# 基础类型

### 布尔值

let isDone: boolean = false/true

### 数字

let num: number = 6

支持二进制，八进制，十进制，十六进制

### 字符串

let name: string = ‘ Jone’ 可使用双引号或单引号

支持模板字符串，反引号`` 以$｛｝的形式和字符串拼接

### 数组

定义数组的两种方式：

元素类型后加[ ] : let list : number [ ] = [ 1,2,3,4 ] 元素只能为number类型

多种数据类型，（number | string）[ ]

数组泛型（Array<元素类型>）: let list: Array<number> = [ 1,2 ]

多种数据类型，Array<number | string >

### 元组 Tuple

元组类型允许表示一个已知元素数量和类型的数组，各数组的类型不必相同

定义一个值分别为string和number的元组，顺序不可变

let x : [ number, string ]

访问一个已知索引的元素，会得到元素的数据类型.

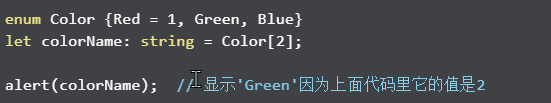
### 枚举

enum类型是对javascript标准数据类型的一个补充，使用枚举类型为一组数据赋予友好的名字

定义枚举后，使用对象的方式（对象.属性）给变量赋值为属性在枚举中的元素编号



其c的值为元素编号1



枚举类型可以根据枚举的值得到它的名字

### 任意值

在编程阶段还不清楚类型的变量，它的值可能来自动态的内容，这种情况下不希望类型检查器对这些值进行检查，直接让他们通过编译阶段的检查，可以使用any类型来标记这些变量

### 空值

void（空值）类型与any（任意值）类型相反，它表示没有任何类型。当一个函数没有返回值时，会见到其返回值类型是void。

声明一个void类型的变量没什么用，因为只能为它赋予undefined和null

### Null和Undefined

Typescript里，undefined和null两者各自有自己的类型分别叫undefinede和null。和void（空值）相似，本身的类型用处不大。默认情况下null和undefined是所有类型的子类型。

### Never

never类型表示那些用不存在的值的类型。例如：never类型是那些总是会抛出异常或根本就不会有返回值的函数表达式或箭头函数表达式的返回值的类型

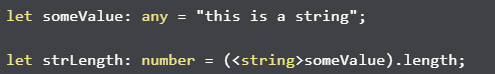
never类型是任何类型的子类型，也可以赋值给任何类型

### 类型断言

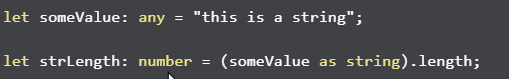
通过类型断言的方式告诉编译器知道自己在干么，告诉编译器：我已经进行了必须的检查

类型断言的两种方式：

1. 尖括号语法



1. as语法



两种方式是等价的，当在Typescript里使用JSX时，只有as语法断言是被允许的

# 变量声明

声明变量的类型和初始值

var 变量名：类型 = 值；

var 变量名：类型； 没有初始值，变量值会为undefined

var 变量名=值； 声明变量并有初始值，该变量可以为任意类型

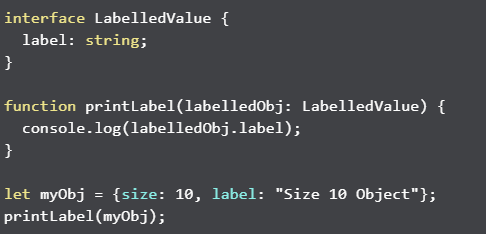
var 变量名； 声明变量没有设置类型和初始值，类型可以为任意类型，值undefined

# 接口（interface）

Typescript的核心原则之一是对值所具有的结构进行类型检查。接口的作用就是为这些类型命名和为你的代码或第三方代码定义契约（规范代码）

### 初识

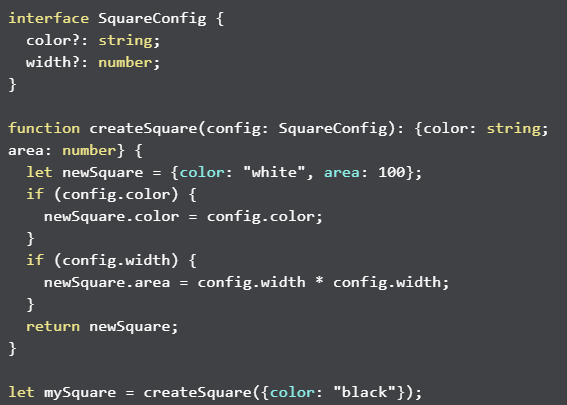
接口描述：必须包含一个label属性且类型为string

、

LavelledValue接口类似一个规则，要求有一个lable属性且类型为strign的对象

### 可选属性

接口里的属性不全都是必需的。有些只是在某些条件下存在，或者根本不存在。



带有可选属性的接口与普通的接口定义差不多，只是在可选属性名字定义的后面加一个?符号

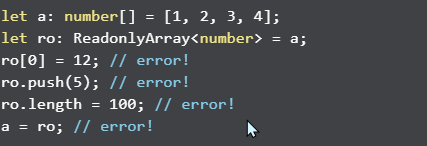
好处：

1. 对可能存在的属性进行预定义
2. 可以捕获引用了不存在的属性时的错误

### 只读属性

一些对象属性只能在对象刚刚创建的时候修改其值。可以再属性名前用readonly来指定只读属性

TypeScript具有ReadonlyArray<T>类型，它与Array<T>相似，只是把所有可变方法去掉了，确保数组创建后不能被修改



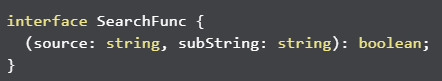
把整个ReadonlyArray赋值到一个普通数组是不可以的。

### readonly VS const

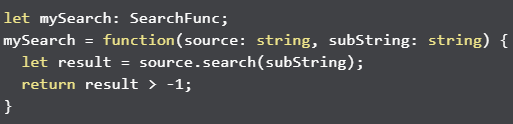
判断改用readonly还是const的方法。是看要把它作为变量使用还是属性使用。

### 函数类型

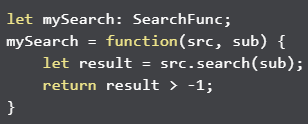
定义一个接口，它只有函数参数列表和返回值。参数列表中每个参数都需要名字和类型。



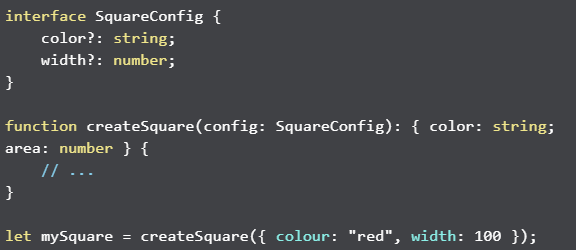
创建一个函数类型的变量



函数的参数会逐个进行检查，要求对应位置上的参数类型是兼容的。如果不想指定类型，TS的类型系统会推断参数类型，因为函数直接赋值给了SearchFunc类型变量。



# 额外的属性检查



传入createSquare的参数为colour而不是color。

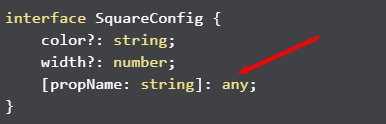
认为这个程序已经正确类型化了，因为width属性是兼容的，不存在color属性，而colour属性无意义，但是TypeScript会认为这段代码有bug。对象字面量会被特殊对待而且会经过额外属性检查，当将他们赋值给变量或作为参数传递时。如果一个对象字面量存在任何‘目标类型’不包含属性时，会报错。

绕过额外属性检查的方式

1. 类型断言



1. 添加一个字符串索引签名，前提是确定这个对象可能具有某些作为特殊用途使用的额外属性



1. 将对象赋值给另一个变量

