1. 注意：
   1. ts完全兼容js语法
   2. 路径中\*\*表示任意目录（可表示零层目录、多层目录）
   3. 语句末加！表示忽略警告
   4. 函数的形参列表中可以赋参数的默认值，若没有该参数，以默认值执行
2. Ts特性：类型、支持ES的新特性、添加ES不具备的新特性、丰富的配置选项、强大的开发工具
3. 编译设置：基本编译tsc 文件名
   1. 监视并自动编译文件：tsc 文件名 -w
   2. 配置文件：使用tsc命名编译全部文件，需要tsconfig.json文件，可以写注释
      1. 配置选项：
         1. “include”：[“需要编译的文件名”] 如“./src/\*\*/\*”，编译src下所有文件
         2. “exclude”：[“不需要编译的文件名”] 如“./hello/\*\*/\*”，不编译hello下所有文件
         3. “files”：[“需要编译的文件名”] 不能使用通配符，必须指定具体文件名
         4. “compilerOptions”：｛子选项｝ 编译器的选项
            1. “targer”：“es5” 转换成es5的版本，不完全，建议用webpack
            2. “module”：“commonjs” 指定模块化的规范
            3. “lib”：[] 指定项目中需要的库，一般不去指定
            4. “outDir”：“编译后文件所在目录”
            5. “outFile”：“编译后的文件名” 将全局作用域中的代码整合成一个文件
            6. “allowJs”：false 是否对js文件进行编译，默认false
            7. “checkJs”：false 是否检查js代码是否符合规范，默认false
            8. “removeComments”：false 是否移除注释，默认false
            9. “noEmit”：false 是否不生成编译后的文件，默认false
            10. “noEmitOnError”：false 在有错误时，是否不生成编译后的文件，默认false
            11. “strict”：false 打开所有严格选项（下面几个），默认false，建议打开
            12. “alwaysStrict”：false 编译后的文件是否使用严格模式，默认false
            13. “noImplicitAny”：false 是否取消隐式any，默认false
            14. “noImplicitThis”：false 是否取消隐式this（类型不明），默认false
            15. “strictNullChecks”：false 是否严格地检查空值，默认false
4. 语法：
   1. 类型：
      1. number、string、boolean、any、unknown、void、never、object、array、tuple、enum、HTMLElement、HTMLCollection、KeyboardEvent
         1. any与unknown：两者都可以赋任何值，但any可以赋值给任何类型的变量，unknown不能直接赋值给其他变量（if判断后可以）
         2. void与never：用来表示空值/没有值，若用于设置函数返回值，表示返回值为空/没有返回值（报错）
         3. object：指定为对象：let b：object
            1. 属性指定：let b：｛name：string，age？：number，[propName：string]：number｝

指定name为必要属性，age为可选属性，任意属性名为字符串且值为数值型的属性

* + - 1. 函数：let d：（a：number，b：number）=>number
      2. array：let e：string[] 或 let e：Array<number>
      3. tuple：元组。let f：[string，number]
      4. enum：枚举，类似于索引。enum Gender｛Male[=0]，Female[=1]｝
         1. 类中使用：let i：｛name：string，gender：Gender｝

i=｛name：‘xx’，gender：Gender.Male｝； console.log（i.gender===Gender.Male）

* + - 1. HTMLCollection：HTML元素的集合，会即时刷新（添加新元素）getElementByTagName返回的是该类型
    1. 字面量进行类型声明：let a：10 等效于let a：number
    2. 逻辑操作符：
       1. let b：值或类型 | 值或类型 指定b需要是指定值或类型中的一个
       2. let j：｛name：string｝&｛age：number｝ 指定j需要同时满足两个条件
    3. 类型别名：type xx = 条件1 | 条件2 | 条件3... xx作为别名，右侧可以是｛a：number，b：string｝
    4. 类型断言：s = e as string 或 s=<string>e 告诉解析器变量e的实际类型，用于解除报错
    5. 指定：
       1. let a：number = 1 声明变量a，并设置a的类型为number，类型错误会报错
          1. 若声明和赋值同时进行，且没有指定类型，则默认把值的类型作为变量的类型
          2. 隐式any：若声明时不赋值不指定类型，则默认视为any
       2. function xx（a：number）：number｛｝ 指定形参和函数返回值的类型
    6. 泛型：类型不明确的时候使用。function fn<XX>（a：XX）：XX｛｝ 或 class 类名<XX>｛add（XX）：XX｛｝｝
       1. 在接收到a后，XX就代指a的类型
       2. 调用时自动判断泛型代表的类型：fn（10）
       3. 调用时指定泛型代表的类型：fn<string>（‘hello’）
       4. 指定多个：function fn<XX，AA>（a：XX，b：AA）：XX｛｝
       5. 继承：function fn<XX extends 接口名或类名>（a：XX）：XX｛｝
  1. 面向对象：
     1. 类：
        1. 基本语法：
           1. 语法同es6：class 类名｛属性名：类型；constructor（参数：类型）｛｝ 方法名（）｛｝｝

属性声明可以转移到构造函数的参数中，但需要加上访问权限修饰符

* + - * 1. static关键字、extends关键字、重写、this、getter与setter同ES6
        2. readonly关键字：在属性前添加，表示只读
      1. 抽象类：abstract class 类名｛...｝ 抽象类不能用于创建对象，可以添加抽象方法 abstract 方法名（）：类型；
      2. 接口：
         1. 定义：interface 接口名｛属性：类型；方法名（）：类型；｝ 多次定义同一个接口，表示叠加定义属性
         2. 使用：

作为类型声明：const obj：接口名 = ｛xx｝

限制类的结构：class 类名 implements 接口1，接口2｛｝

* + - 1. 接口与抽象类的区别：类只能继承一个抽象类，可以实现多个接口；抽象类的方法允许实现，接口不允许
      2. 成员变量（变量、方法）的访问权限修饰符：
         1. public 允许全权访问，无任何限制（先构建对象）
         2. private 仅能在类本身中访问
         3. protected 允许类本身及子类访问（先构建对象）