1. 为什么要理解 TS 类和 ES6 类的区别

TS 类和 ES6 类 看着很像，但又有很多不同，区分 TS 类 和 ES6 类，既可以让我们 对 TS 类 的优势印象更深刻，也会减少 TS 类 和 ES6 类 概念上的混淆。

2. 定义类属性的方式不同

（1）TS 类有多种定义属性的方式，如下：

方式 1：先在类中定义属性然后在构造函数中通过 this 赋值；

方式 2：构造函数直接为参数增加 public，给构造器的参数如果加上 public，这个参数就变成了一个属性，

默认构造函数会给这个属性赋值 [隐式操作]，以上节课的 Order 类为例，具体代码如下：

```
class Order {
    constructor(public orderId: number, public date: Date,public custname: string,
        public phone: string, public orderDetailArray: Array<OrderDetail>) {
        // 无需this赋值
     }
     ......
 }
```

（2）ES6 依然沿袭了 JavaScript 赋值的方式，在构造函数直接 this 来定义属性并赋值，代码如下：

```
class Order {
    constructor(orderId, date, custname, phone, orderDetailArray) {
        this.orderId = orderId;
        this.date = date;
        this.custname = custname;
        this.phone = phone;
        this.orderDetailArray = orderDetailArray;
    }
}
```

3. ES6 类没有访问修饰符，TS 类自带访问修饰符

ES6 类暂时还没有访问修饰符【public protected private】这也让 ES6 类设置访问权限变的异常麻烦，即使借助 call 方法或者 symbol 类型设置了访问权限局限性也很大，其实也并没有真正彻底解决访问权限的问题。这点让 ES6 类在项目中对属性和方法的权限控制很受限制。 TS 类却自带 public protected private 三种访问权限，设置访问权限轻松自如。【不设置默认访问权限为 public 】 理解访问修饰符很简单，后面我们讲解完继承后再讲解，大家很快会理解。

4. TS 类是静态类型语言的类，而 ES6 类按照 JavaScript 的一个语法标准设计而成

TS 是静态类型语言，具有类型注解和类型推导的能力，项目上线后隐藏语法和类型错误的的风险几乎为零，而 ES6 是 JavaScript 的一个语法标准，没有数据类型检查的能力，举一个简单例子来说明问题。

```
// ES6
const x = 3;
x = 10; // ES6没有任何语法错误提示

// TS
const x = 3;
x = 10;//无法分配到 "x" ，因为它是常数。
```

5. TS 类可以生成 ES5 或 ES6 或以上版本的 js 文件

通过设置 tsconfig.json 文件的 target 属性值为 ES6，那么生成的 js 文件就是 ES6 版本的 js 文件。