

# 29 | 泛型的基本概念和泛型工具

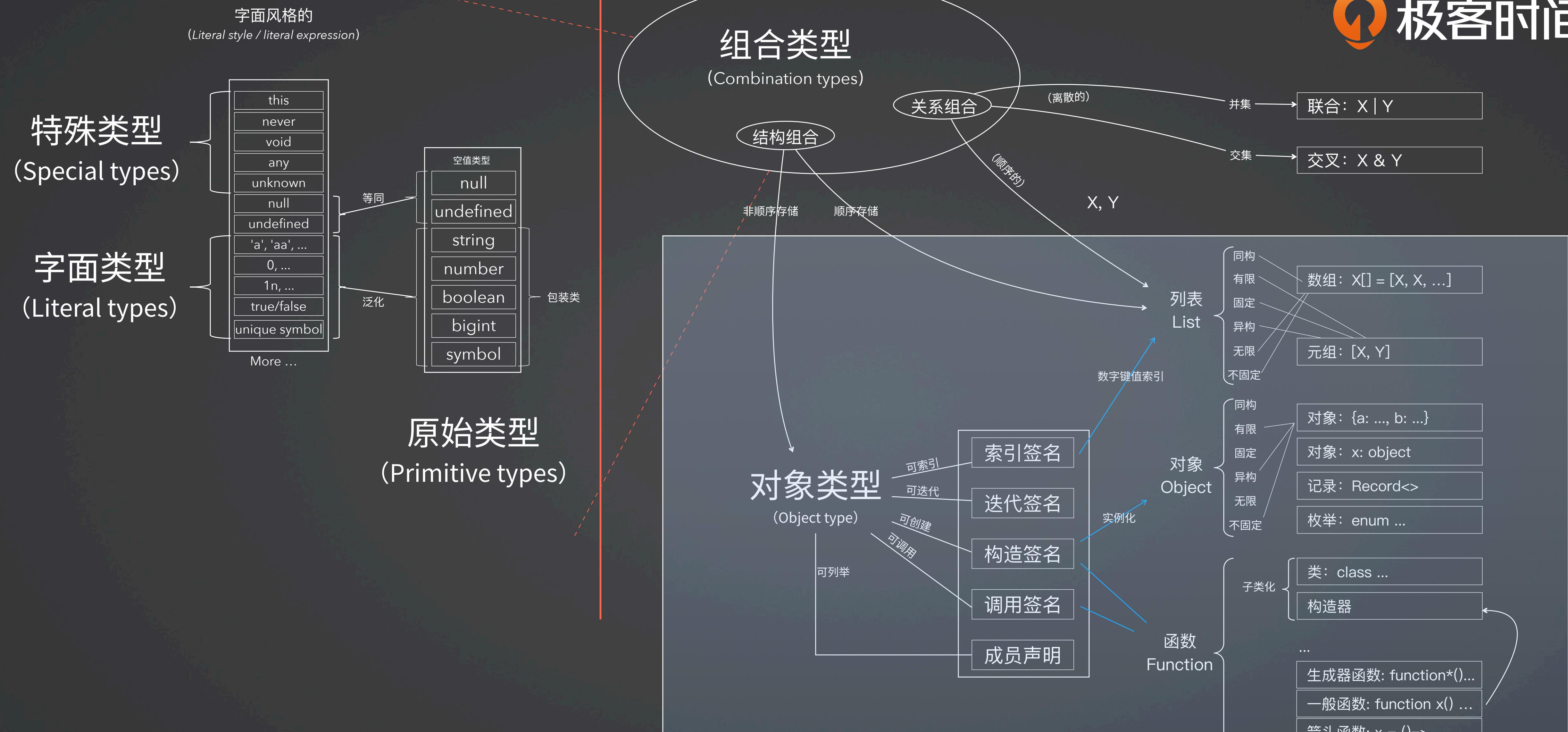
周爱民 (Aimingoo)

# 目录

1 两种泛型

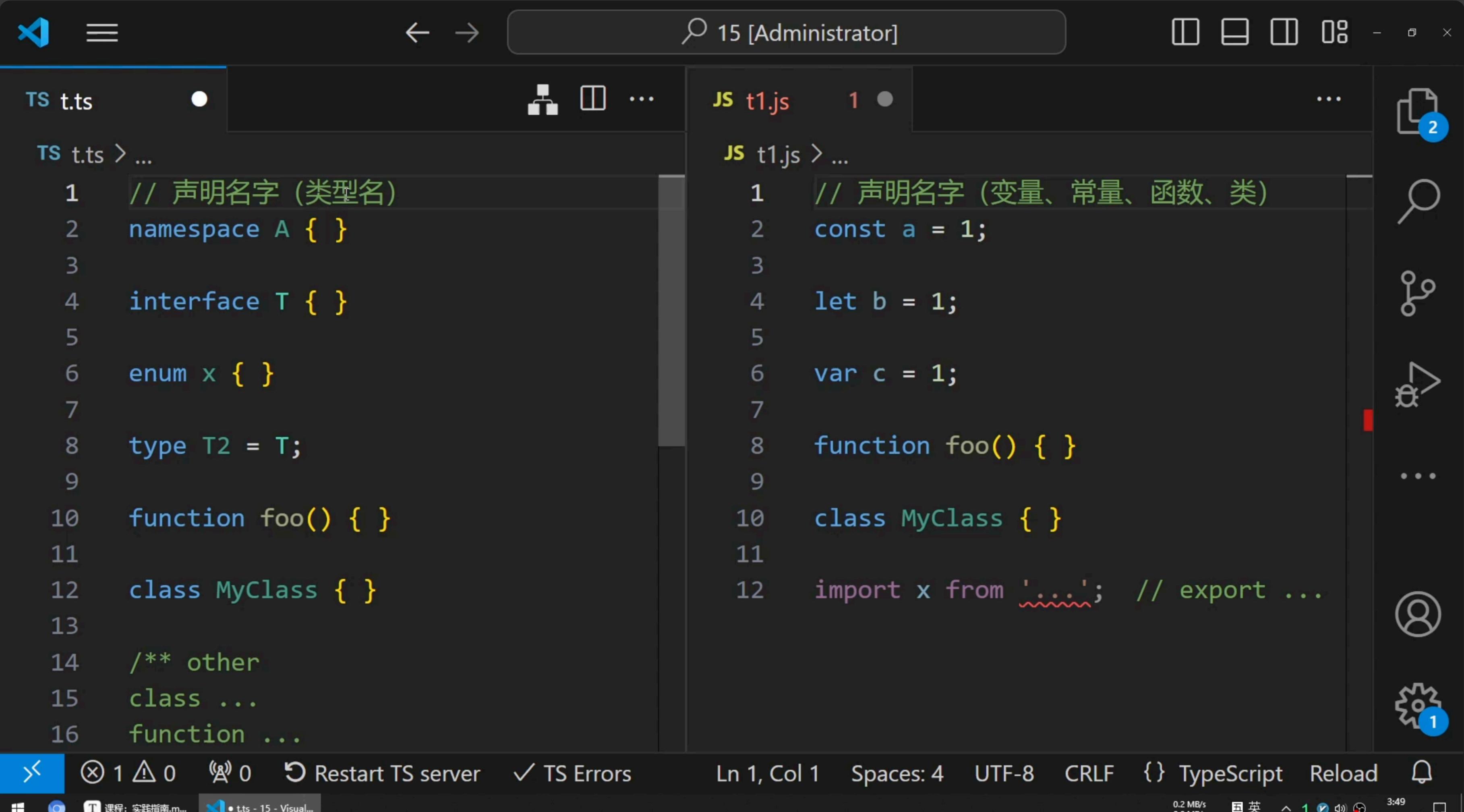
2 常见的泛型工具（工具类型）

3 总结



05 理论课：TypeScript类型系统全景

接口类型  
(Interface type)



VS Code screenshot showing two files side-by-side:

**t.ts:**

```
// 声明名字 (类型名)
namespace A { }

interface T { }

enum X { }

type T2 = T;

function foo() { }

class MyClass { }

/** other
class ...
function ...
```

**t1.js:**

```
// 声明名字 (变量、常量、函数、类)
const a = 1;

let b = 1;

var c = 1;

function foo() { }

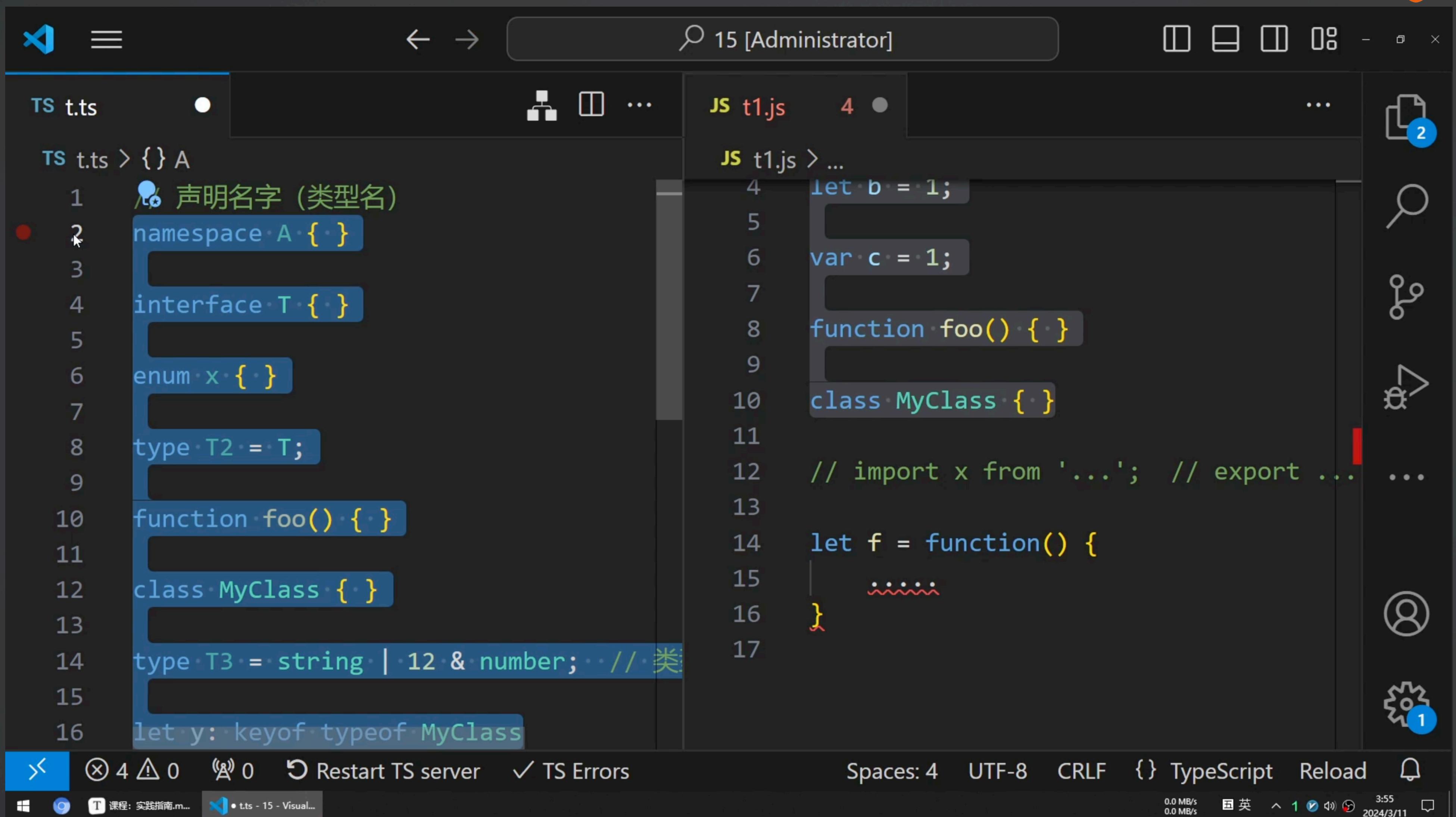
class MyClass { }

import x from '...'; // export ...
```

Bottom status bar:

- 1 TS Errors
- Restart TS server
- LN 1, Col 1
- Spaces: 4
- UTF-8
- CRLF
- { } TypeScript
- Reload

## 15 理论课：在TypeScript中的“语句”与“名字”



The screenshot shows two files open in Visual Studio Code:

- t.ts** (TypeScript file):

```
1  // 声明名字 (类型名)
2  namespace A {
3
4  interface T {
5
6  enum x {
7
8  type T2 = T;
9
10 function foo() {
11
12 class MyClass {
13
14 type T3 = string | 12 & number; // 类
15
16 let y: keyof typeof MyClass
```
- t1.js** (JavaScript file):

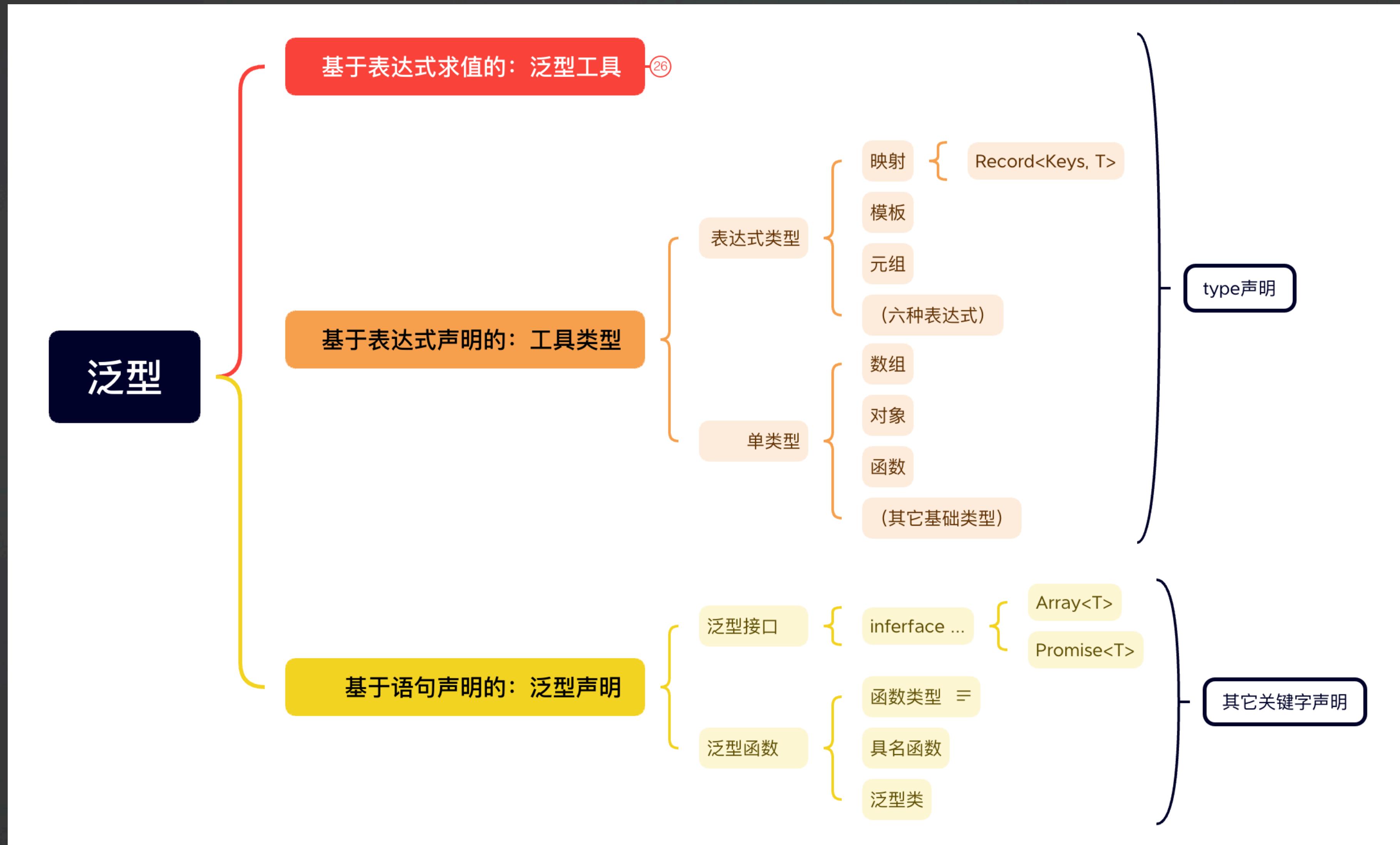
```
4  let b = 1;
5
6 var c = 1;
7
8 function foo() {
9
10 class MyClass {
11
12 // import x from '...'; // export ...
13
14 let f = function() {
15
16 }
```

The code editor features dark mode with syntax highlighting for both TypeScript and JavaScript. The status bar at the bottom provides information about the files and the environment.

# 主要概念

- 1 名字
- 2 值
- 3 语句
- 4 表达式
- 5 关键字

# 泛型



# 泛型工具

Exclude<>

Extract<>

Omit<>

Pick<>

Lowercase<>

21讲，模板字面量字符串

Promise<>

22讲，非裸类型

Record<>, Array<>

25讲，“未实例化的类型”

# 总结

## 1. 两个概念、两种泛型

- ▶ 泛型工具：基于表达式求值的
- ▶ 声明：工具类型，和（一般）泛型声明
  - 基于表达式声明的，称为工具类型，一般来说也是指“未实例化的类型”
  - 基于语句声明的，称为泛型声明（这里特指用class/function/interface关键字/语句声明的）

## 2. 常见的泛型工具

- ▶ 在TypeScript中，泛型工具与工具类型是没有明确区分的。
- ▶ Array、Record、Promise是典型的工具类型。

# 作业

>1、回忆一些在我们之前的课程中提出的泛型（类型），并尝试将它们分类。或者也可以尝试自己来为思维导图中的每个类别手写一些。

---

```
type Trans<T> = T;  
...
```

>2、查看TypeScript手册，了解每一个泛型工具的参数与用法。

- 参考阅读：《玩转TypeScript工具类型（上、中、下）》  
@see <https://xie.infoq.cn/article/61395075db832acd5d264de01>

# 名词/概念

## 名词、术语

泛型、单型：Generics / Generic types, Standalone types

### 概念

- 在TypeScript中，泛型（Generics）是一种通过类型泛化来构建类型系统的技术。“某个泛型（A generic type）”指的是一种未实例化的类型，它实例化的结果一定是某种具体类型。在TypeScript中，泛型声明的实例化是基于替换的，而泛型表达式（即泛型工具）的实例化是基于求值计算的。
- 一个泛型声明（A generic declare, or normal generic type）要么是基于语句声明的，要么是基于表达式声明的（is declared either based on a statement or based on an expression），而一个泛型工具（A general utility）总是基于表达式求值的（is evaluated based on expressions）。
- 泛化（Generalize）在一定程度上指的是概括、归纳，以及使之“普遍化”之意。从类型概念的角度上来说，就是使得某些类型含义“更抽象”。对象化通常是从根类型（Root）向下思考并设计的，而泛化则反之从下向上寻求共性抽象（abstracted from across concrete examples of algorithms and data structures and formalized as concepts）。所以泛型更适宜表达共性（而不是继承性）系统，例如类型系统。

@see: [https://en.wikipedia.org/wiki/Generic\\_programming](https://en.wikipedia.org/wiki/Generic_programming)

- 泛型声明通常以“T”为类型名前缀，这是因为C++等语言也将泛型称为“模板 / Templates”（called templates in other languages like C++），这其实也是“泛型声明的实例化是基于替换的”这一论断的来由。

@see: <https://basarat.gitbook.io/typescript/type-system/generics>

- 泛型的意义不仅在于类型间静态的共性抽象，也在于类型可以通过动态地计算求值来产生，这意味着“抽象”是一个函数化的过程。这在TypeScript中被称为从类型创建类型（Creating Types from Types），这表明泛型也可以看作表达式类型的一种求值操作，或者反过来，表达式类型也是泛型的一种特例。这其实是将函数式编程的观念引入了类型编程环境中。

@see: <https://www.typescriptlang.org/docs/handbook/2/types-from-types.html>

# Q&A

Q: 为什么“泛型声明/泛型类型”跟“泛型工具”不同。

A: 以 $R<T>$ 为例，如果设计为泛型工具，那么 $R<>$ 将转换 $T$ 为其它的某种东西，这个转换过程决定了类型 $T$ 不可能逆向推断 (infer)。如果是设计为泛型声明或泛型类型，那么在 $R<>$ 中就应该只是引用 $T$ ，而不会转换 $T$ 。

所谓“引用”和“转换”，也决定了两种“生成新类型”的不同策略。前者是基于模式替换的，亦即是将 $R<>$ 当成一个样式，而 $T$ 是其中的替换组件；后者则将 $R<>$ 当成一个运算过程，运算（求值）的目的是创建新的类型。

Q: 什么是“泛型 (Generics) ”。

A: 参考上一页PPT中的概念，泛型有两个层面的定义。包括：

1、泛型是一种类型系统的构建技术 (Generics)。在TypeScript中，它包括泛型声明 (Declare) 和泛型工具 (Utility) 两种具体的实现技术。其中，泛型声明是以“基于表达式和语句的声明 (Declare)”为核心的方法，包括：

- \* 一种类型声明的方法，其结果是“泛型/泛型XX (Generic Types / Generics XX Types)”；
- \* 可以将函数声明或类声明“模板化”的方法，从而使JS支持了“泛型函数或类 (Generic Functions/Classes)”。

2、泛型是一种未实例化的类型 (Generic Types)。



THANKS