类型别名

关键字type可创建类型别名，用来给类型起一个新名字，常用于联合类型

字符串字面量类型2

约束取值只能是某几个字符串中的一个，**类型别名与字符串字面量类型都是使用** type **进行定义。**

元组

数组合并了相同类型的对象，而元组（Tuple）合并了不同类型的对象。

元组起源于函数编程语言（如 F#），这些语言中会频繁使用元组。

let tom: [string, number] = ['Tom', 25];

当赋值或访问一个已知索引的元素时，会得到正确的类型。

当直接对元组类型的变量进行初始化或者赋值的时候，需要提供所有元组类型中指定的项。

当添加越界的元素时，它的类型会被限制为元组中每个类型的联合类型

枚举

枚举使用 enum 关键字来定义

枚举成员会被赋值为从 0 开始递增的数字，同时也会对枚举值到枚举名进行反向映射

也可以给枚举项手动赋值，如果未手动赋值的枚举项与手动赋值的重复了，前面的值会被后面的覆盖

手动赋值的枚举项可以不是数字，此时需要使用类型断言来让 tsc 无视类型检查

手动赋值的枚举项也可以为小数或负数，此时后续未手动赋值的项的递增步长仍为 1

枚举项有两种类型：常数项（constant member）和计算所得项（computed member）

类

- **类(Class)：定义了一件事物的抽象特点，包含它的属性和方法**
- 对象（Object）：类的实例，通过 new 生成
- 面向对象（OOP）的三大特性：封装、继承、多态
- 封装（Encapsulation）：将对数据的操作细节隐藏起来，只暴露对外的接口。外界调用端不需要（也不可能）知道细节，就能通过对外提供的接口来访问该对象，同时也保证了外界无法任意更改对象内部的数据
- 继承（Inheritance）：子类继承父类，子类除了拥有父类的所有特性外，还有一些更具体的特性
- 多态（Polymorphism）：由继承而产生了相关的不同的类，对同一个方法可以有不同的响应。比如 Cat 和 Dog 都继承自 Animal，但是分别实现了自己的 eat 方

法。此时针对某一个实例，我们无需了解它是 Cat 还是 Dog，就可以直接调用 eat 方法，程序会自动判断出来应该如何执行 eat

- 存取器（getter & setter）：用以改变属性的读取和赋值行为
- 修饰符（Modifiers）：修饰符是一些关键字，用于限定成员或类型的性质。比如 public 表示公有属性或方法
- 抽象类（Abstract Class）：抽象类是供其他类继承的基类，抽象类不允许被实例化。抽象类中的抽象方法必须在子类中被实现

接口（Interfaces）

- 不同类之间公有的属性或方法，可以抽象成一个接口
- 接口可以被类实现（implements）
- 一个类只能继承自另一个类，但是可以实现多个接口

public private 和 protected：TypeScript 可以使用三种访问修饰符（Access Modifiers）
分别是 public、private 和 protected。

- public 修饰的属性或方法是公有的，可以在任何地方被访问到，默认所有的属性和方法都是 public
- private 修饰的属性或方法是私有的，不能在声明它的类的外部访问
- protected 修饰的属性或方法是受保护的，它和 private 类似，区别是它在子类中也是允许被访问的

- **readonly 只读属性关键字，只允许出现在属性声明或索引签名中。**
- **abstract 用于定义抽象类和其中的抽象方法 abstract 用于定义抽象类和其中的抽象方法**

类的类型：ts的类也可以使用指定的类进行数据类型检测

类与接口
实现（implements）是面向对象中的一个重要概念。一般来讲，一个类只能继承自另一个类，有时候不同类之间可以有一些共有的特性，这时候就可以把特性提取成接口（interfaces），用 implements 关键字来实现。这个特性大大提高了面向对象的灵活性。

- 一个类可以实现多个接口
- 接口与接口之间可以是继承关系，接口也可以继承类

- 混合类型：可以使用接口的方式来定义一个函数需要符合的形状，一个函数还可以有自己的属性和方法

泛型

泛型（Generics）是指在定义函数、接口或类的时候，不预先指定具体的类型，而在使用的时候再指定类型的一种特性。

- 泛型的时候，可以一次定义多个类型参数
- 泛型约束：在函数内部使用泛型变量的时候，由于事先不知道它是哪种类型，所以不能随意的操作它的属性或方法
- 泛型可以继承泛型
- 接口可以一个函数的形状，接口里面也可以使用泛型定义函数的形状
- 泛型类：泛型也可以用于类的类型定义中泛型也可以用于类的类型定义中
- 泛型参数默认类型：当泛型时没有指定类型参数、且实际值参数中也无法推测出时，触发默认类型（TypeScript 2.3+）

声明合并

- ~~函数合并：重载可以定义多个函数参数的类型（什么乱七八糟的，自己写的时候报错）~~
- 接口合并：接口中的属性在合并时会简单的合并到一个接口中，**合并的属性的类型必须是唯一的**
- 类合并：类的合并与接口的合并规则一致