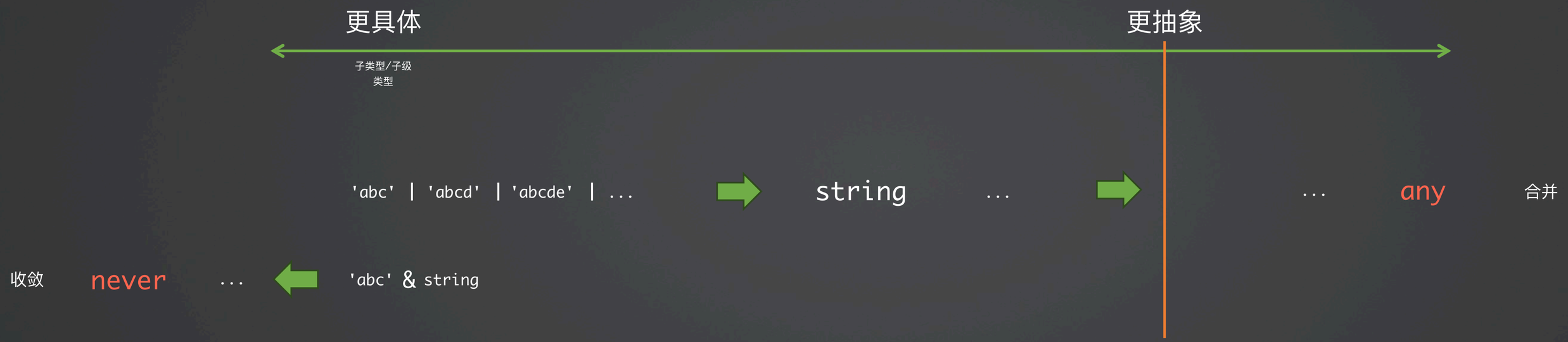
06 | 接口类型的联合与交叉

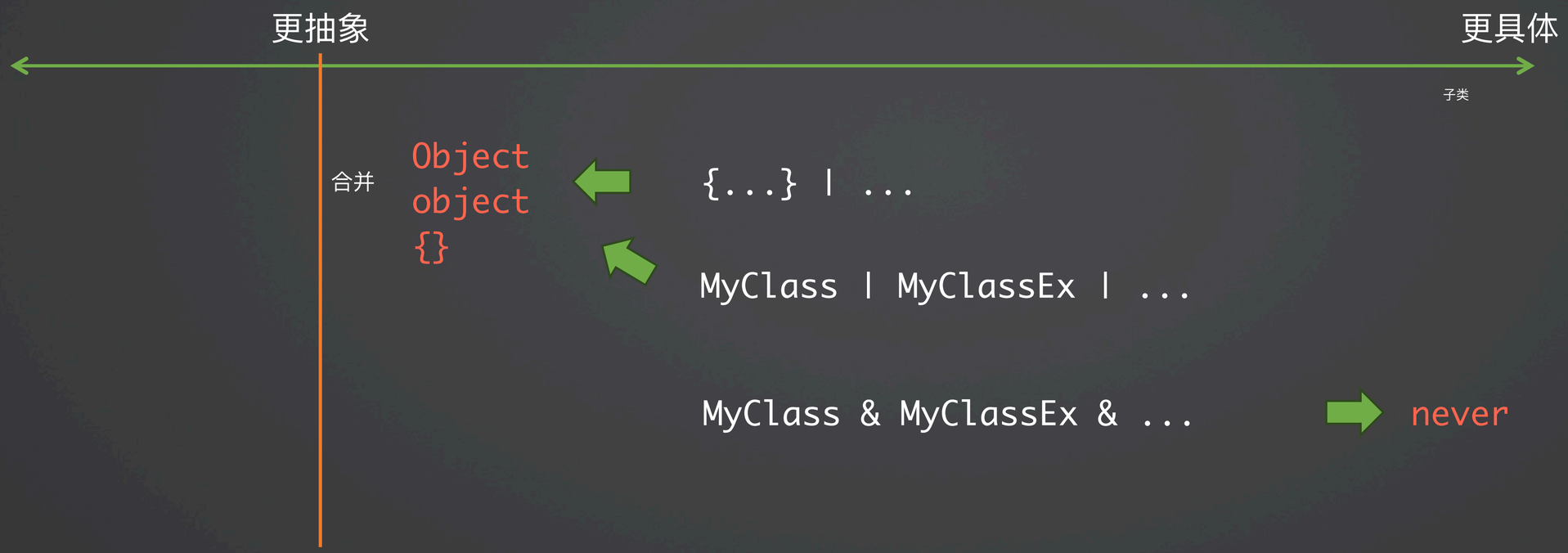
周爱民（Aimingoo）

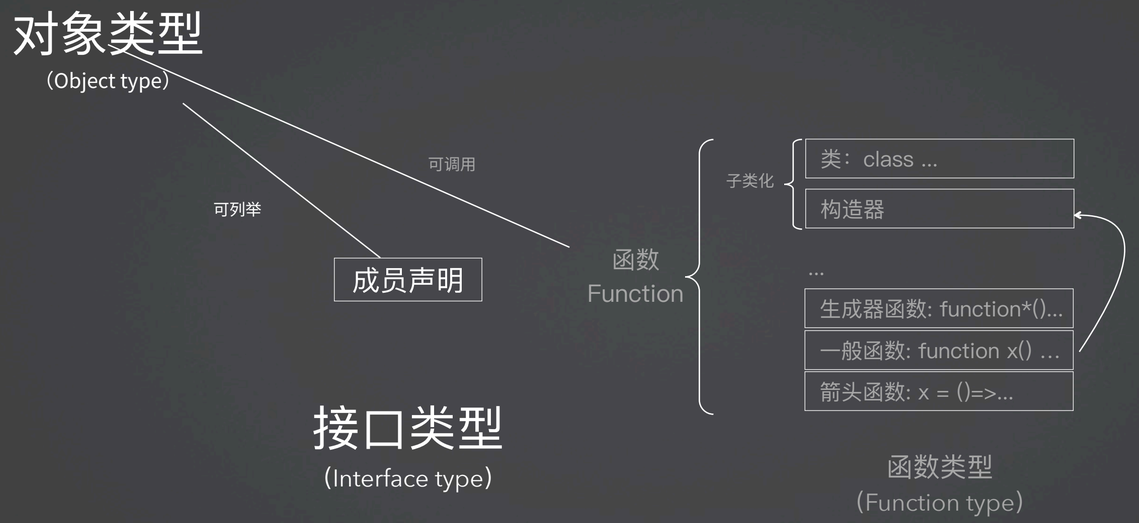
目录

联合与交叉

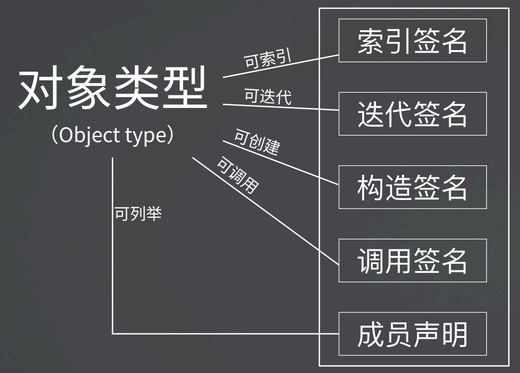

更具体
更抽象
子类型/子级
类型
'abc' | 'abcd' | 'abcde' | ...
string
...
...
any
合并
收敛
never
...
'abc' & string

接口的联合与交叉


更抽象
更具体
子类
合并
Object
object
{}
{...} | ...
MyClass | MyClassEx | ...
MyClass & MyClassEx & ...
never
收敛



❎ 联合的合并　　✅ 交叉的收敛



接口类型
(Interface type)

总结

1. 接口的联合类型可以被求值，但不一定有意义

2. 接口的交叉是深度遍历并对每一个成员求交叉的
   - 接口的交叉会包括它们的签名

3. 接口的交叉类型总是尽量向 never 收敛的
   - 如果没有收敛成 never，则该交叉类型总是可以被求值的

Q&A

Q: 在课程中提及的“不需要初始化”的语法是什么?

A: 这称为“确定赋值断言（*definite assignment assertion*）”，该语法将一个“!”用作修饰符（modifiers）用在一个变量的类型标注之前，表明该变量在后续使用时不需要初始化。注意这里的“!”是一个修饰符语法，而不是运算符。 Ref: ts(2454)

Q: Omit<>是什么，有什么用?

A: 在上一讲中我们提到过一个用相同语法表示的东西，就是Record<>，它们都称为泛型工具（Generic utilities），也称为工具类型（Utility Types）。工具类型都使用“X<>”这样的语法来“调用”，类似于一般函数语法，例如“X()”。`Omit<T, U>` 表明它有两个参数，T 通常是一个接口类型，而U是一个联合，用于从 T 的成员列表中排除掉 U 联合中所指明的那些 keys，并返回一个新的接口类型。——注意所有工具类型的返回结果都是新的类型，所谓参数、返回结果等概念与函数的语义是一致的。

Q: 接口继承与类继承有什么不同?

A: 在使用 interface 和 class 关键字时，都可以使用 extends 子句，以指示所声明类型的父级（super）。不同的是，1、在概念上，声明`` `interface X extends Y { }` ``时，X是Y的子类型，而换成 class 声明时 X 是 Y 的子类；2、interface 可以使用多重继承，也就是说在`` `extends …` ``子句中可以用“,”分隔符来声明一个类型列表，表明 X “继承自”多个接口，而 class 不支持多重继承，只能在它的`` `implements …` ``子句中使用接口列表，以表明类X“实现了”多个接口。

Q&A

Q: 根级的类型为什么有三个？有什么不同?

A: 概念上来说，这里指的是对象系统（Object type system）中继承树上的根类型（Root type）。TypeScript 在 OOP 以及相关语义中使用了三个根类型。其中，`{ }`称为空白接口或空白对象类型，这两个概念用在不同的上下文中，用以表示接口类型的根类型，或字面对象类型的根类型。而 `object` 表达的不是“根类型”这样的带有面向对象继承性语义的概念，而是所有对象字面类型的“父类型”，这是“子类型系统（Subtyping system）”中的概念。

而 `Object` 是JavaScript中所有类的根类，因此对应的 Object 接口类型，也就成了所有对象的根级接口类型。——这里还存在一个概念转换，即，“类声明对象实现的接口”，亦即是说“Object是它的实例（亦即是对象）的接口”。关于“类声明与接口”的关系，在后续的“08 | 类、接口以及对象类型的相互操作”这一讲会专门讲述。

最后需要补充的是，1、如果你需要强调变量x使用“任意的字面对象”，那么更推荐使用“`x: object`”，而不是“`x: { }`”来声明；2、除非你明确地需要使用 `new Object()` 来为对象x赋值，否则任何情况下都不建议使用“`x: Object`”来声明该变量。——所有使用包装类来声明变量类型的方式，都是不推荐的，例如 String、Number 或 Boolean 等。

Q: “联合和交叉是用来计算的类型”是什么意思?

A: 这是一个非常重要的概念，涉及到“类型计算（Computing with types）”或者“计算类型（Computed types）”，相关的内容会在本课程的第二篇（15讲+）之后再讲述。在本讲中，我们强调某些计算“不会发生”，是刻意将TypeScript的类型系统缩减到一个“静态化的类型系统”来讨论的，这是本课程的第一篇（15讲之前）的主题。


极客时间
THANKS