08 使用类型断言达到预期

更新时间：2019-06-12 16:37:05

> 书是人类进步的阶梯。
>
> ——高尔基

学完前面的小节，你已经学习完了TypeScript的基本类型。从本小节开始，你将开始接触逻辑。在这之前，先来学习一个概念：*类型断言*。

虽然 TypeScript 很强大，但有时它还是不如我们了解一个值的类型，这时候我们更希望 TypeScript 不要帮我们进行类型检查，而是交给我们自己来，所以就用到了类型断言。类型断言有点像是一种类型转换，它把某个值强行指定为特定类型，我们先看个例子：

```
const getLength = target => {
  if (target.length) {
    return target.length;
  } else {
    return target.toString().length;
  }
};
```

这个函数能够接收一个参数，并返回它的长度，我们可以传入字符串、数组或数值等类型的值。如果有 length 属性，说明参数是数组或字符串类型，如果是数值类型是没有 length 属性的，所以需要把数值类型转为字符串然后再获取 length 值。现在我们限定传入的值只能是字符串或数值类型的值：

```
const getLength = (target: string | number): number => {
  if (target.length) { // error 报错信息看下方
    return target.length; // error 报错信息看下方
  } else {
    return target.toString().length;
  }
};
```

> 当 TypeScript 不确定一个联合类型的变量到底是哪个类型的时候，我们只能访问此联合类型的所有类型里共有的属性或方法，所以现在加了对参数 target 和返回值的类型定义之后就会报错：

```
// 类型"string | number"上不存在属性"length"
// 类型"number"上不存在属性"length"
```

很显然，我们是要做判断的，我们判断如果 target.length 不为 undefined， 说明它是有 length 属性的，但我们的参数是 string | number 联合类型，所以在我们开始做判断的时候就会报错。这个时候就要用类型断言，将 tagrget 的类型断言成 string 类型。它有两种写法，一种是 <type>value，一种是 value as type，下面例子中我们用两种形式都写出来：

```
const getStrLength = (target: string | number): number => {
  if ((<string>target).length) { // 这种形式在JSX代码中不可以使用，而且也是TSLint不建议的写法
    return (target as string).length; // 这种形式是没有任何问题的写法，所以建议大家始终使用这种形式
  } else {
    return target.toString().length;
  }
};
```

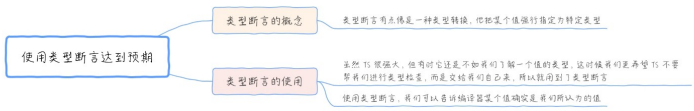
例子的函数体用到了三次 target，前两次都是访问了 target.length 属性，所以都要用类型断言来表明这个地方是 string 类型；而最后的 target 调用了 toString方法，因为 number 和 string 类型的值都有 toString 方法，所以没有报错。

这样虽然没问题了，但是每一处不同值会有不同情况的地方都需要用类型断言，后面讲到**高级类型**的时候会讲如何使用**自定义类型保护**来简化这里。

> 注意了，这两种写法都可以，但是 tslint 推荐使用 as 关键字，而且在 JSX 中只能使用 as 这种写法。

小结

本小节我们学习了类型断言的使用。使用类型断言，我们可以告诉编译器某个值确实是我们所认为的值，从而让编译器进行正确的类型推断，让类型检查符合我们的预期。下个小节我们将学习接口，学习了接口后，我们就可以定义几乎所有的数据结构了。



07 深入学习枚举

09 使用接口定义几乎任意结构