23 | 赋值兼容性的再说明

周爱民（Aimingoo）

极客时间

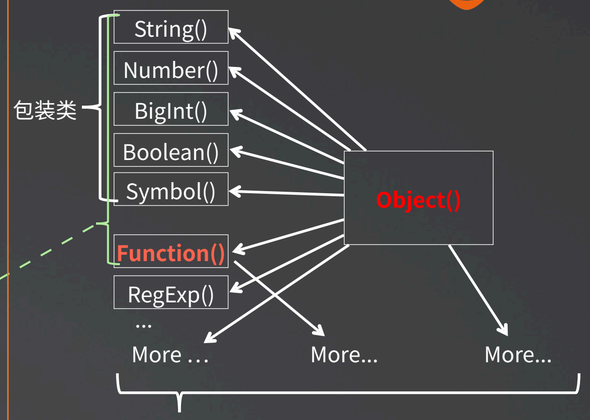
目录

JS类型 => TS类型

Literal style

Construct style
(with/without package classes)

极客时间



对应的类型

对象类型系统
(instanceof)

原始值类型(同es)
(Primitive types)

null | undefined | string | number | bigint | boolean | symbol | object | function

Above is collective type of literal types

基础类型系统
(typeof)

类/构造类型
(Class/Construct type)

never | void | any | unknown | this

特殊类型
(Special types)

null | undefined | 'a', 'aa', ... | 0, ... | 1n, ... | true/false | Uni-symbol

字面类型
(Literal types)

object

字面对象：{ ... }
数组与元组对象：[...]
字面正则对象: /.../...
...

对象类型
(object type)

function

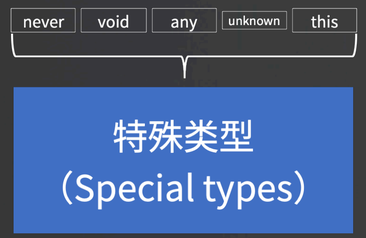
箭头函数: () => ...
具名函数: function f() { ... }
匿名函数: function () { ... }
...

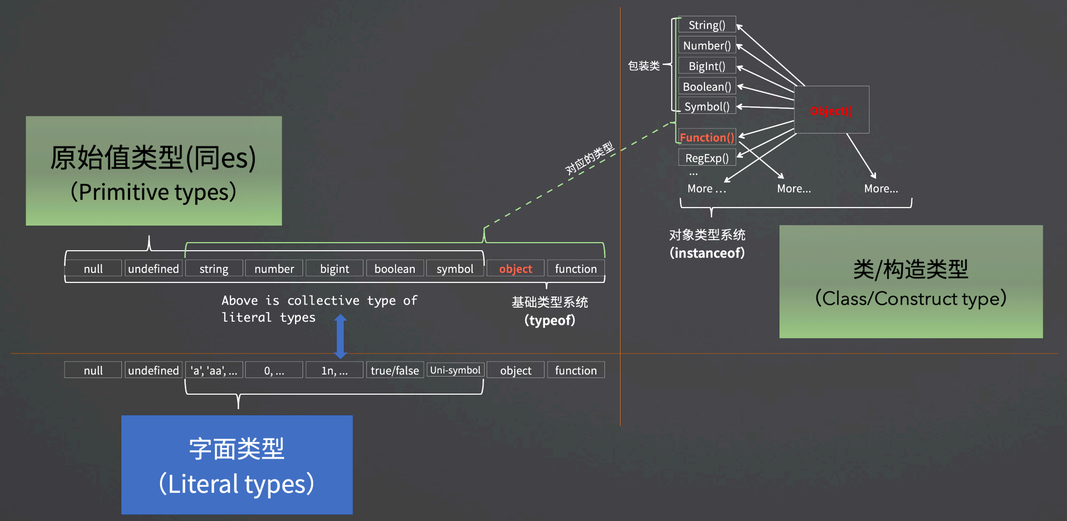
函数类型
(function type)

接口类型
(Interface type)

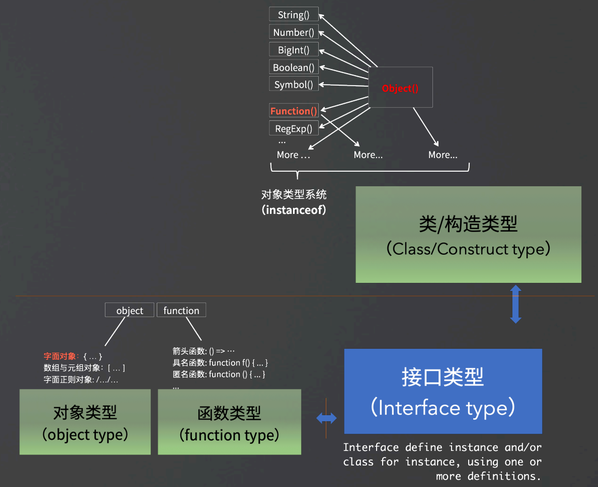
Interface define instance and/or class for instance, using one or more definitions.

赋值兼容

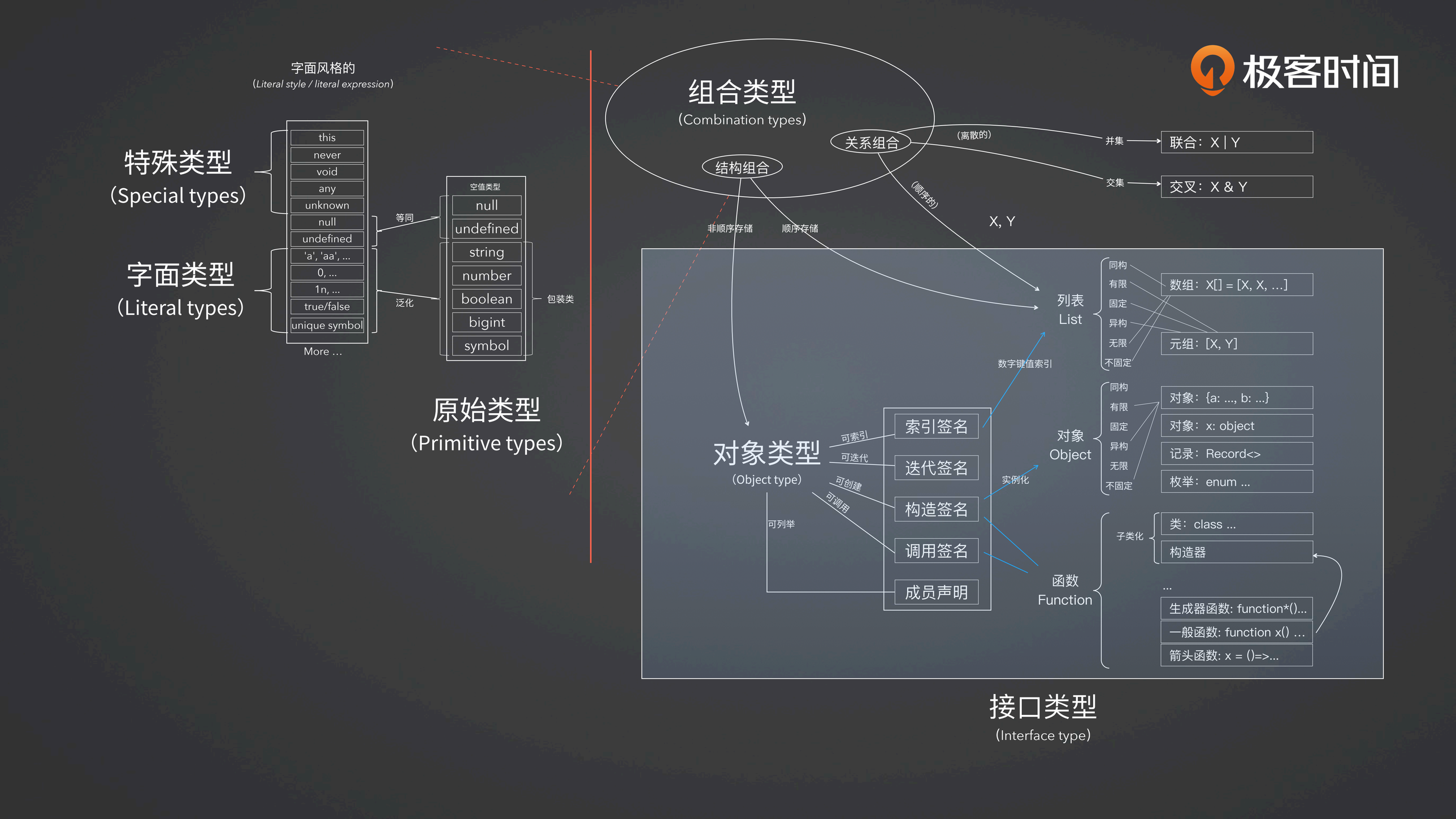


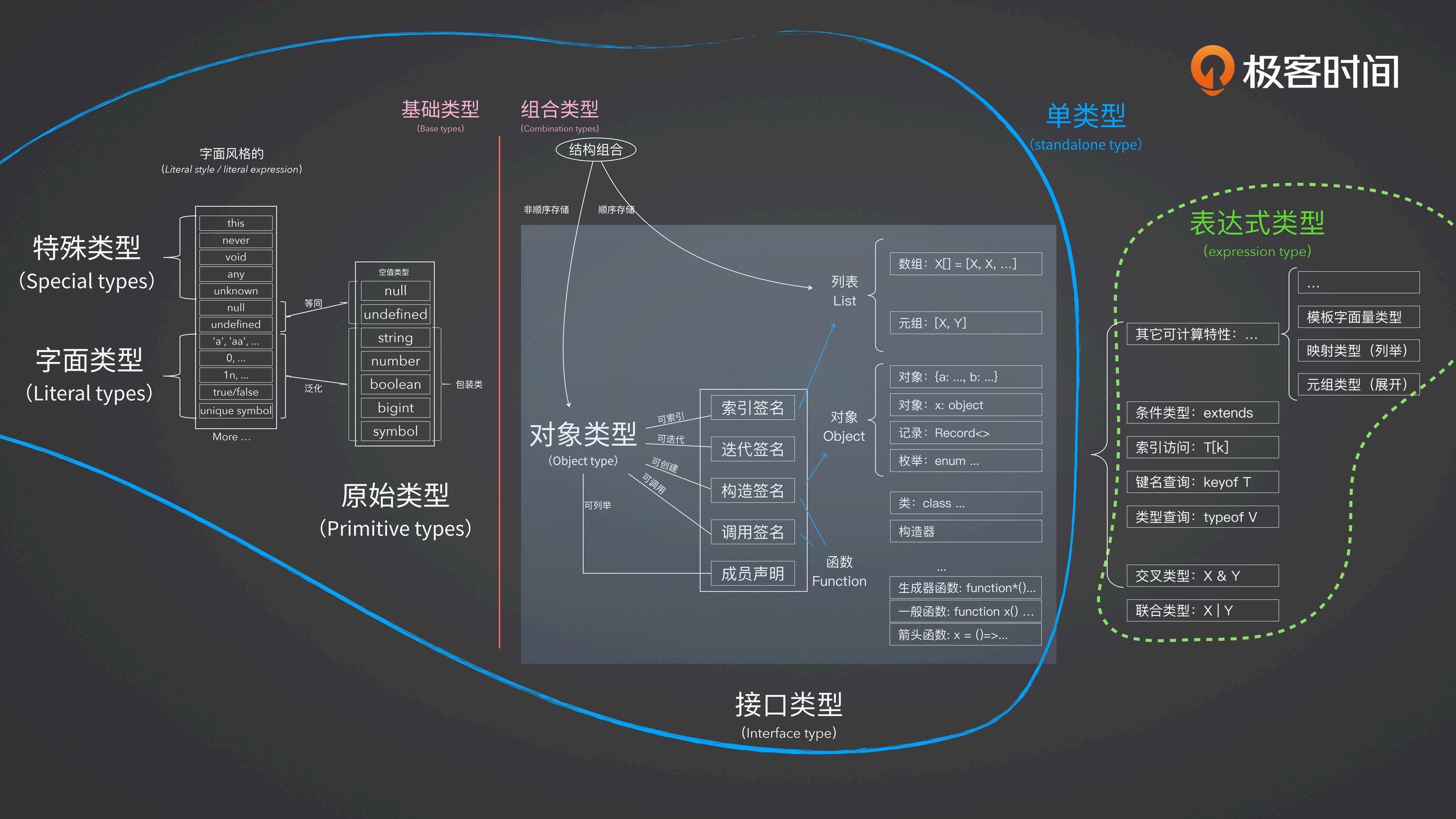


子类型兼容



结构类型兼容


字面风格的
(Literal style / literal expression)
特殊类型
(Special types)
字面类型
(Literal types)
this
never
void
any
unknown
null
undefined
'a', 'aa', ...
0, ...
1n, ...
true/false
unique symbol
More ...
等同
泛化
空值类型
null
undefined
string
number
boolean
bigint
symbol
包装类
原始类型
(Primitive types)
组合类型
(Combination types)
关系组合
结构组合
(离散的)
(顺序的)
并集
交集
联合：X | Y
交叉：X & Y
X, Y
非顺序存储
顺序存储
对象类型
(Object type)
可索引
可迭代
可创建
可调用
可列举
索引签名
迭代签名
构造签名
调用签名
成员声明
数字键值索引
实例化
列表
List
对象
Object
函数
Function
同构
有限
固定
异构
无限
不固定
数组：X[] = [X, X, ...]
元组：[X, Y]
对象：{a: ..., b: ...}
对象：x: object
记录：Record<>
枚举：enum ...
子类化
类：class ...
构造器
...
生成器函数: function*()...
一般函数: function x() ...
箭头函数: x = ()=>...
接口类型
(Interface type)


基础类型
(Base types)
组合类型
(Combination types)
单类型
(standalone type)
字面风格的
(Literal style / literal expression)
特殊类型
(Special types)
字面类型
(Literal types)
this
never
void
any
unknown
null
undefined
'a', 'aa', ...
0, ...
1n, ...
true/false
unique symbol
More ...
等同
泛化
空值类型
null
undefined
string
number
boolean
bigint
symbol
包装类
原始类型
(Primitive types)
结构组合
非顺序存储
顺序存储
对象类型
(Object type)
可索引
可迭代
可创建
可调用
可列举
索引签名
迭代签名
构造签名
调用签名
成员声明
列表
List
数组：X[] = [X, X, ...]
元组：[X, Y]
对象
Object
对象：{a: ..., b: ...}
对象：x: object
记录：Record<>
枚举：enum ...
函数
Function
类：class ...
构造器
...
生成器函数: function*()...
一般函数: function x() ...
箭头函数: x = ()=>...
接口类型
(Interface type)
表达式类型
(expression type)
其它可计算特性：...
...
模板字面量类型
映射类型（列举）
元组类型（展开）
条件类型：extends
索引访问：T[k]
键名查询：keyof T
类型查询：typeof V
交叉类型：X & Y
联合类型：X | Y

简单类型间的赋值兼容性

x \ T	null	undefined	void	never	any	unknown	'a'	1	true	false	string
T_null	TRUE	FALSE	FALSE	TRUE	boolean	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
T_undefined	FALSE	TRUE	FALSE	TRUE	boolean	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
T_void	FALSE	TRUE	TRUE	TRUE	boolean	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
never	FALSE	FALSE	FALSE	TRUE	boolean	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
any	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
unknown	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
'a'	FALSE	FALSE	FALSE	TRUE	boolean	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	FALSE	TRUE	boolean	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
T_true	FALSE	FALSE	FALSE	TRUE	boolean	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
T_false	FALSE	FALSE	FALSE	TRUE	boolean	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
string	FALSE	FALSE	FALSE	TRUE	boolean	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE

* never 可以赋给任何类型，但只能被 any 和 never 赋值
* unknown 只能赋给 any 和 unknown，但可以被任何类型赋值

组合类型与赋值兼容性

```
{ ... } extends { ... } ? ...

()=>... extends (...) => ... ? ...

signatures/list … extends ... ? ...
```

表达式类型与赋值兼容性

```
a|b|c extends a&b&c ? ...
a&b&c extends a|b|c ? ...
(A extends B ? ...) extends
    (M extends N ? ...) ? . : ..
```

总结

1. 赋值兼容通过“条件表达式”将结构兼容和子类型兼容统一在同一语义之下
   - 语法 `A extends B ? X : Y`
   - 兼容性不再是静态的约定，它可以被计算了，并且这个计算过程可以用于驱动计算流程（分支和递归）

2. 注意TypeScript在处理“裸类型参数”时会有一些潜规则
   - 元组的生成
   - keyof 的处理
   - never 和联合类型用在 extends 的左侧时的特殊性（上一讲）

作业

> 为 MapMartix<> 返回的二维元组生成csv或xls文件，然后在表格软件中分析该矩阵。

> 为 MapMartix<> 返回的二维元组添加首行和首列（标题行与标题列）。

```
// 提示1、向元组中添加新项的方法是使用展开运算（...），结果会是一个新元组
type X = [x, ...T]

// 提示2、新的泛型工具需要添加一个 类型约束，以使得元组在展开时通过类型检查。如下：
type MapMartix2<Source extends any[]> = {
  ...
}
```


极客时间
THANKS