第十五课 泛型基础

学习目录

- **泛型基础介绍**
- **泛型基础用法**

一．泛型基础介绍

我们之前学习的知识中类型都是我们在程序运行之前指定好的类型，这么做的好处是一颗构建出统一的 API 类型约定，不过如果从程序的灵活性来说，我们其实还想让我们的函数、类等组件更加灵活。那这就需要我们的类型约定既要能够满足当前已定义好的数据类型要求，也可以满足未来在程序运行时的数据类型要求，泛型就是来完成这个任务的。

简单来说，我们定义了一个函数，这个函数参数、返回值等可以支持多种数据类型，用户可以以自己的需要传递符合自己数据类型要求的参数与返回值来使用函数。

定义一个普通函数，这个函数的参数和返回值类型相同，这里是数字类型。

```
function fn1(a: number): number {
    return a;
}
```

假如允许用户自定义任意类型，我们能想到的是函数参数和返回值是 any 类型。

```
function fn1(a: any): any{
    return a;
}
```

这个函数确实可以做到用户根据需要来传递符合自己数据类型要求的参数与返回值数据，不过这个函数的数据类型就不受控了，传入的参数类型与返回的类型没办法是相同的，我们只知道任何类型的值都有可能被返回。但是我们的目标是假如我们传入一个数字类型，我们希望返回的类型也是数字类型。

因此我们可以使用泛型来完成，在泛型中有一个新的概念类型变量，这个变量不是具体的值，而是类型。

```
function fn1<T>(a: T): T {
    return a;
}
```

我们给 fn1 添加了类型变量 T，一般建议这样命名， T 类型帮助我们捕获用户传入的类型，比如用户传入 number 类型，因此我们就可以使用这个类型。另外我们还可以使用 T 当做返回值类型，这允许我们跟踪函数里使用的类型的信息，现在我们可以知道参数类型与返回值类型是相同的了。

```
let f1 = fn1<string>("xingxingclassroom");
let f2 = fn1(100);
```

我们可以调用泛型函数，传入不同的数据类型进去，也会得到相应的返回值类型，第一种调用是通过尖括号语法显示的指定类型变量，第二种是利用 ts 的类型推论来使用泛型。

我们把 fn1 函数叫做泛型，也可以称之为泛型函数，这个泛型可以适用于多个类型，泛型的好处是它不会丢失类型信息，既可以像第一个 number 数字类型参数那个函数一样保

持类型准确性，又可以保证用户可以根据自己需要来使用符合自己要求的参数类型与返回值类型。

谢谢观看！如果觉得课程还不错的话，记得给个好评！

我是星星课堂老师：周小周