对比于es5进行：

1. 变量声明const和let 区别于es5的var变量声明：

在ES6之前，我们都是用var关键字声明变量。无论声明在何处，都会被视为声明在**函数的最顶部**(不在函数内即在**全局作用域的最顶部**)。这就是函数**变量提升**例如:

function aa() {

if(flag) {

var test = 'hello man'

} else {

console.log(test)

}

}

以上的代码实际上是：

function aa() {

var test *// 变量提升，函数最顶部*

if(flag) {

test = 'hello man'

} else {

*//此处访问 test 值为 undefined*

console.log(test)

}

*//此处访问 test 值为 undefined*

}

接下来ES6主角登场：  
 我们通常用 let 和 const 来声明，let 表示**变量**、const 表示**常量**。let 和 const 都 是块级作用域。怎么理解这个块级作用域？说白了只要在**{}花括号内**的代码块即可以 为 let 和 const 的作用域。

* 在一个函数内部(let)
* 在一个代码块内部(const)





1. 字符串的拼接使用

1)Es5使用字符串拼接变量的时候需要用+连接 eg：var name=’123’ console.log(“hello”+name)

Es6可以将表达式嵌入字符串中进行拼接，使用${}来界定eg: console.log(`hello ${name}`)

2)Es5我们在拼接多行字符串时需要使用\反斜杠来完成 eg：

var msg = "Hi \

man!"

Es6我们直接使用``(叹号左边的点) eg：const template = `<div>

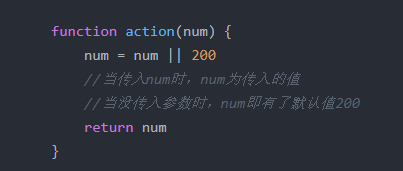
<span>hello world</span>

</div>`

3）Es6对于字符串也提供了很多方法

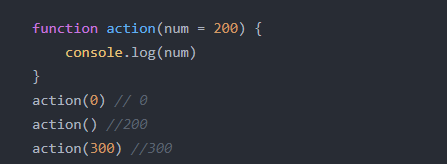


1. 函数
2. 函数默认参数赋值，ES5



当num传入值为0时为false，200就会赋值给num，与实际想象的结果不同

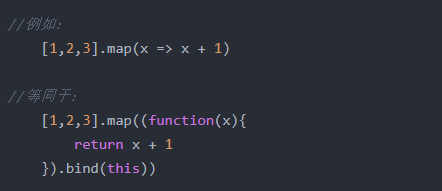
Es6中在函数定义的时候便可以初始化值



1. 箭头函数

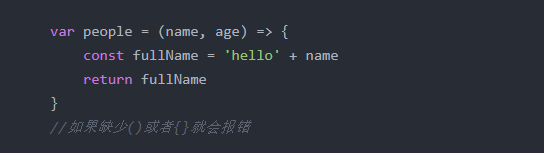
Es6提供了函数的快捷写法，箭头函数。

箭头函数的三个特点：1.不需要function关键字来创建函数2.省略return关键字3.继承当前上下文的this关键字

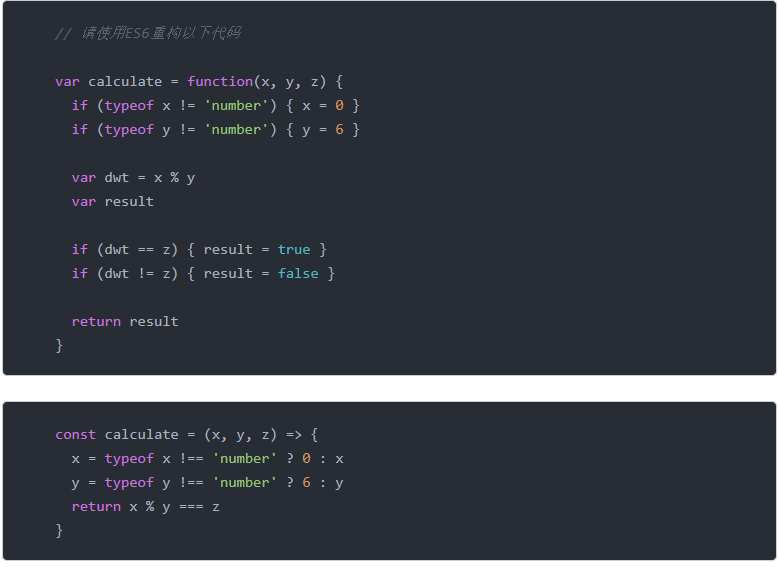


当你的函数**有且仅有**一个参数的时候，是可以省略掉括号的。当你函数返回**有且仅有**一个表达式的时候可以省略{} 和 return；例如:

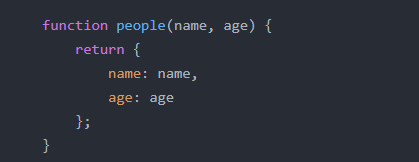
var people = name => `hello ${name}`



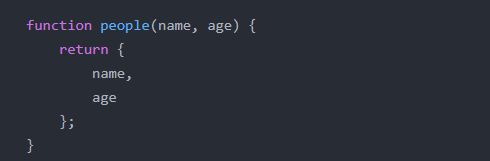
如果有两个参数的话就不可以省略了



1. 拓展的对象功能
2. 对象初始化的简写。Es5中我们对象在声明的时候都是用键值对的形式，有可能出现键值对重名



在Es6中可以简写为



1. 在给对象赋方法的时候Es6也做了一定的简化

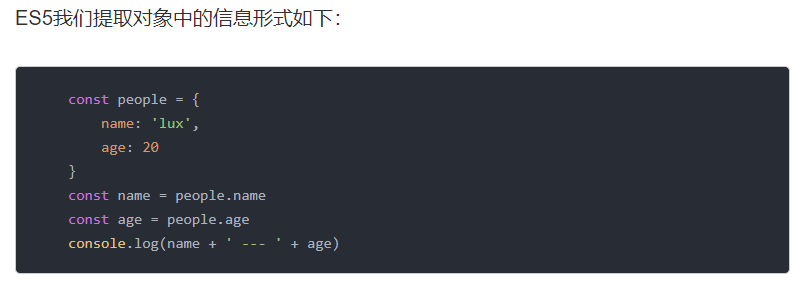


1. Es6中提供了对象的浅拷贝方法Object.assign

Object.assign() 可以把任意多个源对象自身可枚举的属性拷贝给目标对象，然后返回目标对象。第一参数即为目标对象。在实际项目中，我们为了不改变源对象。一般会把目标对象传为{}

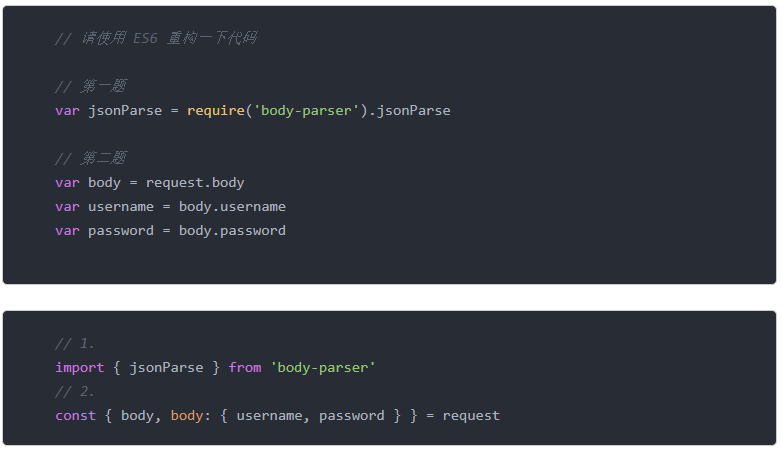
1. 更方便的数据访问结构

数组和对象是js中最常用的表达方式，es6中简化了信息的提取

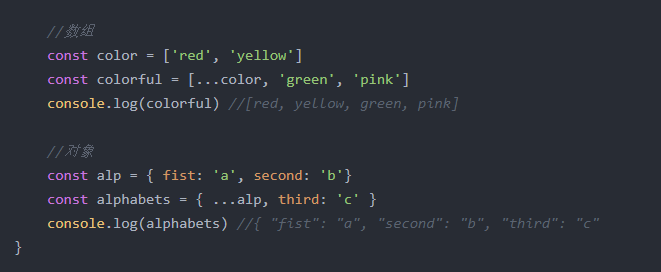


ES6中我们可以这样写：

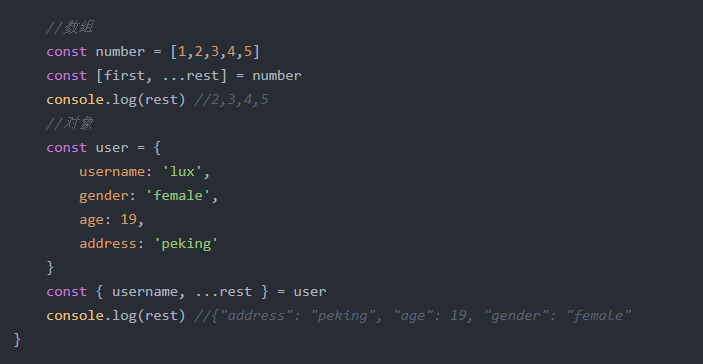


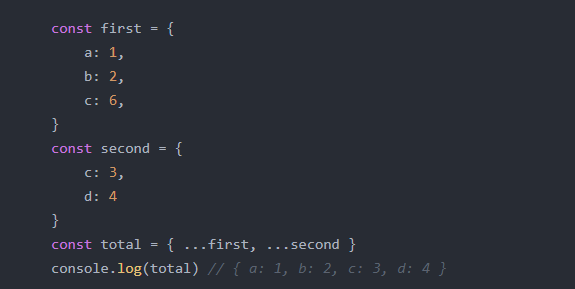


1. Spread Operator展开运算符（三个点 ...）

用来简单的组装对象或者数组

有时候我们想获取数组或者对象除了前几项或者除了某几项的其他项



对于 Object 而言，还可以用于组合成新的 Object 。

1. import 和 export



1. **promise（主要针对之前的异步操作代码量大）**

在promise之前代码过多的回调或者嵌套，可读性差、耦合度高、扩展性低。通过Promise机制，扁平化的代码机构，大大提高了代码可读性；用同步编程的方式来编写异步代码，保存线性的代码逻辑，极大的降低了代码耦合性而提高了程序的可扩展性。

Eg：发起异步请求：