19 | keyof 的特殊性

周爱民（Aimingoo）

目录

1 keyof T 可以计算任何类型

2 keyof T 中的 T 是联合时的计算方式

3 索引存取运算中的枚举类型（上一讲遗留）

4 keyof T 中的 T 是带索引签名的接口类型时的计算方式

5 总结

表达式类型

1、作为表达式，自身要求值（表达式求值）

2、作为表达式类型，要参与其它运算符的运算（操作数）

类别	名称	运算符	（注）	求值结果	优先级
语法（上下文受限）	分组/括号	(...)			10
	模板变量	${T}	（注8）		10
	展开	···T			10
	映射	x in X as T			10
	模板字面字符串类型	`...`	（注7）		9
	元组类型	[...]			9
	映射类型（注3）	{ ... }			9
表达式	类型查询	typeof V	(* V是变量名)	任意类型	8
	索引访问（类型）	T[K]	(* T和K都被立即求值)	联合（包括任何单类型）	7
	键名查询	keyof T	(* T被立即求值)	联合（包括任何单类型）	6
	交叉类型	A & B		单类型	5
	联合类型（注2）	A | B		联合（包括单类型或或交叉得到的单类型）	4
	条件类型	extends		X, Y, X | Y	3
		（保留）			2
语法（上下文受限）	断言、标注等	as, <>, :			1
	约束	extends ···			1
	别名/缺省值（注6）	= ...			1
		(保留, 例如,号等)			0

总结

1. `keyof T` 中的 T 总是先求值的
   - T 可以是任意单类型
   - 如果 T 是表达式类型，那么它的求值结果要么是单类型，要么是一个联合类型
2. `keyof U` 的计算中，返回的结果是 `keyof U1 & ... & keyof Un`
3. `keyof T` 中，如果 T 是带索引签名的类型时，结果“可能”会被合并
   - 所谓“可能”合并其实是一个表面现象，`keyof` 是惰性求值的，因此是否合并还取决于后续的求值运算
   - keyof T 在“映射类型”的 `in` 运算中有着不同的表现

作业

> 论述题：试说明 `keyof U` 的计算方式与“接口的联合相当于求它们的公共父类”之间的关系。


极客时间
THANKS