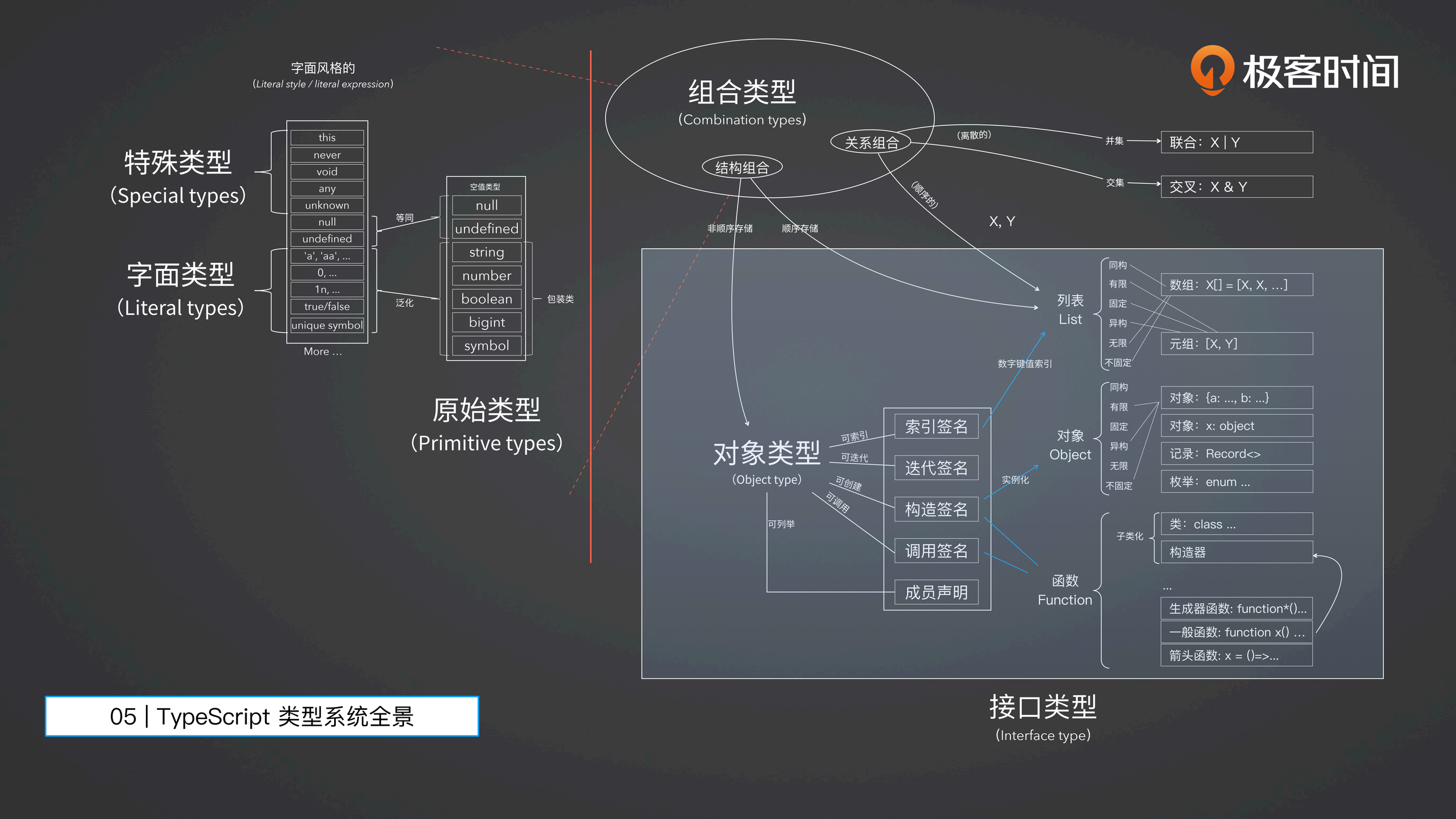
10 | 索引签名与列表（数组和元组）

周爱民（Aimingoo）

目录



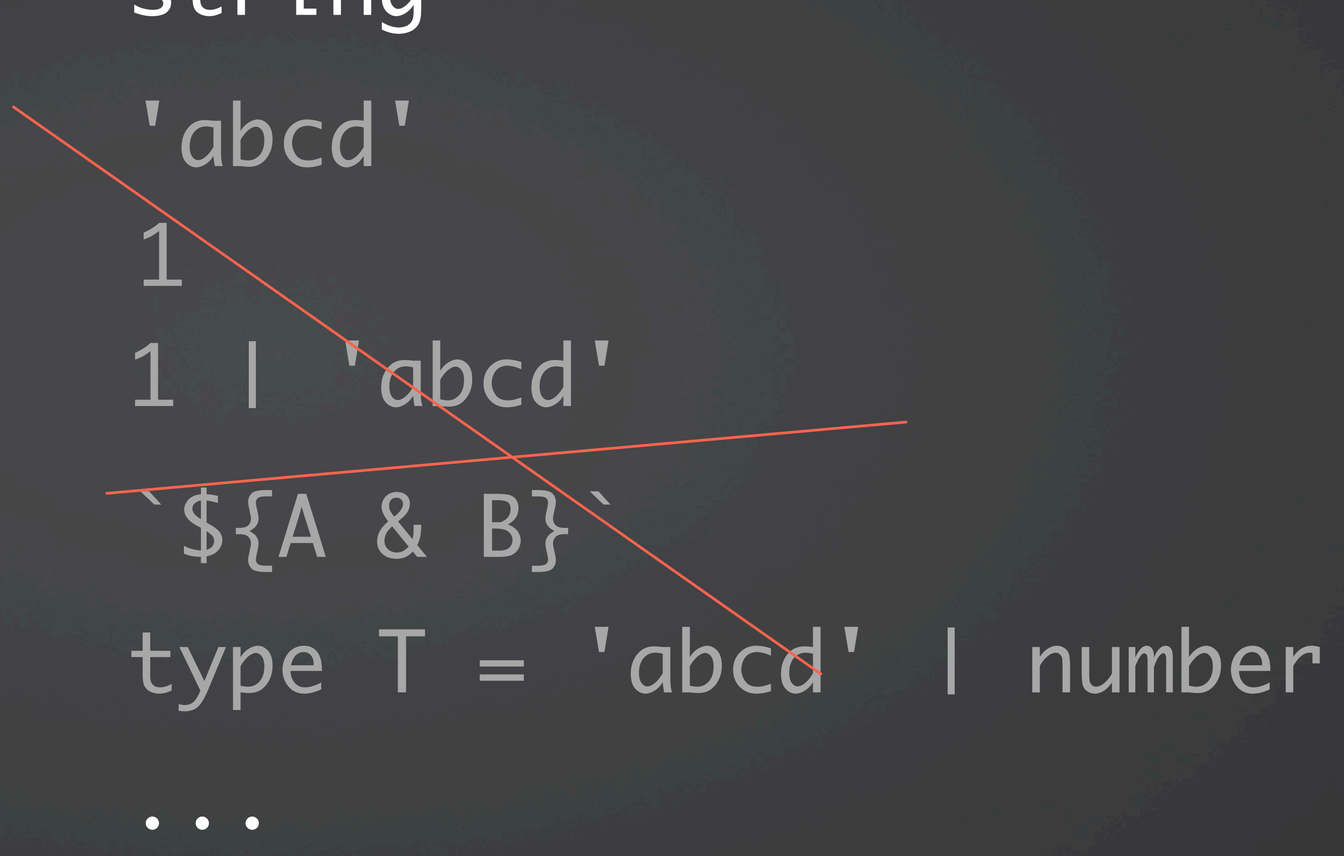
极客时间

05 | TypeScript 类型系统全景

索引签名的写法

```ts
// 签名与多重签名
interface X {
  [key: ...]: string;
  [key: ...]: number;
  ...
}

// (同上)
type X = {
  ...
}
```



一般签名用法

支持使用 obj[x] 语法：

- 在声明中添加签名
- 临时限制（使用as）
- 中间类型（使用&）

将签名用作限制

在类声明时，implements 一个签名的意义在于成员规格的限制。而具体使用时，仍然可以使用类名来正常访问成员。

```
interface X {
    [key: string | '1' | '2' | '3']: string;
}

class MyClass implements X {
  [x: string|number|symbol]: any; // override
  a: string = 'hi';
  foo() {
    return 1;
  }
}

let x: X = new MyClass;
x[1] = 100;  // fail
let obj = new MyClass;
obj[1] = 100; // pass
```

总结

1. 索引签名只有一种写法，可以用于对象、接口和类类型，并且只对三种基本类型有意义
   - 不可重复声明
   - number 必须与 string 兼容
   - 签名的作用：限制成员类型 + 支持计算属性存取

2. 数组和元组都是在对象的索引签名上做文章
   - 数组成员的类型可以是混合的（例如 any 类型，或者联合类型）
   - 在索引签名上，数组与元组的区别主要在于 length 属性，并且元组显式地指定了确定下标的类型
   - 数组可以用两种方法来声明，并且兼容/支持 JavaScript 中的全部创建与初始化方式
   - 元组的需求来自于函数参数列表，并且也因此有了标签化元组

3. 只读数组和类数组（课后）

作业

> 分析如下代码，解释 interface MyClass 声明的意义和 foo() 为什么声明失败：

```
interface MyClass {
    [key: string | '1' | '2' | '3']: string;
 }

class MyClass {
  a: string = 'hi';
  foo() { // fail
    return 1;
  }
}

let obj = new MyClass;
obj[1] = 100; // fail
```

名词/概念

名词、术语

索引签名、索引集：	Index Signatures, Indexed Collections
数组、只读数组、类数组：	Array type, ReadonlyArray type, ArrayLike type
元组、只读元组、标签化元组成员：	Tuple type, readonly Tuple type, Labeled tuple elements

概念

- 数组（Array）是由相同类型的元素的集合组成的数据结构（*consisting of a collection of elements (values or variables), of same memory size*）。在JavaScript中的“数组”可以是异构的（由不相同类型的元素构成），但TypeScript中强制要求所有的Array类型必须是同构的。使用联合类型作为元素类型（基类型）可以在某种程度上实现异构的效果，但是也会在代码中引用更复杂的处理逻辑。
  @see: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures
  @see: https://www.typescriptlang.org/docs/handbook/2/objects.html#the-array-type
- 使用ReadonlyArray<>泛型工具可以将数组声明为只读的，但也可以使用语法`readonly X[]`来实现完全相同的声明。
  @see: https://www.typescriptlang.org/docs/handbook/2/objects.html#the-readonlyarray-type
- 元组类型（Tuple）也是一种数组类型，但它确切地知道它包含多少个元素，以及它在特定位置包含哪些类型（*knows exactly how many elements it contains，and … at specific positions*）。它可以使用修饰字来说明它是只读的，例如`readonly [string, number]`。
  @see: https://www.typescriptlang.org/docs/handbook/2/objects.html#tuple-types
- 在JavaScript中，数组和类型数组被称为索引集（可索引集合类型，*Indexed Collections*），与之对应的是有键集（*Keyed Collections*）。后者包括Map、Set等（事实上一般的Object对象也包括在这个集合中）。之所以这里称“有键”而不是“键值”，是因为Set类型是有键无值的。
  @see: https://tc39.es/ecma262/#sec-indexed-collections
- 索引签名用于描述类型的“可能值的类型（*types of possible values*）”。索引签名的类型（这里指的是`[key: keyType]: valueType`中的keyType）可以是字符串、数值、符号、模板字符串模式，以及它们的联合（*string, number, symbol, template string patterns, and union types consisting only of these*）。
  @see: https://www.typescriptlang.org/docs/handbook/2/objects.html#index-signatures

Q&A

Q: 为什么我明明没有写索引签名，但能使用下标存取

A: 对于一般对象，`obj[x]` 这个语法并不总是违例。如果一个计算属性的结果（例如字面值）落在已知的成员清单里，那么这个存取总是有效的；只有当计算属性的结果不能直接求值，或者求值结果可能不在成员清单里时，才需要索引签名支持。

Q: 为什么 any 类型可以访问任何下标

A: 因为 any 类型内置了下面的签名（完整签名），所以 any 类型可以作为对象使用，并访问任意类型的下标。

```
{
  [key: string | number | symbol] : any
}
```

Q: 索引签名语法“[key: keyType]: valueType]”中的 key 是什么?

A: 这个 key 只是一个占位符，不强制使用任何的名字，并且在多个签名中也可以重名。习惯中，可以使用 key、name、n、s 以及 index 等等。

Q: 在 keyType 中可以使用的“模板字符串模式”是什么?

A: 在 TypeScript 的手册中指出这里可以使用“模板字符串模式（template string patterns）”类型。这可以理解为带模式匹配的字符串类型，但是我们要到本课程的第二篇中才会讲到它的时候，才会再次讲到它中索引签名中的用法和影响。


极客时间
THANKS