接口（Interfaces）定义对象的类型

在面向对象语言中，接口（Interfaces）是一个很重要的概念，它是对行为的抽象，而具体如何行动需要由类（classes）去实现（implement）。
TypeScript 中的接口是一个非常灵活的概念，除了可用于对类的一部分行为进行抽象以外，也常用于对「对象的形状（Shape）」进行描述。

赋值的时候，变量的形状必须和接口的形状保持一致。
可选属性：该属性可以不存在

任意属性： **一旦定义了任意属性，那么确定属性和可选属性的类型都必须是它的类型的子集**
只读属性： **只读的约束存在于第一次给对象赋值的时候，而不是第一次给只读属性赋值的时候**

数组：多种定义方式

1. **元组类型「类型 + 方括号」已知元素数量和类型的数组，各元素的类型不必相同**
2. **数组泛型 Array<elemType>**
3. 接口
4. 类数组　arguments

函数的类型

两种定义函数的方式

1. 函数声明
2. 函数表达式
3. 接口定义

- **输入多余的（或者少于要求的）参数，是不被允许的**
- **=>　在 TypeScript 的类型定义中，=> 用来表示函数的定义，左边是输入类型，需要用括号括起来，右边是输出类型。**
- **可选参数　`?`表示　后面不允许再出现必需参数了**

- **参数默认值 不受「可选参数必须接在必需参数后面」的限制**
- **rest参数 只能是最后一个参数**
- **重载 允许一个函数接受不同数据或类型的参数时做出不同的处理**

内置对象

js提供的内置对象在ts核心库的定义文件是已被定义的类型。这些都是预置在ts中的，ts做了很多类型判断的工作。

TypeScript 核心库的定义中不包含 Node.js 部分，如果想用 TypeScript 写 Node.js，则需要引入第三方声明文件。

npm install @types/node --save-dev