第九课 interface 接口进阶

学习目录

- **额外任意属性**
- **函数类型接口**
- **接口继承**

一．额外任意属性

有时候我们定义好的接口需要检查的对象可能比较复杂，也去除了检查必备属性之外，还需要检查额外属性。ts 对象字面量会有额外属性检查，假如一个对象字面量存在任何“目标类型”不包含的属性时，ts 会抛出一个错误。

```
interface IConfig{
  title?: string;
  num?: number;
}
let book: IConfig= {
    abc: 800
}
```

这种错误肯定是说明我们写的对象不符合接口要求，其实是正确的，不过有时候我们期望能绕开这些检查，让我们可以给对象定义任意额外的属性。

```
interface IConfig{
```

```
    title?: string;
    num?: number;
     [propName: string]: any;
}
```

接口中允许定义任意字符串属性名称的任意类型的属性值，IConfig 可以有任意数量的属性，同时只要它们不是 title 与 num 即可。

二．函数类型接口

使用接口定义一个函数调用签名，在这个签名中定义参数列表和函数返回值类型的函数类型， 参数列表里的每个参数都需要名字与类型。

```
interface IFn{
    (a: number, b: number): boolean;
}
let sort: fn1 = function(x: number, y: number) {
    return x > y;
}
```

函数类型定义的函数参数名称不用与接口里定义的参数名称相匹配，只要类型一样即可，函数的参数会逐个进行检查，因此参数位置需要保持固定不变。

三．接口继承

ts 中的接口可以像类那样进行继承，因此我们能够从一个接口里复制类型成员到另一个接口里，接口也是用关键字 extends 来继承。

```
interface IWork{
    technology: string;
}
interface IProgrammer extends IWork{
    language: string;
    income:number;
}
```

一个 interface 接口可以继承多个 interface 接口，用逗号隔开要继承的接口。

```
interface IWork{
    technology: string;
}
interface IHobby{
    game: string;
}
interface IProgrammer extends IWork , IHobby{
    language: string;
    income:number;
    sport:string;
}
```

备注一下：假如继承的子接口中定义了与父接口中相同属性名称的类型，那么这个子接口的同名属性必须和父接口同名属性类型相同，如果类型不同的话，编译器会报错。

目前我们介绍了一些接口方面的基础实用知识，其实还有不少接口的高级用法和技巧型的用法，我们可以在之后用到的时候进一步介绍。

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