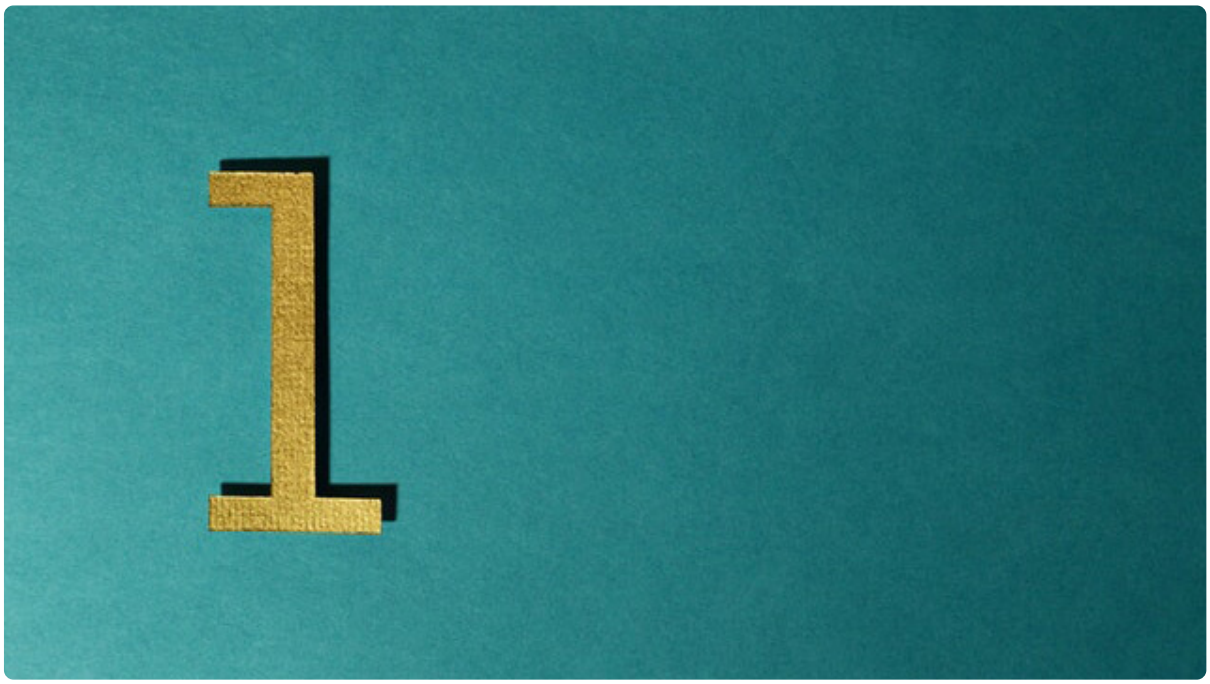
18 类型别名和字面量类型—单调的类型

更新时间：2019-06-25 17:21:46



> 学习从来无捷径，循序渐进登高峰。
>
> —— 高永祚

本小节我们来学习类型别名和字面量类型。类型别名我们之前在讲泛型的时候接触过，现在来详细学习下。

3.5.1 类型别名

类型别名就是给一种类型起个别的名字，之后只要使用这个类型的地方，都可以用这个名字作为类型代替，但是它只是起了一个名字，并不是创建了一个新类型。这种感觉就像 JS 中对象的赋值，你可以把一个对象赋给一个变量，使用这个对象的地方都可以用这个变量代替，但你并不是创建了一个新对象，而是通过引用来使用这个对象。

我们来看下怎么定义类型别名，使用 `type` 关键字：

```
type TypeString = string;
let str: TypeString;
str = 123; // error Type '123' is not assignable to type 'string'
```

类型别名也可以使用泛型，来看例子：

```
type PositionType<T> = { x: T; y: T };
const position1: PositionType<number> = {
  x: 1,
  y: -1
};
const position2: PositionType<string> = {
  x: "right",
  y: "top"
};
```

使用类型别名时也可以在属性中引用自己：

```
type Child<T> = {
  current: T;
  child?: Child<T>;
};
let ccc: Child<string> = {
  current: "first",
  child: {
    // error
    current: "second",
    child: {
      current: "third",
      child: "test" // 这个地方不符合type，造成最外层child处报错
    }
  }
};
```

但是要注意，只可以在对象属性中引用类型别名自己，不能直接使用，比如下面这样是不对的：

```
type Child = Child[]; // error 类型别名“Child”循环引用自身
```

另外要注意，因为类型别名只是为其它类型起了个新名字来引用这个类型，所以当它为接口起别名时，不能使用 `extends` 和 `implements` 。

接口和类型别名有时可以起到同样作用，比如下面这个例子：

```
type Alias = {
  num: number;
};
interface Interface {
  num: number;
}
let _alias: Alias = {
  num: 123
};
let _interface: Interface = {
  num: 321
};
_alias = _interface;
```

可以看到用类型别名和接口都可以定义一个只包含 num 属性的对象类型，而且类型是兼容的。那么什么时候用类型别名，什么时候用接口呢？可以通过两点来选择：

- 当你定义的类型要用于拓展，即使用 implements 等修饰符时，用接口。
- 当无法通过接口，并且需要使用联合类型或元组类型，用类型别名。

3.5.2. 字面量类型

字面量类型其实比较基础，但是它又不适合放到基本类型里讲，因为字符串字面量类型和字符串类型其实并不一样，所以接下来我们来学习两种字面量类型。

(1) 字符串字面量类型

字符串字面量类型其实就是字符串常量，与字符串类型不同的是它是具体的值。

```
type Name = "Lison";
const name1: Name = "test"; // error 不能将类型“"test"”分配给类型“"Lison"”
const name2: Name = "Lison";
```

你还可以使用联合类型来使用多个字符串：

```ts
type Direction = "north" | "east" | "south" | "west";
function getDirectionFirstLetter(direction: Direction) {
  return direction.substr(0, 1);
}
getDirectionFirstLetter("test"); // error 类型“"test"”的参数不能赋给类型“Direction”的参数
getDirectionFirstLetter("east");
```

(2) 数字字面量类型

另一个字面量类型就是数字字面量类型，它和字符串字面量类型差不多，都是指定类型为具体的值。

```ts
type Age = 18;
interface Info {
  name: string;
  age: Age;
}
const info: Info = {
  name: "Lison",
  age: 28 // error 不能将类型“28”分配给类型“18”
};
```

这里补充一个比较经典的逻辑错误，来看例子：

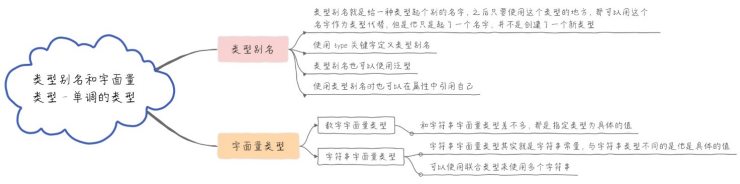
```ts
function getValue(index: number) {
  if (index !== 0 || index !== 1) {
    // error This condition will always return 'true' since the types '0' and '1' have no overlap
    // ...
  }
}
```

这个例子中，在判断逻辑处使用了 `||` 符，当 index !== 0 不成立时，说明 index 就是 0，则不应该再判断 index 是否不等于 1；而如果 index !== 0 成立，那后面的判断也不会再执行；所以这个地方会报错。

本节小结

本小节我们学习了类型别名和字面量类型，类型别名就是给一个类型起个别名，以后我们可以使用类型别名将较为复杂的类型抽离出来，这样任何需要使用这个类型的地方都可以使用这个别名代替；使用类型别名的好处有时和使用变量一样，我们可以将复杂的逻辑判断语句赋给一个变量，然后再进行判断，只需要判断这个变量的true或false即可；我们使用类型别名也可以起到简化代码的作用。我们还学习了两种字面量类型：数字字面量类型和字符串字面量类型，它们都是使用具体的字面量值来作为一种类型，所以我们叫它单调类型。

下个小节我们将学习可辨识联合类型，我们可以使用可辨识联合并保证每个case都被处理。



17 使用显式复制断言给TS一个你一定会赋值的承诺

19 使用可辨识联合并保证每个case都被处理

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论