类型断言好处

6.3.1 真实应用

我们上一节刚学过 父类断言成子类，下面我们看一个父类断言成子类的真实项目场景：

父类断言成子类一般会结合泛型、多态【提前了解下有好处，后面有讲到】在大厂 koa，nestjs 等 Nodejs 服务器大中后端项目中用到，可大幅提升项目的可扩展性和代码质量。

举例【这是个真实项目的代码片段，代码较多，后面章节后讲，可以提前了解，训练自己的架构思维，耐心看懂思路，对日后开发大中项目会很受益！ 标注 S94 行的代码就是把父类断言成子类的行代码】

```
// 需求:汽车租赁功能实现: 有小轿车,大巴,卡车三种类型的车,顾客可以租任意一种或多种不同类型的车,按照租用的天计算租金， 同时为了响应国家对各类车安全的管理， 对在租赁期内有过各种超载，超乘客数，酒后驾车等违规的车需额外支付
// 汽车租赁
class Vechile {// 父类
  static count: number = 3;
  public brand: string;// 品牌
  public vechileNo: string;// 车牌号
  public days: number;// 租赁天数
  public total: number = 0;// 支付的租赁总费用
  public deposit: number;// 押金
  constructor(brand_: string, vechileNo_: string, days_: number, deposit_: number) {
    // 赋值......
  }
  // 计算租赁车的价格 ( calculateRent)
  public calculateRent() {
    console.log(this.brand + " 车牌号为:" + this.vechileNo + "开始被租");
    return 0;
  }
}

// 2 子类 Car
class Car extends Vechile {
  // public brand: string = "nobrand"
  public type: string;//车的型号
  constructor(brand_: string, vechileNo_: string, days_: number,
    deposit_: number, type_: string) {
    //  Vechile.call(this,brand_, vechileNo_, days_, total_, deposit_)
    super(brand_, vechileNo_, days_, deposit_);
    this.type = type_;

  }


  public calculateRent() {//方法重写 [override]
    // 具体业务代码省略....
    return 3
  }

  //  子类独有的方法-检查是否超重
  public checkIsWeigui(isOverWeight: boolean) {
    if (isOverWeight) {
      this.total = this.total + 500;//租金加500
    }
  }
}


// 3 子类 Bus
class Bus extends Vechile {
  public seatNum: number // 座位数
  constructor(brand_: string, vechileNo_: string, days_: number,
    deposit_: number, seatNum_: number) {
    super(brand_, vechileNo_, days_, deposit_);//使用父类的构造函数的好处
    this.seatNum = seatNum_;
    //.....
  }


  public calculateRent() {// 重写了父类的方法
    // 具体业务代码省略....
    return 10
  }

  // 子类独有的方法-检查是否超载
  public checkIsOverNum(isOverWeight: boolean) {
    if (isOverWeight) {
      this.total = this.total + 2000;//租金加2000
    }
  }
}

// 4. 子类 Truck
class Truck extends Vechile {
  ton!: number // 座位数
  constructor(brand_: string, type_: string,
    days_: number, deposit_: number, ton_: number) {
    super(brand_, type_, days_, deposit_);
    this.ton = ton_;
    //.....
  }

  checkIsOverWeight(isOverWeight: boolean) {
    if (isOverWeight) {
      console.log("超载了");
      this.total = this.total + 2000;
    }
  }

  public calRent() {
    // ......
    return 100
  }
}

// 5 Customer类
class Customer {
  // 多态在koa服务器后端大中项目中的使用
  // 父类的引用接受不同类的子类对象
  rentVechile(vechile: Vechile) {
    vechile.calculateRent();//
    if (vechile instanceof Vechile) {
      // 父类对象变量断言成子类后,调用子类独有方法
      (vechile as Bus).checkIsOverNum(true)// S94
    }
  }
}

let cust = new Customer()
cust.rentVechile(new Car("本田", "京G113", 35, 400, "1"))
cust.rentVechile(new Bus("大巴", "京G115", 89, 700, 16))
export { }
```


6.3.2 除了父类断言成子类的真实应用，很重要！另外好处还有：

1. 是学习其他断言的根基。

2. 可以帮助理解其他类型的断言，帮助理解其他断言实现的自动推导

比如：本课程 第 12 章 12-6 讲到的 Vuex4 中的 getters 方法 通过断言，最终在组件上可以自动推导 getter 方法。 **也间接通过父类断言成子类来帮助理解这个看似费解的 getter 断言，** 在其他 前端框架【比如 React] ,Vue3 源码中 同样有应用，。 再比如：在 Vue3 +TS 的项目中 断言可以巧妙解决一些响应式对象赋值出现的错误，并能巧妙的在组件上实现自动推导。