# **第五单元 为函数和函数参数定义类型**

**知识回顾**

1. 索引类型

使用接口描述索引的类型和通过索引得到的值的类型,也可以给索引设置readonly,防止索引被更改

2. 继承类型

接口可以继承,这和类一样,这提高了接口的可复用性

一个接口可以被多个接口继承,同样,一个接口也可以继承多个接口,多个接口用逗号隔开

3. 混合接口

js的类型是灵活的,在js中,函数是对象类型,对象可以有属性,所以有时我们的一个对象既是函数,也包含一些属性

*在3.1版本中,ts直接支持给对象添加属性*

1. **函数类型**
2. **为函数定义类型**：我们可以给函数定义类型，这个定义包括对参数和返回值的类型定义。**注**：如果在这里省略参数的类型，TypeScript 会默认这个参数是 any 类型；如果省略返回值的类型，如果函数无返回值，那么 TypeScript 会默认函数返回值是 void 类型；如果函数有返回值，那么 TypeScript 会根据我们定义的逻辑推断出返回类型。

(2)**完整的函数类型**：一个函数的定义包括函数名、参数、逻辑和返回值。我们为一个函数定义类型时，完整的定义应该包括**参数类型**和**返回值类型**。函数中如果使用了函数体之外定义的变量，这个变量的类型是不体现在函数类型定义的。

(3)**使用接口定义函数类型：**使用接口可以清晰地定义函数类型。

(4)**使用类型别名：**可以使用类型别名来定义函数类型。

注：使用type关键字可以为原始值、联合类型、元组以及任何我们定义的类型起一个别名。

**优点**：使用类型别名定义函数类型更直观易读。

1. **参数**
2. **可选参数**：TypeScript 会帮我们在编写代码的时候就检查出调用函数时参数中存在的一些错误。

可选参数需要在参数名后跟随一个？即可。

**参数位置的要求：**接口形式定义的函数类型必选参数和可选参数的位置前后是无所谓的，但是今天学到的定义形式，可选参数必须放在必选参数后面，这和在 JS 中定义函数是一致的。

TypeScript与JavaScript

*在TypeScript中，可选参数放到最后才行。*

*在 JavaScript 中，其实是没有可选参数这个概念的，只不过是我们在写逻辑的时候，我们可能会判断某个参数是否为undefined，如果是则说明调用该函数的时候没有传这个参数，要做下兼容处理。*

1. **默认参数**：可选参数和带默认值的参数在函数调用时都是可以不传实参的，但是区别在于定义函数的时候，可选参数必须放在必选参数后面，而带默认值的参数则可放在必须参数前后都可。
2. **剩余参数**：**①**在 JS 中，如果我们定义一个函数，这个函数可以输入任意个数的参数，那么我们就无法在定义参数列表的时候挨个定义。**②**在 ES6 发布之前，我们需要用到 arguments 来获取参数列表。arguments 是每一个函数都包含的一个**类数组**对象，它包含在函数调用时传入函数的所有实际参数（简称实参），它还包含一个 length 属性，记录参数个数。**③**在 ES6 中，加入了"…"拓展运算符，它可以将一个函数或对象进行拆解。
3. **函数重载，此重载VS彼重载**

**定义：**函数重载是指定义几个函数名相同，但参数个数或类型不同的函数，在调用时传入不同的参数，编译器会自动调用适合的函数。

JavaScript 作为一个动态语言是没有函数重载的，只能我们自己在函数体内通过判断参数的个数、类型来指定不同的处理逻辑。

在 TypeScript 中有函数重载的概念，但并不是定义几个同名实体函数，然后根据不同的参数个数或类型来自动调用相应的函数。

TypeScript的函数重载是在类型系统层面的，是为了更好地进行类型推断。TypeScript的函数重载通过为一个函数指定多个函数类型定义，从而对函数调用的返回值进行检查。

Type和Interface的区别

1. 定义不同

①Interface：接口，TS设计出来主要用于定义【对象类型】，可以对【对象】的形状进行描述。

②Type：类型别名，为类型创建一个新名称；但其不是一个类型，只是一个别名。

**作用：**类型别名可以起到类似接口的作用。但是，有一些细微的差别。主要区别在于 type 一旦定义就不能再添加新的属性，而 interface 总是可扩展的。

【**type声明值类型，做类型判断的；interface只能声明对象，不能声明基础类型，所以校验的函数参数只能是对象**】

**异同点：**

1. type是赋值，进行别名的修改；
2. Interface可以声明一个类型；
3. Type可以声明基础类型数据，interface只能声明对象；
4. Interface可以使用extends进行继承，但是type不可以，因为类型别名属于赋值的一种；
5. Interface可以重复声明，自动合并一起；
6. Type不可以重复声明。

**相同点：**

1. 都可以描述一个对象或者函数
2. 都允许继承（官网话叫拓展extends）：interface 和 type 都可以拓展，并且两者并不是相互独立的，也就是说 interface 可以 extends type, type 也可以 extends interface 。 虽然效果差不多，但是两者语法不同。

**type的使用：**可以定义基本类型，联合类型，元组类型

1. 基本类型：type StringType = string
2. 联合类型：type paramtype = number | string
3. 元组类型：type arrType = [string, string, number]
4. 定义函数类型--类型别名：

type AddFunc = {name: string, age: number}

### **总结**

1.如果需要被extends或者implements, 则尽量使用接口。

2.如果需要使用联合类型或者元组类型，类型别名会更合适。

3.如果是定义对象或函数，则都可以。

4.如果实在不想选择的话，就能用interface实现，用interface，如果不能就用type。