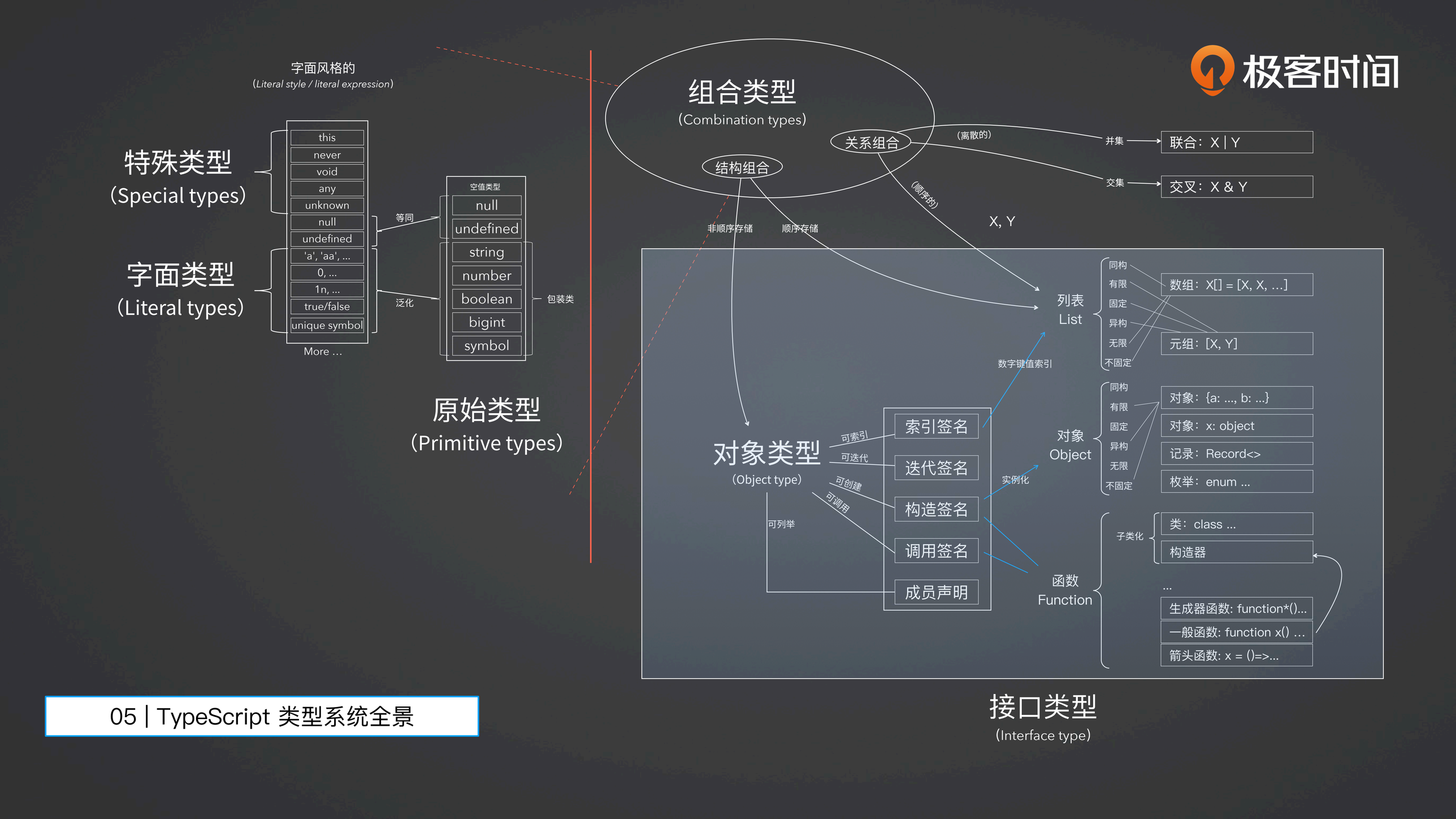
09 | 函数签名与传统的类（构造器）

周爱民（Aimingoo）

目录



极客时间

05 | TypeScript 类型系统全景

重载的本质是多重签名

1. 重载是“具名的函数类型”所需的语法支持
   - 方法、构造方法和具名函数

2. 要注意语义上的差别（type vs. interface）
   - type总是声明出一个类型（的别名）
   - interface总是声明出一个对象的界面（签名用于声明附加性质）

函数

```
let f: () => void = function() { ... };

type Foo = (...args: any[]) => any;

interface IFoo {
  (): void;
  (...args: any[]): any; // or void
}
```

签名的不同写法

```
// （带调用签名的）函数类型Foo
type Foo = () => void;
```

```
// （带签名的）接口类型T
interface T {
  (): void;
  foo() { ... };
  foo(x: any) { ... };
  a: string;
  ...
}
```

```
// 1. 某个函数对象的接口IFoo
// 2. 接口语法的签名
interface IFoo {
  (): void;
  new (...args: any[]): ...;
}
```

```
// 函数foo()的重载语法的调用签名
function foo(): void;
function foo() { // 具名函数声明
  ...
}
```

```
// 类MyClass的重载语法的构造签名（方法签名略）
class MyClass {
  constructor(s: string);
  constructor() { // 构造方法声明
    ...
  }
}
```

类 vs. 构造器

```
interface MyClass {
  b: ...
}

type MyClassConstructor = typeof MyClass;

class MyClass implements MyClass {
    a: string = 'abcd';
    ...
}
MyClass.prototype.b = …
```

```
interface MyClass {
   a: string;
   ...
}

interface MyClassConstructor {
    new (): MyClass;
    (): void;
    readonly prototype: MyClass;
    ...
}

let MyClass = function() {
    this.a = 'abcd';
    ...
} as MyClassConstructor;
MyClass.prototype.b = …
```

```
let x: MyClass = new MyClass();
```

总结

1. 什么是签名?

2. 调用签名与构造签名
   - 补充了签名与重载之间的关系
   - 函数签名的最基础形式（根级的函数）
   - 具名函数在作为“接口的成员列表”这个语义时采用了其它的方法

3. 构造器的写法

作业

1. 如下代码在构造器语法中如何实现？请写出示例和测试。

```
interface IFoo {
  (): void;
}

class MyClass implements IFoo {
  ...
}
```

2. 查阅资料，写出“抽象构造函数”签名的写法。亦即是如下代码在传统构造器中的写法：

```
abstract class MyClass {
  ...
}
```

名词/概念

名词、术语

签名、调用签名、构造签名：Signatures, Call signatures, Construct signatures

构造器：Constructor

抽象构造签名：Abstract construct signatures

概念

- 当抽象类用作基类（Base）时，其子类必须实现抽象类所有的抽象成员（*for subclasses which do implement all the abstract members*）；当类中没有任何抽象成员时，并且没有未实现的基类抽象成员时，称为具体的（*it is said to be concrete*）。
  @see: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures
- 类声明中，类名最终被用作构造器，注意类声明是构造函数的语法糖（*classes are really just syntactic sugar to prototype-based constructor functions*）。因此构造签名会出现在类名所对应的类型上，例如`typeof MyClass`；同样的原因，abstract这个修饰字也作用于类声明而非构造方法声明，例如`abstract class MyClass …`。
  @see: https://www.typescriptlang.org/docs/handbook/decorators.html#class-decorators

Q&A

Q: “函数类型”与“有调用签名的接口”有什么不同?

A: 二者只是语义概念上的区别。函数类型表达为一个字面的函数声明语法，例如`() => void`，它不显式地说明“函数是一个对象”，某种程度上说，它反映的是 JavaScript 中的“`typeof foo === 'function'`”的语义。而“有调用签名的接口”先说明该类型是一个接口类型，这反映了 foo 的如下特性：1、必然首先是一个对象，2、然后才是对象的接口上“有调用签名”，3、以及作为接口，还“可以有更多的成员或签名”。

同样的原因，“函数类型”没有办法添加静态成员声明和其它签名，而“有调用签名的接口”则没有这些限制。但是函数类型也可以通过“交叉（&）”其它类型（通常是接口）来实现类似的语法特性。

Q: 本讲中在函数参数中出现的“this”是 `this` 类型吗?

A: 不是。在这一讲中，我们只提及了参数表中的第一个参数可以使用“`this`”这个关键字，用于声明 `new` 构造出来的实例的类型。这种“将参数列表的第一个参数声明为 this”的语法还可以用于一般的方法声明，在其它语言中也有类似的语法。——这一语法的目的只是将“隐式声明函数传入了一个 this 参数”这件事显性化，以便能够做类型标注而已。

与此不同的是，“`this`类型”在 TypeScript 中是一个类型名（而不是参数名/变量名）。简单地说，它可以出现在类型标注中，例如foo(this: `this`, x: `this`): `this`; 这行声明中的第一个this就是参数名，而后面的三个this都是类型名，这三个才是我们说的“`this`类型”。


极客时间
THANKS