# typeScript

## 使用开发工具VSCode

获取开发工具网址：https://code.visualstudio.com/

## 2、开发工具部分快捷功能

### 1、快速生成html模板文件

第一行输入【!】然后按Tab键

## 3、学习typescript

typescript中文网：<https://www.tslang.cn/>

### 1、基础类型

#### 1、布尔值：boolean

最基本的数据类型就是简单的true/false值，在JavaScript和TypeScript里叫做boolean（其它语言中也一样）。



#### 2、数字：number

和JavaScript一样，TypeScript里的所有数字都是浮点数。 这些浮点数的类型是 number。 除了支持十进制和十六进制字面量，Typescript还支持ECMAScript 2015中引入的二进制和八进制字面量。



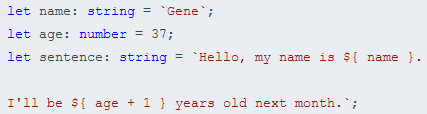


#### 3、字符串：string

JavaScript程序的另一项基本操作是处理网页或服务器端的文本数据。 像其它语言里一样，我们使用string表示文本数据类型。 和JavaScript一样，可以使用双引号（ "）或单引号（'）表示字符串。



你还可以使用模版字符串，它可以定义多行文本和内嵌表达式。 这种字符串是被反引号包围（ `），并且以${ expr }这种形式嵌入表达式



这与下面定义sentence的方式效果相同：



#### 4、数组

TypeScript像JavaScript一样可以操作数组元素。 有两种方式可以定义数组。 第一种，可以在元素类型后面接上 []，表示由此类型元素组成的一个数组：

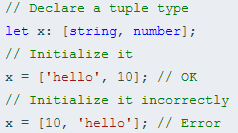


第二种方式是使用数组泛型，Array<元素类型>：



#### 5、元组：Tuple [tʌpl]

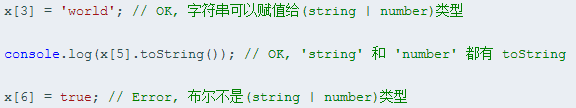
元组类型允许表示一个已知元素数量和类型的数组，各元素的类型不必相同。 比如，你可以定义一对值分别为 string和number类型的元组。



当访问一个已知索引的元素，会得到正确的类型：



当访问一个越界的元素，会使用联合类型替代：



联合类型是高级主题，我们会在以后的章节里讨论它。

#### 6、枚举：enum

enum类型是对JavaScript标准数据类型的一个补充。 像C#等其它语言一样，使用枚举类型可以为一组数值赋予友好的名字。



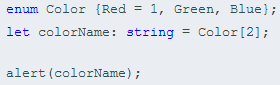
默认情况下，从0开始为元素编号。 你也可以手动的指定成员的数值。 例如，我们将上面的例子改成从 1开始编号：



或者，全部都采用手动赋值：

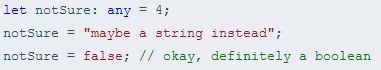


枚举类型提供的一个便利是你可以由枚举的值得到它的名字。 例如，我们知道数值为2，但是不确定它映射到Color里的哪个名字，我们可以查找相应的名字：

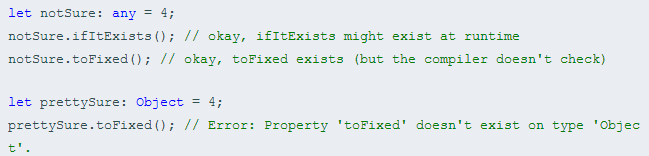


#### 7、任意值：any

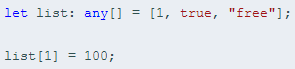
有时候，我们会想要为那些在编程阶段还不清楚类型的变量指定一个类型。 这些值可能来自于动态的内容，比如来自用户输入或第三方代码库。 这种情况下，我们不希望类型检查器对这些值进行检查而是直接让它们通过编译阶段的检查。 那么我们可以使用 any类型来标记这些变量：



在对现有代码进行改写的时候，any类型是十分有用的，它允许你在编译时可选择地包含或移除类型检查。 你可能认为 Object有相似的作用，就像它在其它语言中那样。 但是 Object类型的变量只是允许你给它赋任意值 - 但是却不能够在它上面调用任意的方法，即便它真的有这些方法：

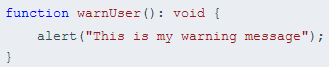


当你只知道一部分数据的类型时，any类型也是有用的。 比如，你有一个数组，它包含了不同的类型的数据：



#### 8、空值：void

某种程度上来说，void类型像是与any类型相反，它表示没有任何类型。 当一个函数没有返回值时，你通常会见到其返回值类型是 void：

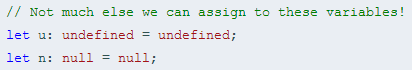


声明一个void类型的变量没有什么大用，因为你只能为它赋予undefined和null：



#### 9、Null和Undefined

TypeScript里，undefined和null两者各自有自己的类型分别叫做undefined和null。 和 void相似，它们的本身的类型用处不是很大：



默认情况下null和undefined是所有类型的子类型。 就是说你可以把 null和undefined赋值给number类型的变量。

然而，当你指定了--strictNullChecks标记，null和undefined只能赋值给void和它们各自。 这能避免 很多常见的问题。 也许在某处你想传入一个 string或null或undefined，你可以使用联合类型string | null | undefined。 再次说明，稍后我们会介绍联合类型。

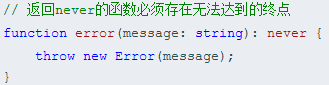
注意：我们鼓励尽可能地使用--strictNullChecks，但在本手册里我们假设这个标记是关闭的。

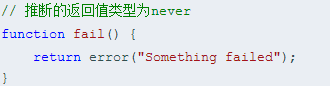
#### 10、Never

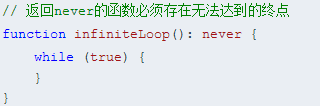
never类型表示的是那些永不存在的值的类型。 例如， never类型是那些总是会抛出异常或根本就不会有返回值的函数表达式或箭头函数表达式的返回值类型； 变量也可能是 never类型，当它们被永不为真的类型保护所约束时。

never类型是任何类型的子类型，也可以赋值给任何类型；然而，没有类型是never的子类型或可以赋值给never类型（除了never本身之外）。 即使 any也不可以赋值给never。

下面是一些返回never类型的函数：





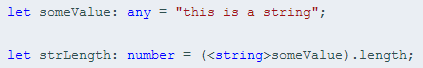


#### 11、类型断言

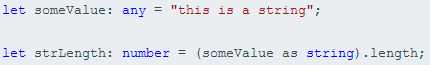
有时候你会遇到这样的情况，你会比TypeScript更了解某个值的详细信息。 通常这会发生在你清楚地知道一个实体具有比它现有类型更确切的类型。

通过类型断言这种方式可以告诉编译器，“相信我，我知道自己在干什么”。 类型断言好比其它语言里的类型转换，但是不进行特殊的数据检查和解构。 它没有运行时的影响，只是在编译阶段起作用。 TypeScript会假设你，程序员，已经进行了必须的检查。

类型断言有两种形式。 其一是“尖括号”语法：



另一个为as语法：



两种形式是等价的。 至于使用哪个大多数情况下是凭个人喜好；然而，当你在TypeScript里使用JSX时，只有as语法断言是被允许的。

### 2、变量声明let和const

let在很多方面与var是相似的，但是可以帮助大家避免在JavaScript里常见一些问题。

const是对let的一个增强，它能阻止对一个变量再次赋值。

const实际上保证的，并不是变量的值不得改动，而是变量指向的那个内存地址不得改动。例如：

const foo = {};

// 为 foo 添加一个属性，可以成功

foo.prop = 123;

foo.prop // 123

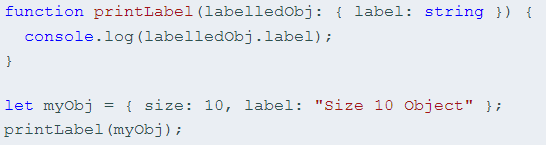
// 将 foo 指向另一个对象，就会报错

foo = {}; // TypeError: "foo" is read-only

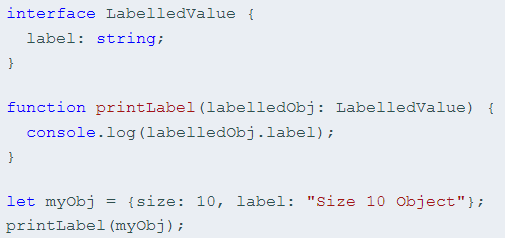
### 3、接口

TypeScript的核心原则之一是对值所具有的shape进行类型检查。 它有时被称做“鸭式辨型法”或“结构性子类型化”。 在TypeScript里，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。

下面通过一个简单示例来观察接口是如何工作的：



类型检查器会查看printLabel的调用。 printLabel有一个参数，并要求这个对象参数有一个名为label类型为string的属性。 需要注意的是，我们传入的对象参数实际上会包含很多属性，但是编译器只会检查那些必需的属性是否存在，并且其类型是否匹配。 然而，有些时候TypeScript却并不会这么宽松，我们下面会稍做讲解。



LabelledValue接口就好比一个名字，用来描述上面例子里的要求。 它代表了有一个 label属性且类型为string的对象。 需要注意的是，我们在这里并不能像在其它语言里一样，说传给 printLabel的对象实现了这个接口。我们只会去关注值的外形。 只要传入的对象满足上面提到的必要条件，那么它就是被允许的。

### 4、类

传统的JavaScript程序使用函数和基于原型的继承来创建可重用的组件，但对于熟悉使用面向对象方式的程序员来讲就有些棘手，因为他们用的是基于类的继承并且对象是由类构建出来的。 从ECMAScript 2015，也就是ECMAScript 6开始，JavaScript程序员将能够使用基于类的面向对象的方式。 使用TypeScript，我们允许开发者现在就使用这些特性，并且编译后的JavaScript可以在所有主流浏览器和平台上运行，而不需要等到下个JavaScript版本。

### 5、函数

函数是JavaScript应用程序的基础。 它帮助你实现抽象层，模拟类，信息隐藏和模块。 在TypeScript里，虽然已经支持类，命名空间和模块，但函数仍然是主要的定义 行为的地方。 TypeScript为JavaScript函数添加了额外的功能，让我们可以更容易地使用。

### 6、泛型

软件工程中，我们不仅要创建一致的定义良好的API，同时也要考虑可重用性。 组件不仅能够支持当前的数据类型，同时也能支持未来的数据类型，这在创建大型系统时为你提供了十分灵活的功能。

在像C#和Java这样的语言中，可以使用泛型来创建可重用的组件，一个组件可以支持多种类型的数据。 这样用户就可以以自己的数据类型来使用组件。

### 7、枚举

使用枚举我们可以定义一些有名字的数字常量。 枚举通过 enum关键字来定义。

### 8、类型推断

这节介绍TypeScript里的类型推论。即，类型是在哪里如何被推断的。

### 9、类型兼容性

TypeScript里的类型兼容性是基于结构子类型的。 结构类型是一种只使用其成员来描述类型的方式。 它正好与名义（nominal）类型形成对比。（译者注：在基于名义类型的类型系统中，数据类型的兼容性或等价性是通过明确的声明和/或类型的名称来决定的。这与结构性类型系统不同，它是基于类型的组成结构，且不要求明确地声明。）

## 4、Promise对象

所谓Promise，简单说就是一个容器，里面保存着某个未来才会结束的事件（通常是一个异步操作）的结果。从语法上说，Promise 是一个对象，从它可以获取异步操作的消息。

特点：

（1）对象的状态不受外界影响。Promise对象代表一个异步操作，有三种状态：Pending（进行中）、Resolved（已完成，又称 Fulfilled）和Rejected（已失败）。只有异步操作的结果，可以决定当前是哪一种状态，任何其他操作都无法改变这个状态。

（2）一旦状态改变，就不会再变，任何时候都可以得到这个结果。Promise对象的状态改变，只有两种可能：从Pending变为Resolved和从Pending变为Rejected。只要这两种情况发生，状态就凝固了，不会再变了，会一直保持这个结果。如果改变已经发生了，你再对Promise对象添加回调函数，也会立即得到这个结果。这与事件（Event）完全不同，事件的特点是，如果你错过了它，再去监听，是得不到结果的。