1. let和const命令

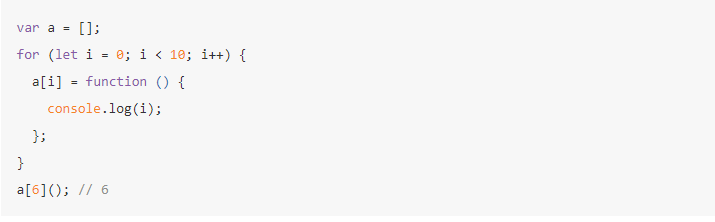
1>.let和const命令只在它自己所在的代码块有效

2>.let和const命令不存在变量提升

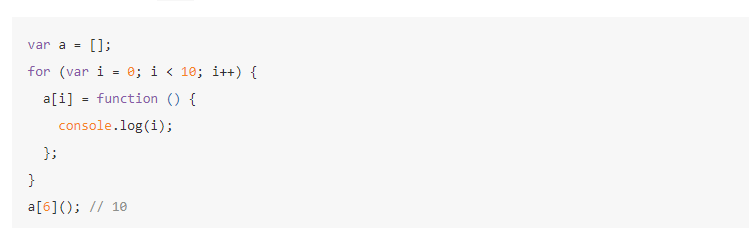
3>.let和const命令都存在暂时性死区(需先声明后访问)

4>.在代码块中不得重复声明变量

-🡪let声明的for循环循环外具体访问循环数返回当前循环数



而var声明的返回最后一个循环变量



二.let和const的区别

const声明的变量只读，一但声明，常量的值就不能改变。(地址不可改变)

-🡪 const PI = 3.14

PI = 3 (执行这行会报错)

🡪const a = [];

a.push('Hello'); // 可执行

a.length = 0; // 可执行

a = ['Dave']; // 报错

常量a是一个数组，这个数组本身是可写的，但是如果将另一个数组赋值给a，就会报错。如果真的想将对象冻结，应该使用Object.freeze方法

三．数组的解构赋值(解构赋值不仅适用于var，也适用于let和const)

1>.🡪

var [a, b, c] = [1, 2, 3] (可以从数组中提取值，按照对应位置，对变量赋值)

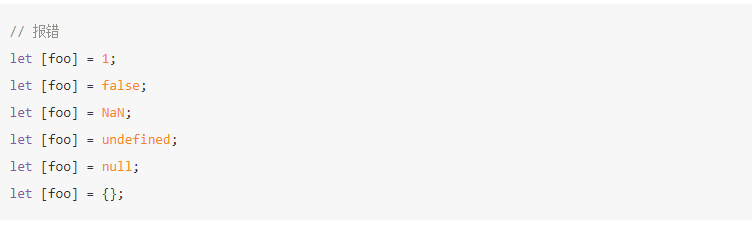
//a = 1, b = 2, c = 3

注意：如果数组里面还套有数组去前面一位的值 🡪

let [a, [b], d] = [1, [2, 3], 4];

//a = 1, b = 2, d = 4

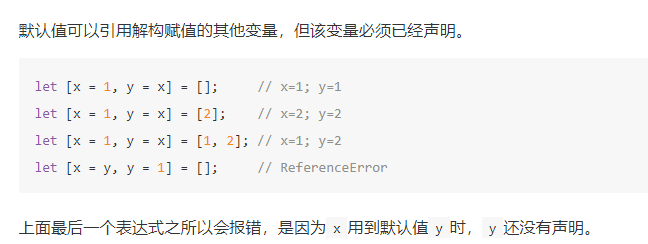
如果等号的右边不是数组（或者严格地说，不是可遍历的结构，参见《Iterator》一章），那么将会报错。

🡪

2>.默认值



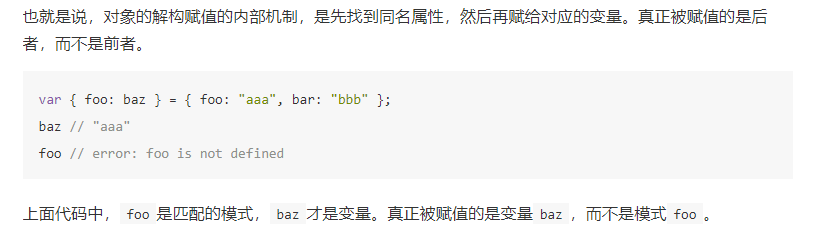
注意：有默认值和赋的值时取赋的值且默认值是function的时候不会执行

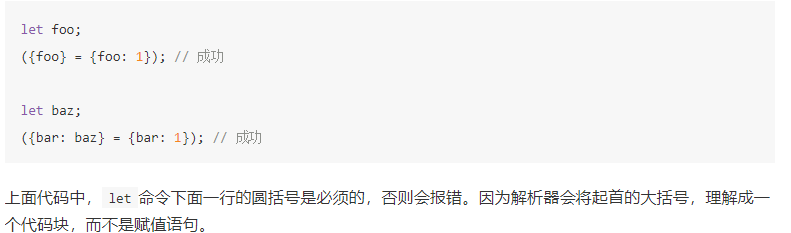


三．对象的解构赋值

1>.对象的解构与数组有一个重要的不同。数组的元素是按次序排列的，变量的取值由它的位置决定；而对象的属性没有次序，变量必须与属性同名，才能取到正确的值。

2>.



3> 

4>.对象嵌套赋值



5>.大括号写在行首, JavaScript将其解释为代码会将其解析成代码块，可以使用放在一个圆括号里面，就可以正确执行🡪

var x;

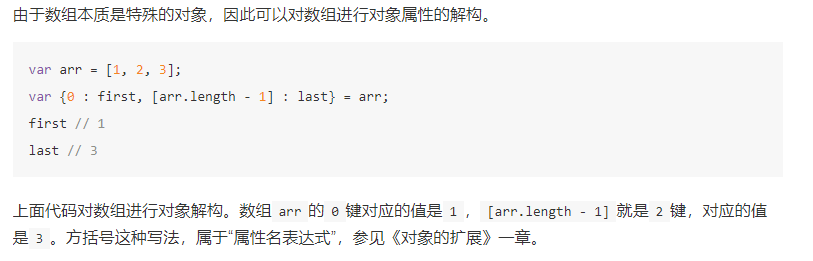
{x} = {x: 1}; //报错

上面代码的写法会报错，因为JavaScript引擎会将{x}理解成一个代码块，从而发生语法错误。只有不将大括号写在行首，避免JavaScript将其解释为代码块，才能解决这个问题。

// 正确的写法

({x} = {x: 1}); //放在一个圆括号里面，就可以正确执行

6>.数组是特殊的对象-🡪



四．字符串的结构赋值

1>.字符串被转换成了一个类似数组的对象。



五．数值和布尔值的解构赋值



六．函数参数的解构赋值

等号右边是数值和布尔值会先将其转化为对象。

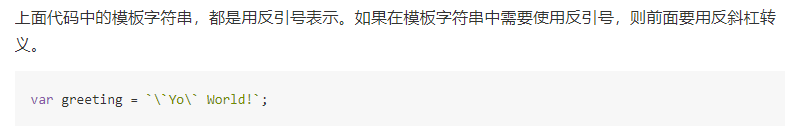


七．模板字符

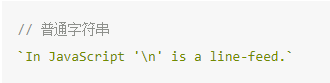
1>.嵌入变量需在${}之中🡪

$(变量) 主要区分$(“字符”) == “字符”

2>.里面嵌入特殊字符



`\和\`表示 ``符号.



/n换行

3>.模板字符默认都会保留代码的空格如果不需要可以使用trim()方法清除🡪

