14 类型推论，看TS有多懂你

更新时间：2019-06-17 19:41:31

> 构成我们学习最大障碍的是已知的东西，而不是未知的东西。
>
> —— 贝尔纳

在学习基础部分的章节时，我们讲过，在一些定义中如果你没有明确指定类型，编译器会自动推断出适合的类型；比如下面的这个简单例子：

```
let name = "lison";
name = 123; // error 不能将类型“123”分配给类型“string”
```

我们看到，在定义变量 name 的时候我们并没有指定 name 的类型，而是直接给它赋一个字符串。当我们再给 name 赋一个数值的时候，就会报错。在这里，TypeScript 根据我们赋给 name 的值的类型，推断出我们的 name 的类型，这里是 string 类型，当我们再给 string 类型的 name 赋其他类型值的时候就会报错。

这个是最基本的类型推论，根据右侧的值推断左侧变量的类型，接下来我们看两个更复杂的推论。

3.1.1 多类型联合

当我们定义一个数组或元组这种包含多个元素的值的时候，多个元素可以有不同的类型，这种时候 TypeScript 会将多个类型合并起来，组成一个联合类型，来看例子：

```
let arr = [1, "a"];
arr = ["b", 2, false]; // error 不能将类型“false”分配给类型“string | number”
```

可以看到，此时的 arr 的元素被推断为`string | number`，也就是元素可以是 string 类型也可以是 number 类型，除此两种类型外的类型是不可以的。再来看个例子：

```
let value = Math.random() * 10 > 5 ? 'abc' : 123
value = false // error 不能将类型“false”分配给类型“string | number”
```

这里我们给value赋值为一个三元操作符表达式，`Math.random() * 10`的值为0-10的随机数。这里判断，如果这个随机值大于5，则赋给value的值为字符串'abc'，否则为数值123，所以最后编译器推断出的类型为联合类型`string | number`，当给它再赋值为false的时候就会报错。

3.1.2 上下文类型

我们上面讲的两个例子都是根据`=`符号右边值的类型，推断左侧值的类型。现在要讲的上下文类型则相反，它是根据左侧的类型推断右侧的一些类型，先来看例子：

```
window.onmousedown = function(mouseEvent) {
  console.log(mouseEvent.a); // error 类型“MouseEvent”上不存在属性“a”
};
```

我们可以看到，表达式左侧是 window.onmousedown(鼠标按下时发生事件)，因此 TypeScript 会推断赋值表达式右侧函数的参数是事件对象，因为左侧是 mousedown 事件，所以 TypeScript 推断 mouseEvent 的类型是 MouseEvent。在回调函数中使用 mouseEvent 的时候，你可以访问鼠标事件对象的所有属性和方法，当访问不存在属性的时候，就会报错。

以上便是我要讲的三种常见的类型推论。在我们日常开发中，必写的类型还是要明确指定的，这样我们才能更准确地得到类型信息和开发辅助。

本节小结

本小节我们学习了TypeScript编译器进行类型推断的论据，其中有两种是由右推左的，也就是在赋值时根据右侧要赋的具体值，推断左侧要赋值的目标的类型，包括基本推论和多类型联合推论。基础推论是最基础的推论，多类型联合推论是根据数组、代码逻辑等，推断出多个符合的类型，然后组成联合类型的推论。还有一种由左推右的推论，我们是通过给元素绑定事件来讲解的，根据左侧要赋值的目标，来推断出右侧要赋的值中的一些类型信息。

下个小节我们将学习类型兼容性，我们知道JavaScript是灵活的，所以TypeScript通过类型兼容性来满足它的灵活特点，下个小节我们将介绍多种情况的兼容性表现。

13 TS中的类，小心它与ES标准的差异

15 类型兼容性，开放心态满足灵活的JS

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