
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
TypeScript 函数基础

目录 Contents

1. 函数概述

需求：计算数组nums中所有元素的和。

```
let nums: number[] = [1, 3, 5]
```

```
let sum: number = 0
for (let i: number = 0; i < nums.length; i++) {
  sum += nums[i]
}
console.log(sum)
```

问题：如果还要计算其他数组（nums2）中所有元素的和呢？ 拷贝一份代码，修改

```
let nums2: number[] = [2, 4, 6]
```

存在的问题：相似的代码重复写，代码冗余。

1. 函数概述

正确的姿势：使用函数来包装（封装）相似的代码，在需要的时候调用函数，相似的代码不再重复写。

```
function getSum(nums: number[]) {
  let sum: number = 0
  for (let i: number = 0; i < nums.length; i++) {
    sum += nums[i]
  }
  console.log(sum)
}
getSum(nums1) // 计算nums1中所有元素的和
getSum(nums2) // 计算nums2中所有元素的和
```

所谓函数，就是声明一次但却可以调用任意多次的一段代码。

意义：实现代码**复用**，提升开发效率。

封装：将一段代码包装起来，隐藏细节。

目录 Contents

2. 函数的使用

函数的使用分为两步：1 声明函数 2 调用函数 （类比变量）。

- 第一步：声明函数

```
function 函数名称() {
  函数体
}
```

解释：

- 函数名称：推荐以动词开头，因为函数表示做一件事情，实现一个功能。

```
function sing() {

}
```

2. 函数的使用

函数的使用分为两步：1 声明函数 2 调用函数 （类比变量）。

- 第一步：声明函数

```
function 函数名称() {
  函数体
}
```

解释：

- 函数名称：推荐以动词开头，因为函数表示做一件事情，实现一个功能。
- 函数体：表示要实现功能的代码，复用的代码。

```
function sing() {
  console.log('五环之歌')
}
```

2. 函数的使用

函数的使用分为两步：1 声明函数　2 调用函数。

- 第二步：调用函数

```
函数名称()
```

比如，调用 sing 函数：

```
sing()
```

注意：只有调用函数后，函数中的代码才会执行。

2. 函数的使用

总结：

1. 函数的基本使用分为哪两步？ 1 声明函数 2 调用函数
2. 声明函数的关键字是什么？ function
3. 不调用函数，函数中的代码会执行吗？不会

```
// 1 声明函数
function sing() {
  console.log('五环之歌')
}
// 2 调用函数
sing()
```

目录 Contents

3. 函数参数

3.1 概述

需求：让唱歌的函数（sing），每次调用时，“唱”不同的歌。

```
function sing() {
  console.log('五环之歌')
}
```

```
sing() // 五环之歌
sing() // 五环之歌
```

原因：函数（sing）中歌曲名称是固定值。

存在的问题：函数（sing）只能“唱”固定的歌，太死板，没有体现出函数复用的灵活性。

3. 函数参数

3.1 概述

使用函数参数来实现：

```
// 调用函数时，告诉函数要唱的歌曲名称
sing('五环之歌')
sing('探清水河')

// 声明函数时，接收传入的歌曲名称
function sing(songName: string) {
  console.log(songName)
}
```

函数（sing）中歌曲名称：固定值 → 动态传入的值。

函数参数的作用：增加了函数的灵活性、通用性，针对相同的功能，能够适应更多的数据（值）。

3. 函数参数

3.2 形参和实参

函数参数分为两部分：1 形参 2 实参。

1. 形参：声明函数时指定的参数，放在声明函数的小括号中（挖坑）。

```
function sing(songName: string) { }
```

- 语法：形参名称: 类型注解，类似于变量声明，但是没有赋值。
- 作用：指定函数可接收的数据。

然后，就可以在函数体中，像使用变量一样使用形参了。

2. 实参：调用函数时传入的参数，放在调用函数的小括号中（填坑）。

```
sing('五环之歌')
```

- 实参是一个具体的值（比如：‘字符串’、18、[]等），用来赋值给形参。

3. 函数参数

形参和实参的总结：

- 声明函数时的参数，叫什么？作用？　　形参，指定函数能够接收什么数据。
- 调用函数时的参数，叫什么？作用？　　实参，是一个具体的值，用来赋值给形参。

```
function sing(songName: string) { }
```

```
sing('五环之歌')
```

通过形参和实参的配合，函数可以接收动态数据，从而让函数变得更加灵活、强大。

3. 函数参数

3.3 其他说明

1. 根据具体的功能，函数参数可以有多个，参数之间使用逗号（,）来分隔。

```
function fn(name: string, age: number) { }
fn('刘老师', 18)
```

2. 实参和形参按照顺序，一一对应。

```
function fn(name: string, age: number) { }
fn('刘老师', 18) // name -> '刘老师', age -> 18
```

3. 实参必须符合形参的类型要求，否则会报错！

```
function sing(songName: string) {}
sing(18) // 报错！ 形参要求是 string 类型，但是，实参是 number 类型。
```

技巧：调用函数时，鼠标放在函数名称上，会显示该函数的参数以及类型。

3. 函数参数

总结：

- 函数形参是 `string` 类型，调用该函数时传入 18 对吗？ 不对，因为实参不符合形参的类型要求
- 函数有多个参数时，多个参数之间用什么符号分隔？ 逗号
- 以下代码会报错吗？ 报错！因为函数`sing`要求有一个参数，但是没有传

```
function sing(songName: string) { }
sing()  // 报错!
```

目录 Contents

4. 函数返回值

4.1 概述

函数返回值的作用：将函数内部计算的结果返回，以便于使用该结果继续参与其他的计算。

需求：计算以下两次调用结果的和。

```
getSum([1, 3, 5])        // 9
getSum([10, 100, 1000]) // 1110
```

```
getSum([1, 3, 5]) + getSum([10, 100, 1000]) // 9 + 1110 => 1119
```

关键点：拿到函数（getSum）内部计算出来的结果，然后，才能进行后续的加法计算。

注意：如果没有指定函数的返回值，那么，函数返回值的默认类型为 void（空，什么都没有）。

4. 函数返回值

4.2 基本使用

步骤：1 指定返回值类型 2 指定返回值

1. 指定返回值类型

```
function fn(): 类型注解 {

}
```

- 在声明函数的小括号后面，通过 ： 类型注解 指定。

```
function fn(): number {

}
```

4. 函数返回值

4.2 基本使用

步骤：1 指定返回值类型 2 指定返回值

2. 指定返回值

```
function fn(): 类型注解 {
  return 返回值
}
```

- 在函数体中，使用 `return` 关键字来返回函数执行的结果。

```
function fn(): number {
  return 18
}
```

- 注意：返回值必须符合返回值类型的类型要求，否则会报错！

4. 函数返回值

4.2 基本使用

1. 使用变量接收函数返回值

```
let result: 类型注解 = fn()
```

使用变量接收函数返回值的时候，相当于：直接将返回值赋值给变量。

```
let result: number = 18
```

注意：变量（result）的类型与函数（fn）的返回值类型要一致。

然后，就可以使用这个变量（返回值），继续进行其他计算了。

2. 直接使用函数调用的结果（返回值），进行其他计算

```
console.log( fn() * 10 )
```

4. 函数返回值

总结：

- 使用哪个关键字来指定返回值？ return
- 以下代码是否正确？ 错误！因为返回值18不符合返回值类型string的要求

```
function foo(): string {
  return 18
}
```

- 如果函数（getSum）返回了数组中所有元素的和，以下代码表示什么？ 计算两个结果的和

```
getSum([1, 3, 5]) + getSum([10, 100, 1000])
```

4. 函数返回值

4.3 return 的说明

1. 将函数内部的计算结果返回。

2. 终止函数代码执行，即：return 后面的代码不会执行。

```
function fn(): number {
  return 18
  console.log('我不会执行，放在这，没有意义')
}
```

3. return 只能在函数中使用，否则会报错。

4. 函数返回值

4.3 return 的说明

4. return 可以单独使用（后面可以不跟内容），用来刻意终止函数的执行。

```
function play(age: number): void {
  if (age < 18) {
    return
  }
  console.log('网吧上网好爽啊，王者、吃鸡两不误')
}
play(16) // 情况1：进入if后return，后续代码不执行
play(20) // 情况2：不进if，直接打印内容：  网吧上网好爽啊，王者、吃鸡两不误
```

注意：如果函数没有返回值，默认返回值类型是：void（空），可省略不写。

```
function play(age: number) { /* ... */ }
```

4. 函数返回值

`return` 的总结：

- 能否在函数外面使用 return？ 不能
- return 后面的代码会执行吗？ 不会执行
- return 后面不跟内容，单独使用，表示什么？ 刻意终止函数代码执行
- 函数没有返回值，默认返回值类型是什么？ `void`

函数基础总结

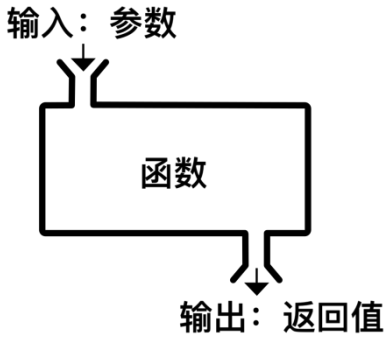
函数，即：声明一次但却可以调用任意多次的一段代码。

通过将要实现的功能，使用函数封装起来，实现代码**复用**，提升开发效率。

函数的三种主要内容： 1 参数 2 函数体 3 返回值。

简化过程：

1. 输入（参数） -- 可选
2. 处理（函数体）
3. 输出（返回值）-- 可选



传智播客旗下高端IT教育品牌