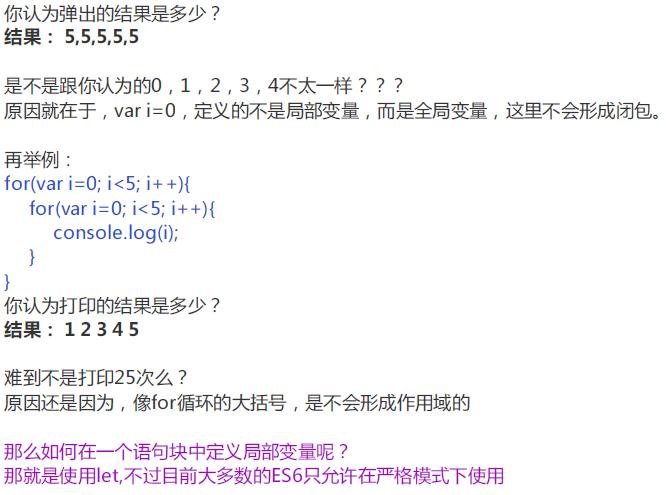
**二、let关键字**

传统局域作用域：通过函数的大括号。

let关键字声明变量

【注】let关键字声明的变量，会生成块级作用域。





**通过let可实现打印25次**

for(let i = 0; i < 5; i++){

for(let i = 0; i < 5; i++){

console.log(i);

}

}

**三、const**

**声明常量 一旦声明，就变成了常量，没有办法被修改。**

const BLACK = "#FFFFFF";

alert(BLACK);

**四、箭头函数**

箭头函数将原函数的 function、return、大括号全部省略。

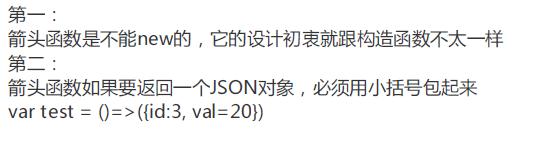
**传统函数：var test = function(x){**

return x + 2;

}

**箭头函数：var test = x => x + 2;**

【缺点】：



当返回值是对象的时候，需要在外面加小括号返回

var test = () => ({id: 1, num: 2});

alert(test().id);

【好处】 箭头函数，可以自动去绑定this

var obj = {

left: 100,

move: function(){

setTimeout(() => {

alert(this.left);

}, 1000);

}

}

obj.move(); //100

**五、字符串**

**【``】字符串**

1、遇到换行，可以自动去识别后续的字符串。

【传统】var str = "hello

world";

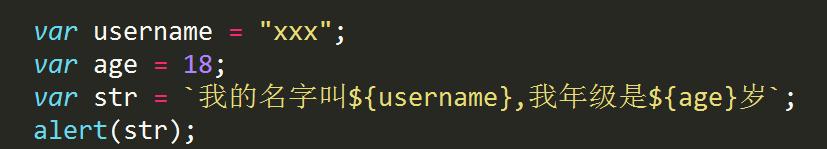
alert(str); //换行不识别

**【``】**var str = `hello

world`;

alert(str); //换行可识别

2、我可以通过占位的方式去拼接字符串 **${}**。



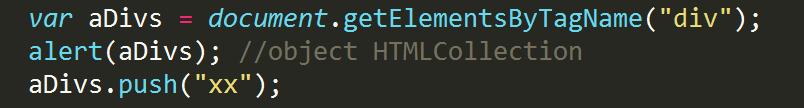
**六、数组新方法**

**1.将伪数组转成数组**

Array.from(伪数组)

【返回值】数组

【例】



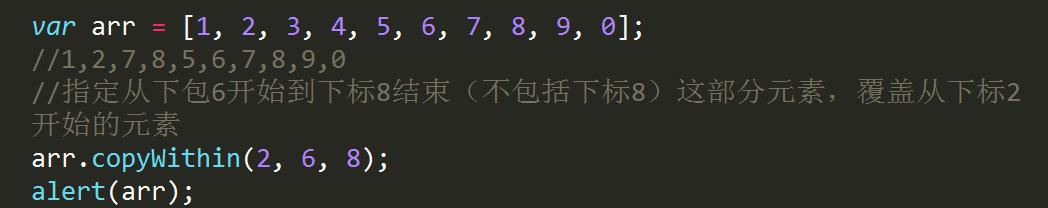
此时aDivs无法使用数组的操作push;

**通过Array.from将伪数组转为数组即可操作**

aDivs = Array.from(aDivs);

aDivs.push("xxx");

**2.copyWithin()**



**3.find()**

功能和filter差不多。

【注】找到第一个符合条件的元素，只找一个

var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0];

var newArr = arr.find(function(item, index, array){

return item > 5;

})

//找到第一个大于5 的元素

**4.findIndex()**

【注】找到第一个符合条件的元素的下标

var tmp = arr.findIndex(function(item, index, array){

return item > 5;

})

alert(tmp);

//找到第一个大于5 的元素对应的下标；