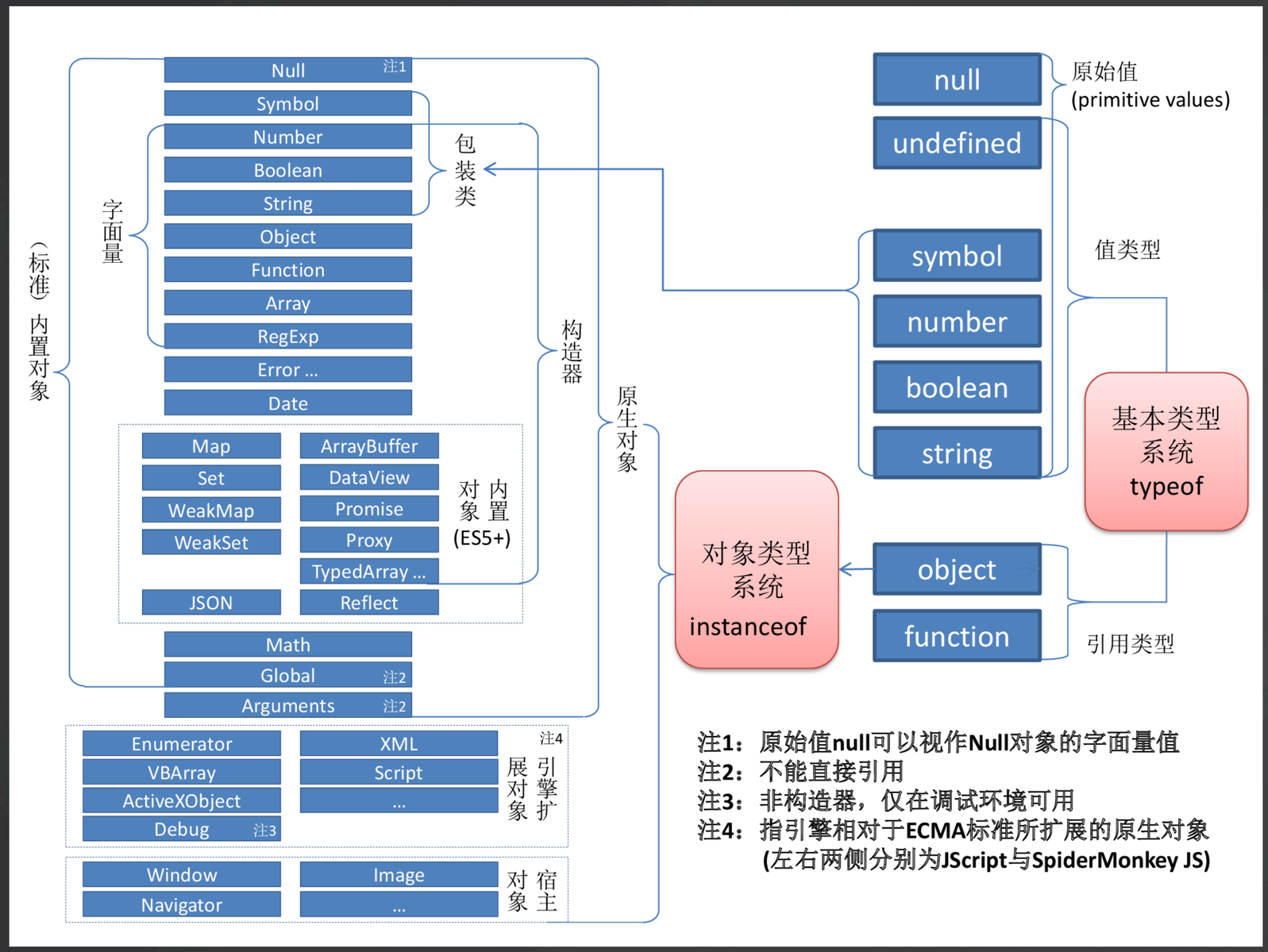
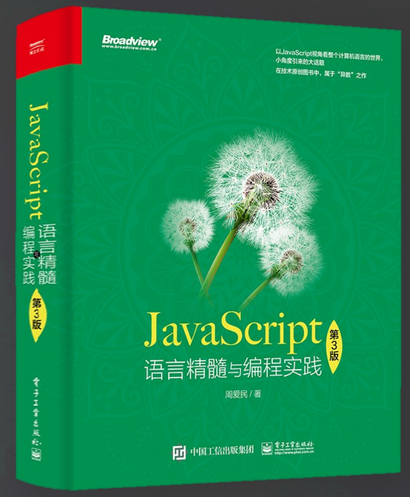
03 | TS与JS相关的那些类型

周爱民（Aimingoo）

目录



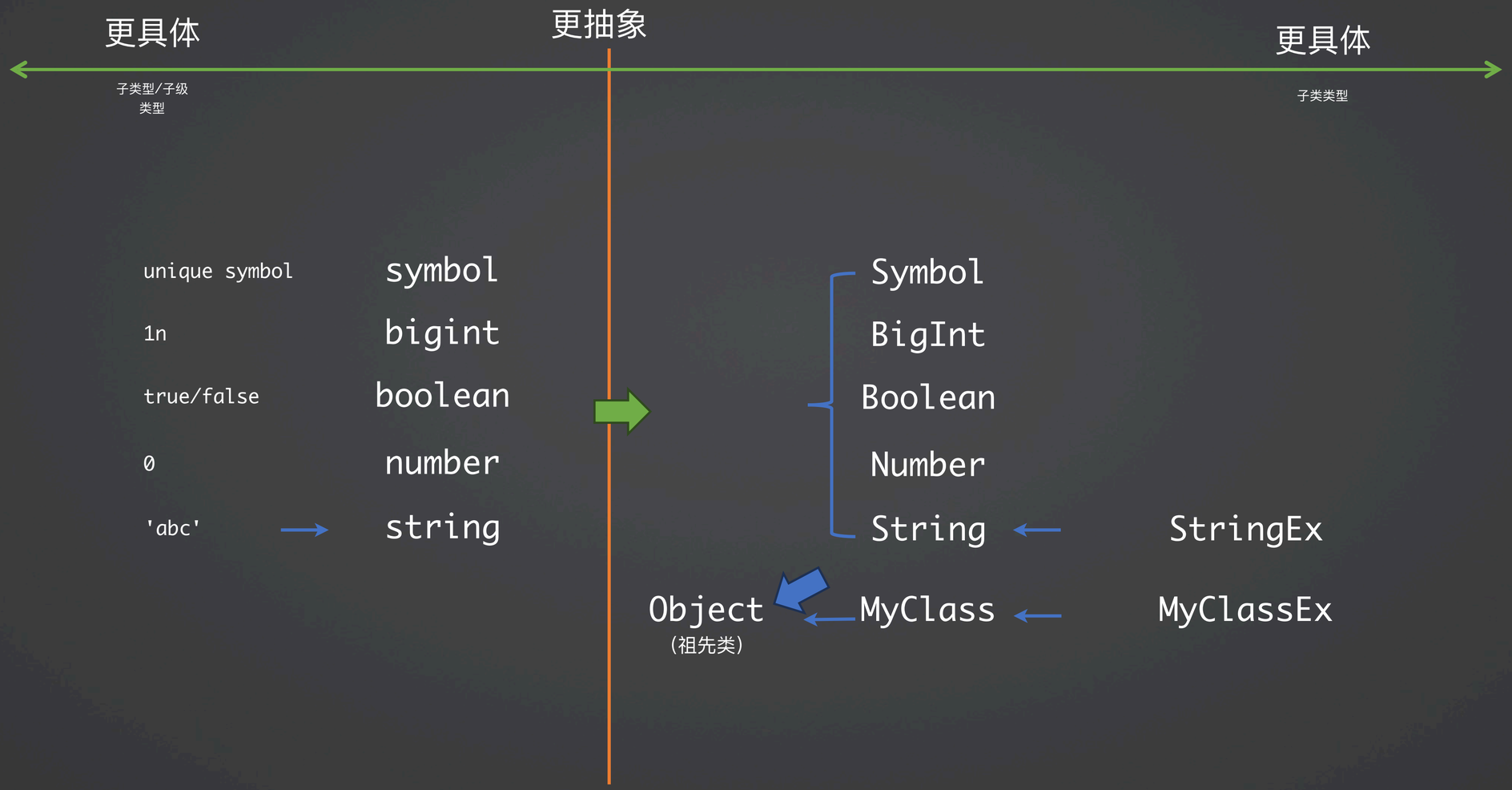


《JavaScript语言精髓与编程实践》

(3.4.4 内置的对象系统)

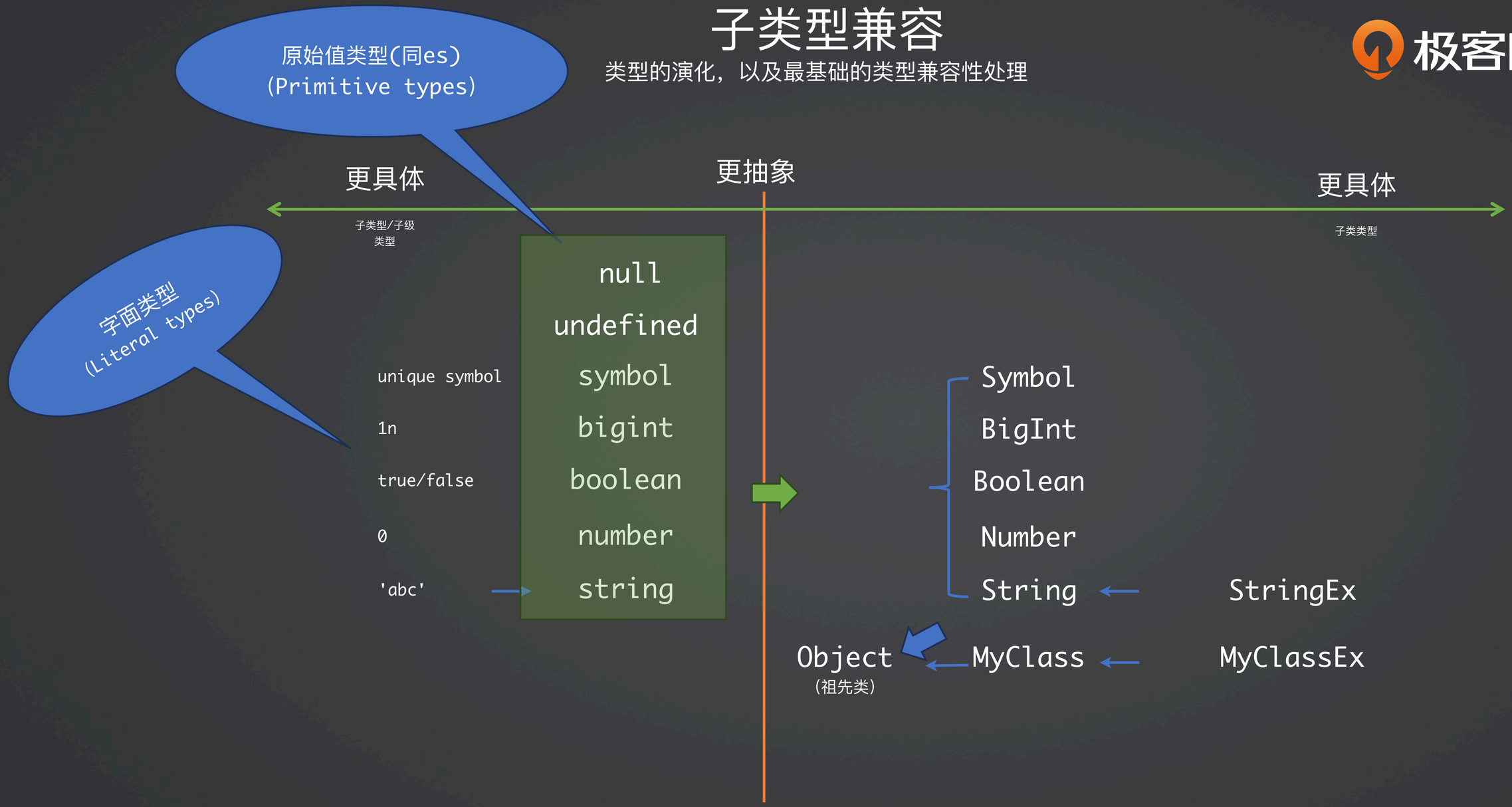
子类型兼容

类型的演化，以及最基础的类型兼容性处理



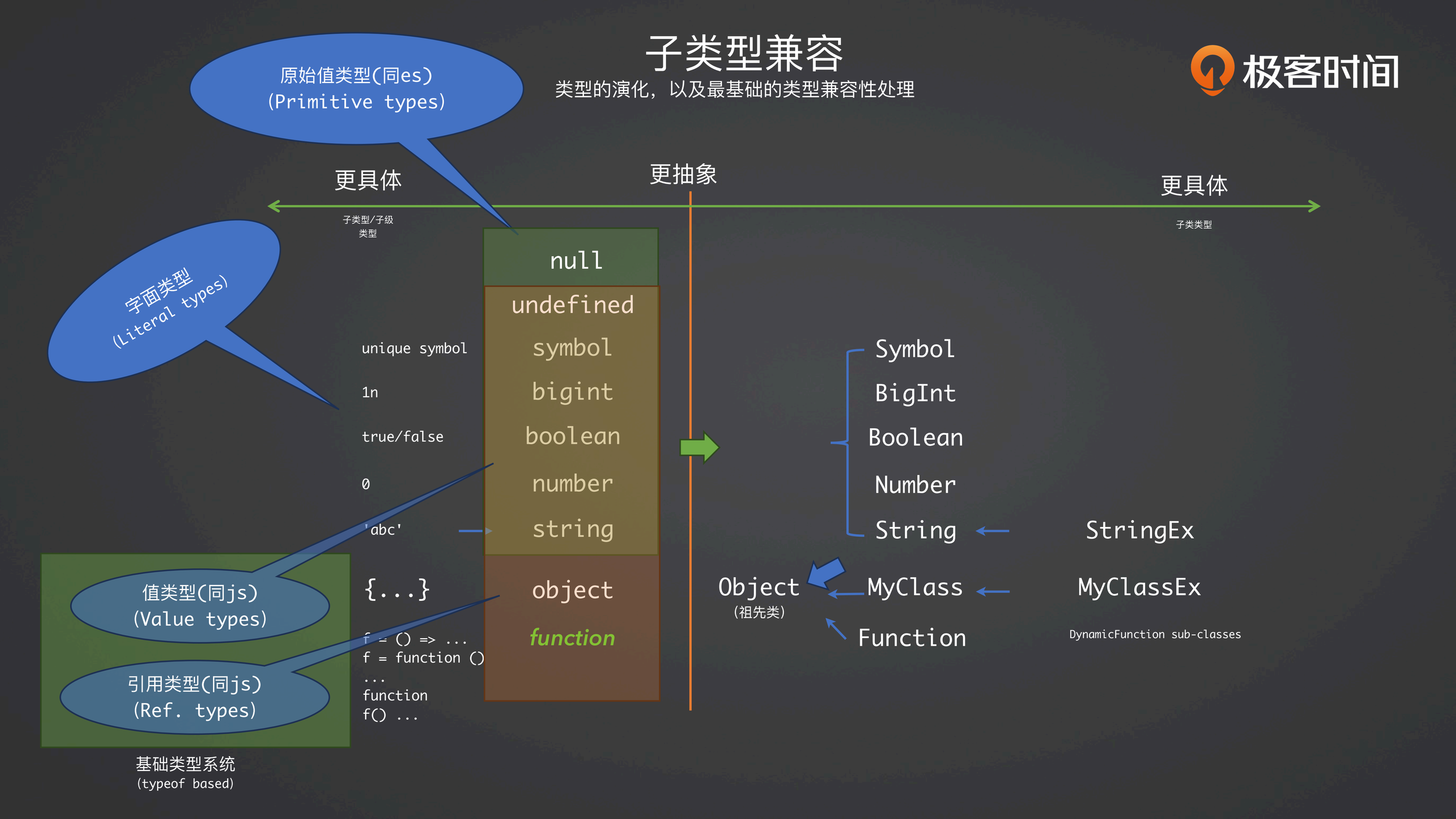
子类型兼容

类型的演化，以及最基础的类型兼容性处理



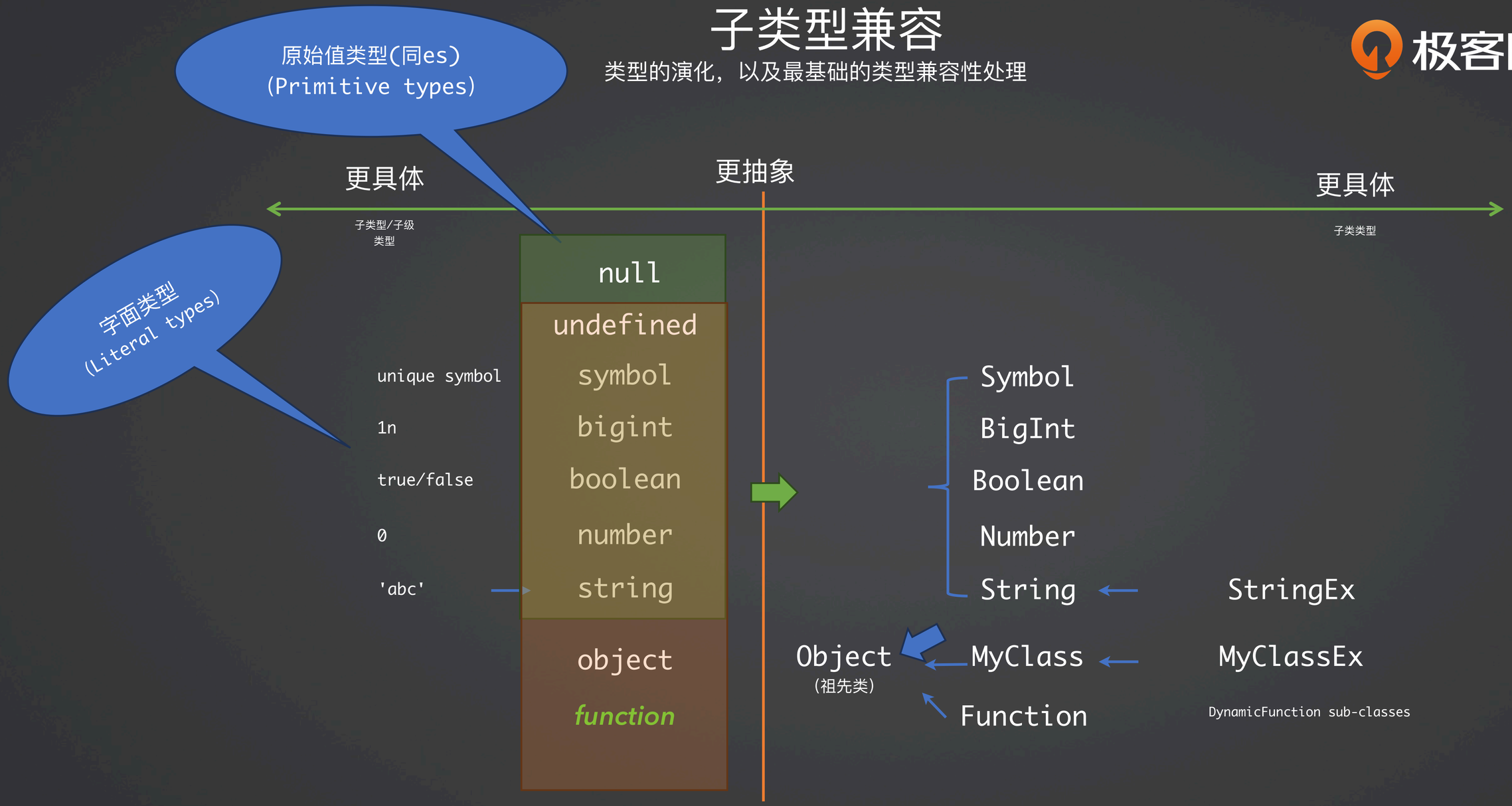
子类型兼容

类型的演化，以及最基础的类型兼容性处理



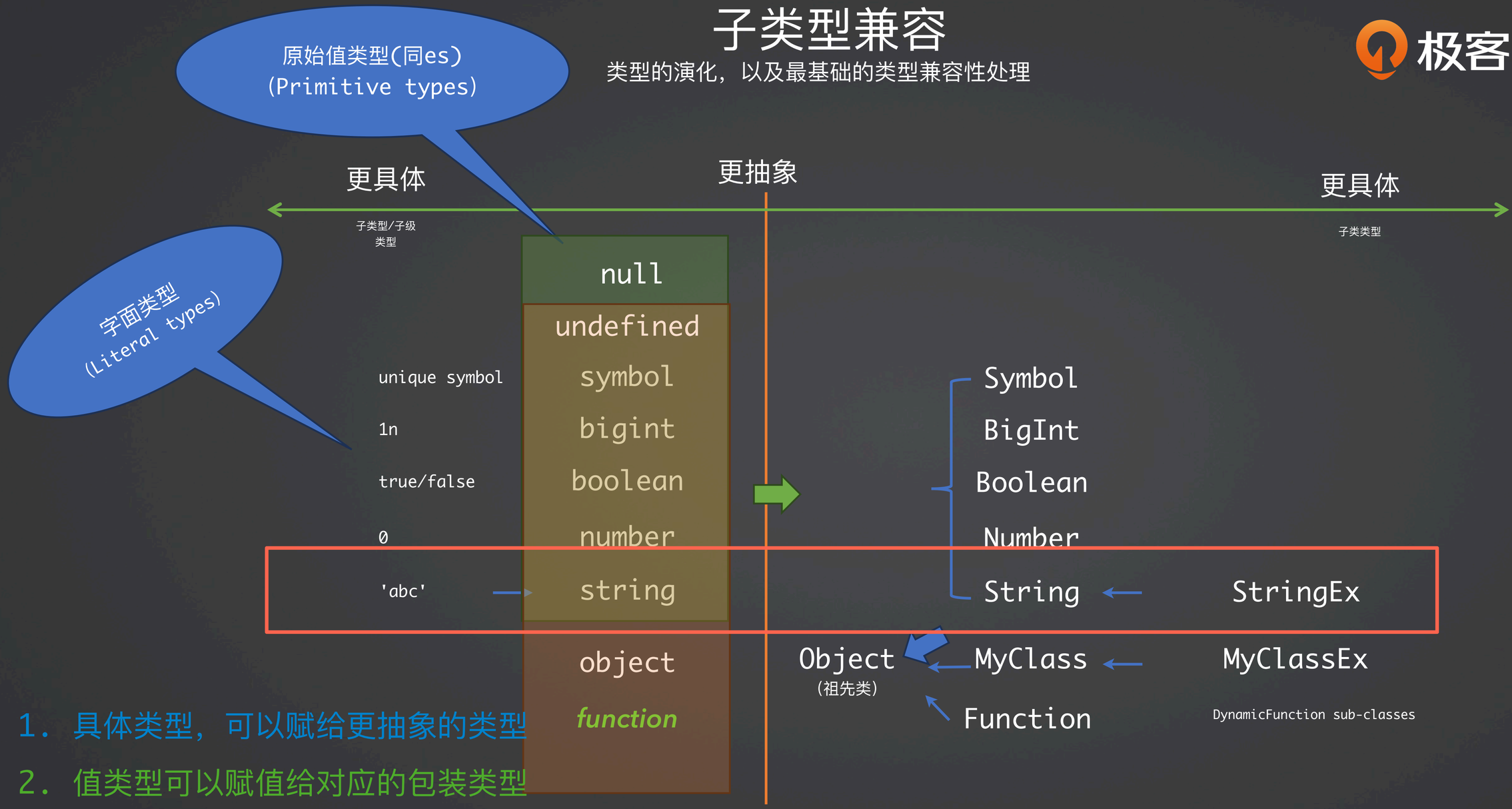
子类型兼容

类型的演化，以及最基础的类型兼容性处理

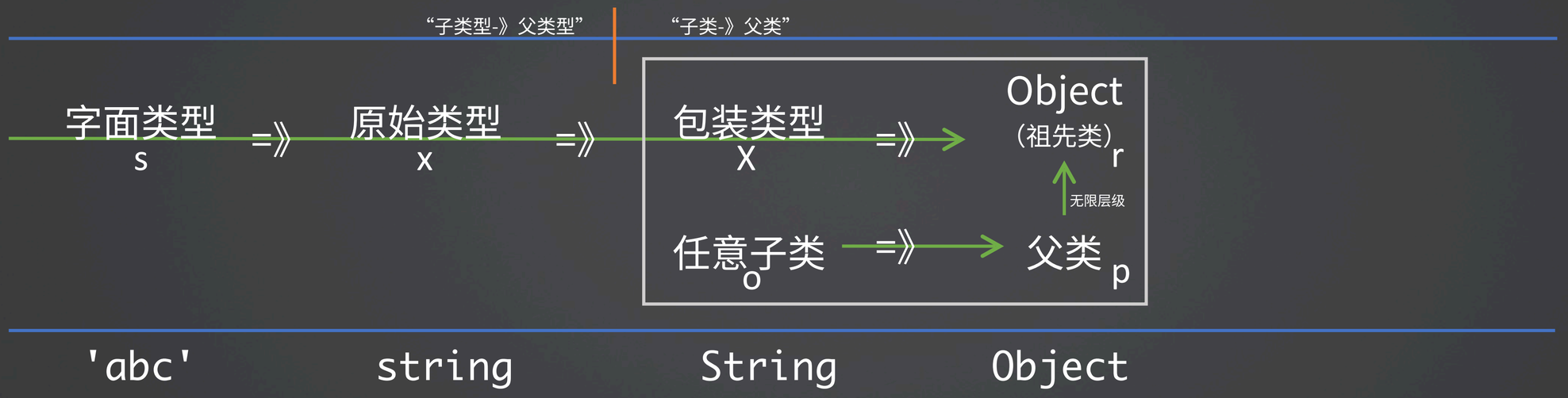


子类型兼容

类型的演化，以及最基础的类型兼容性处理

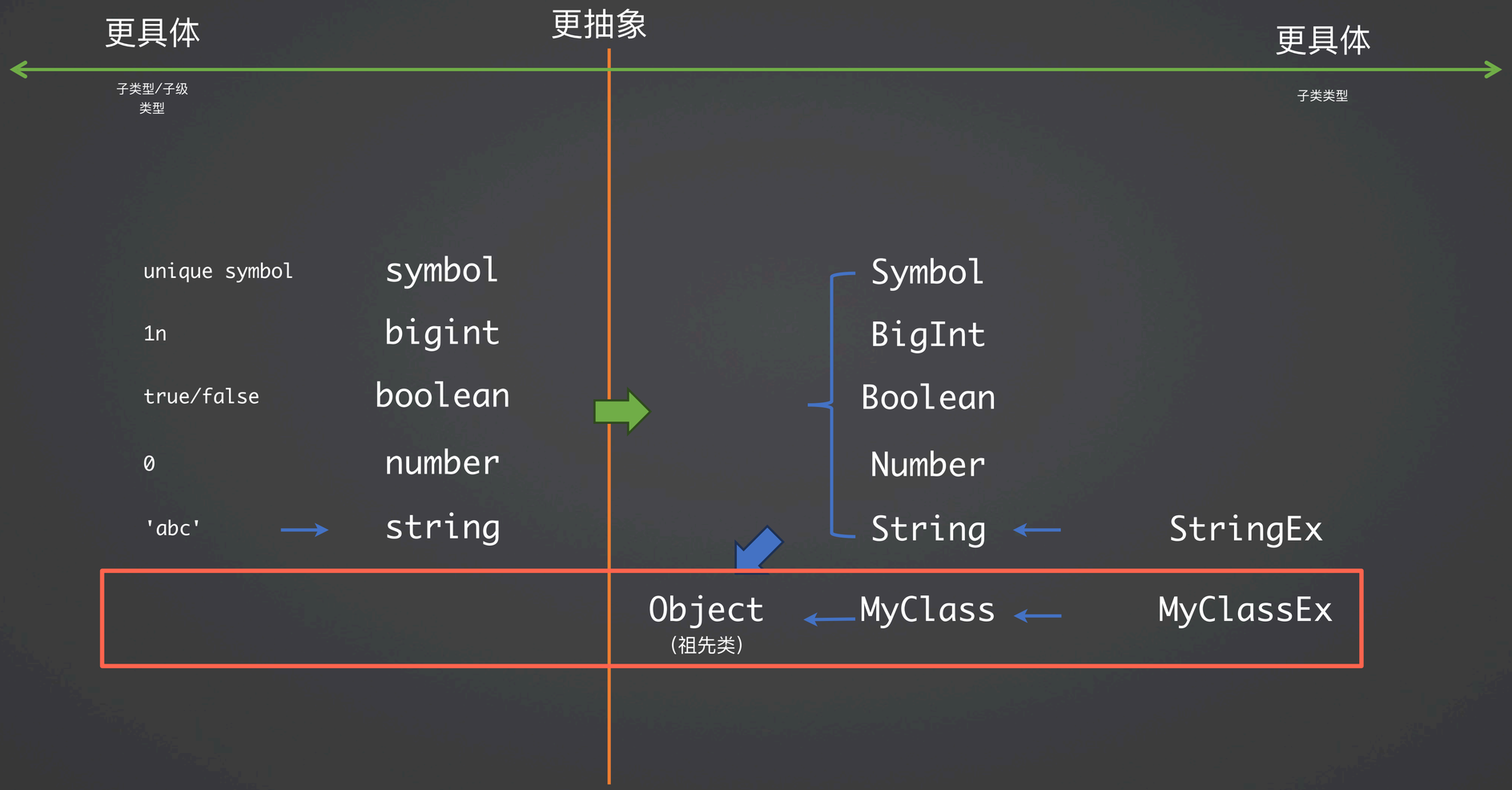


1. 具体类型，可以赋给更抽象的类型
2. 值类型可以赋值给对应的包装类型


“子类型-》父类型”
“子类-》父类”
字面类型
s
=》
原始类型
x
=》
包装类型
X
=》
Object
（祖先类）
r
无限层级
任意子类
o
=》
父类
p
'abc'
string
String
Object

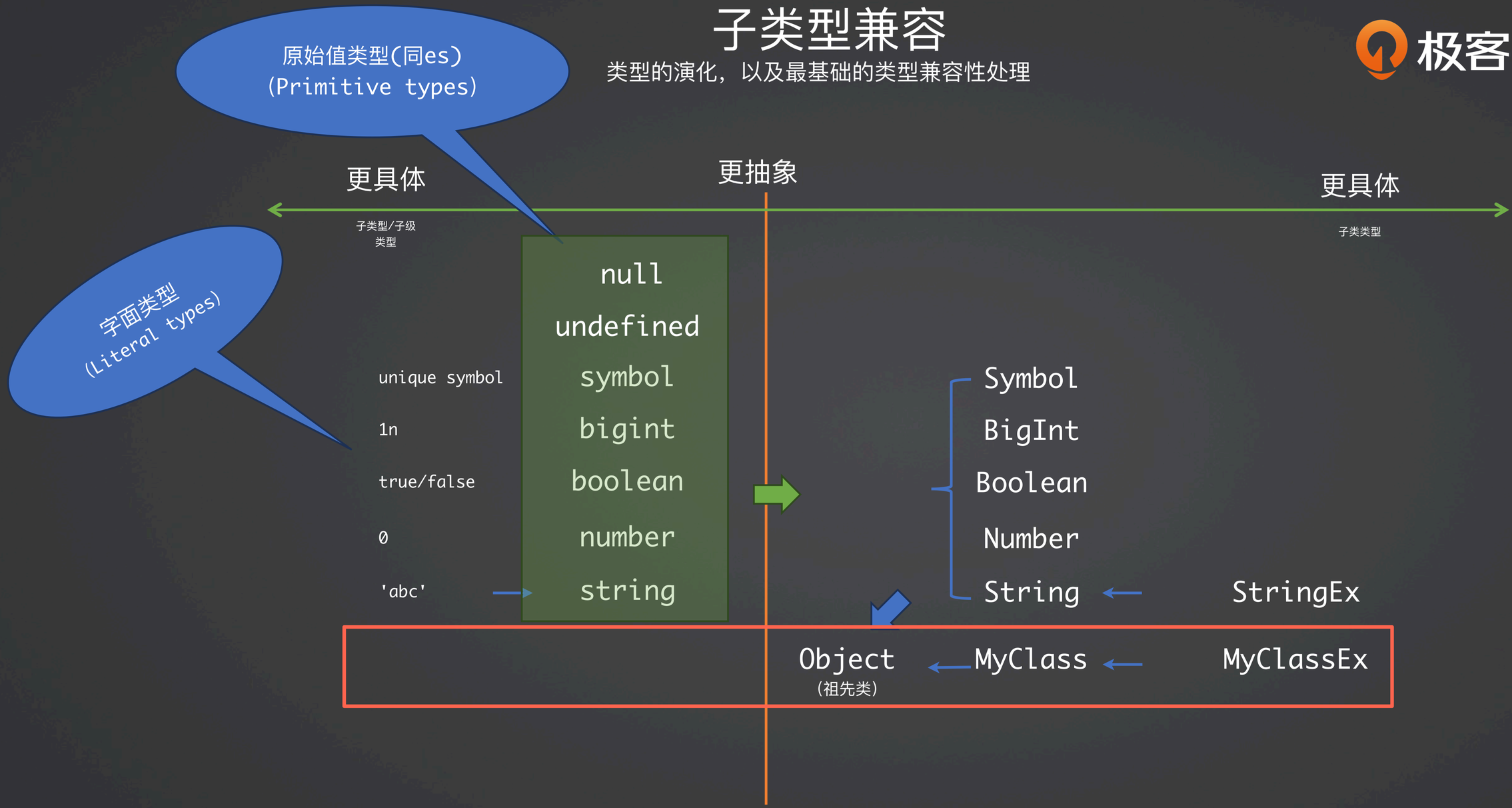
子类型兼容

类型的演化，以及最基础的类型兼容性处理



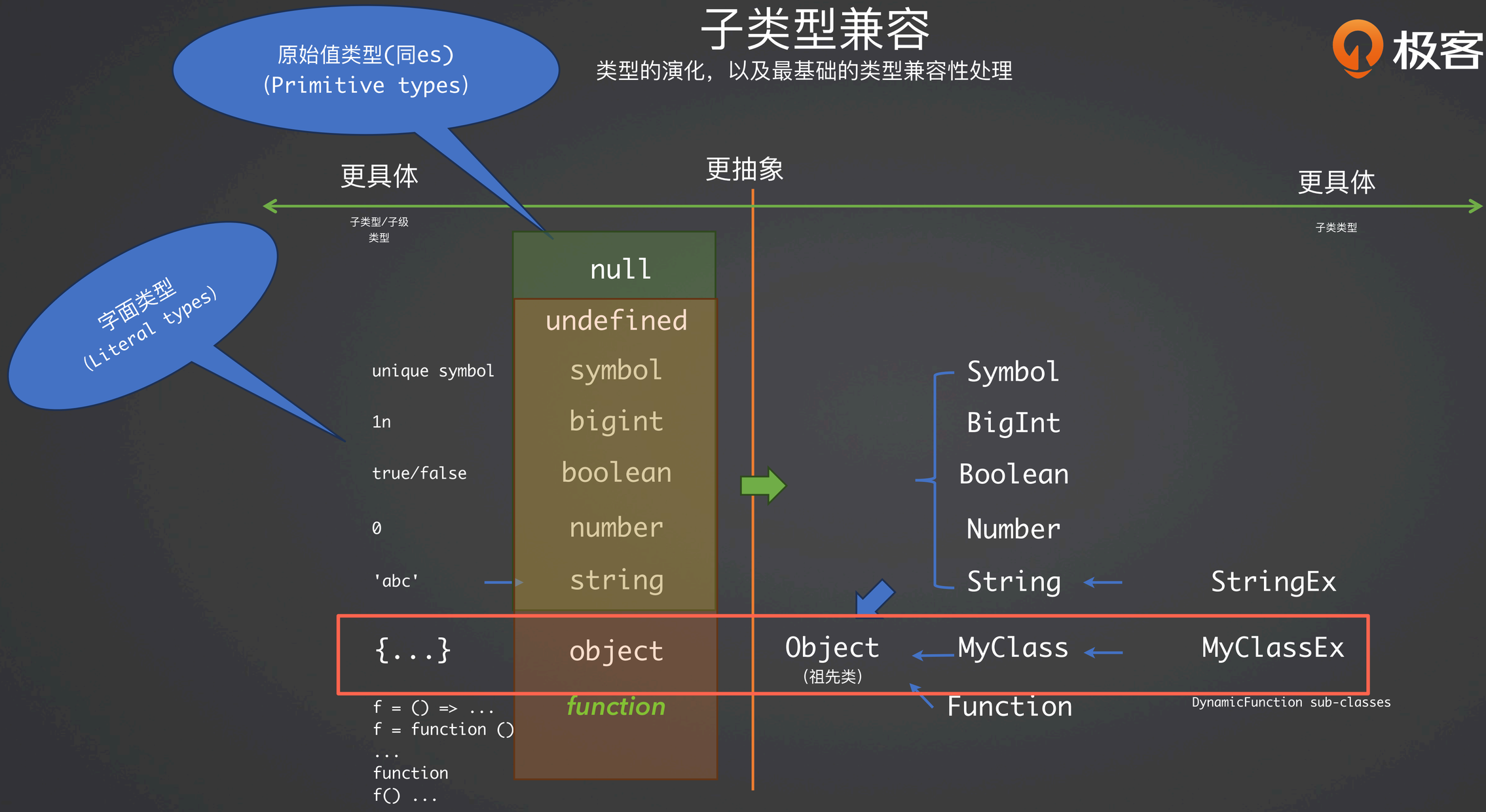
子类型兼容

类型的演化，以及最基础的类型兼容性处理



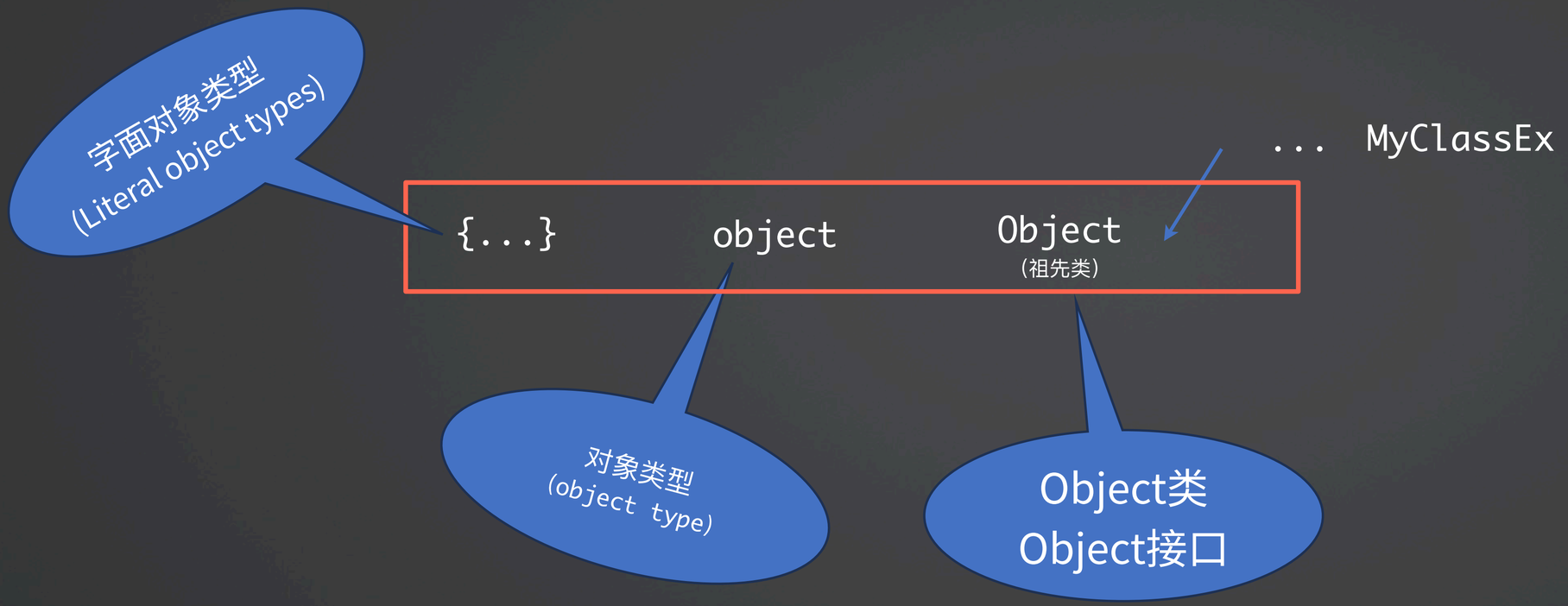
子类型兼容

类型的演化，以及最基础的类型兼容性处理



子类型兼容

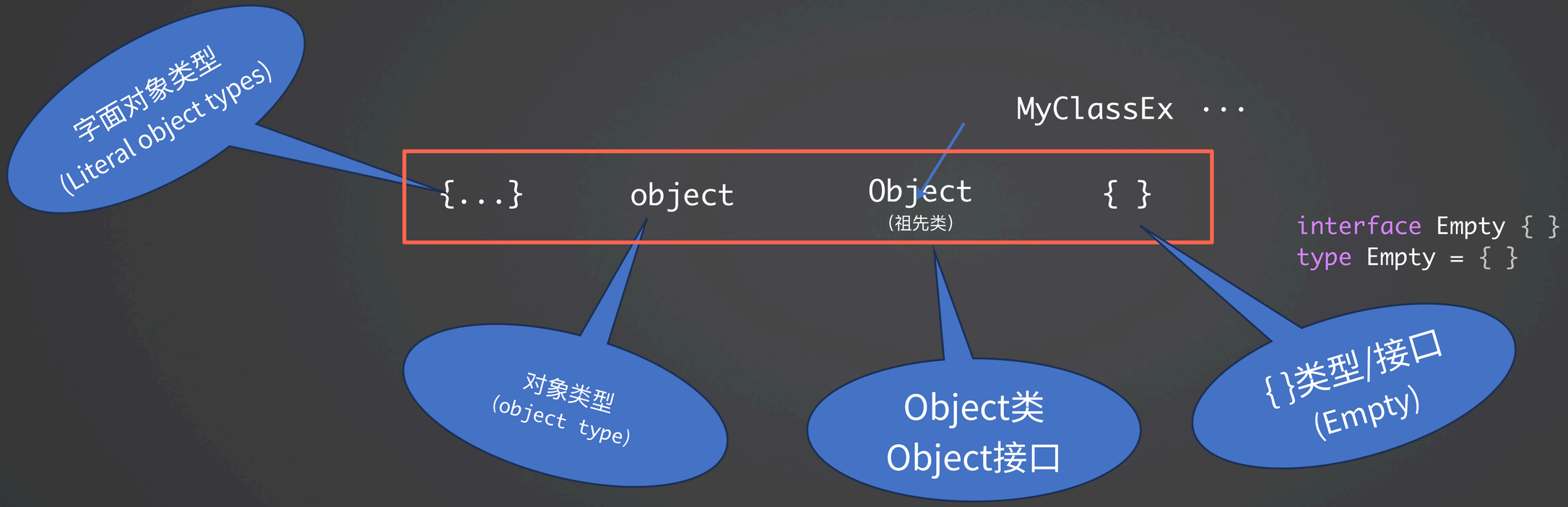
类型的演化，以及最基础的类型兼容性处理



3. 对象类型是类类型的子类型

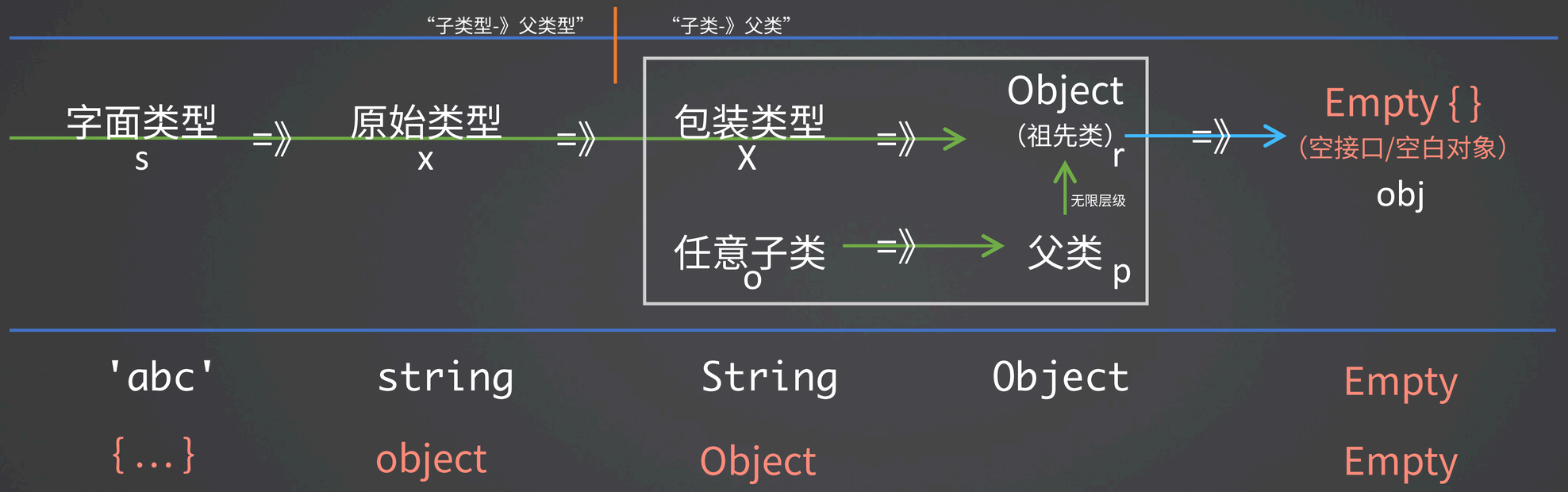
子类型兼容

类型的演化，以及最基础的类型兼容性处理




```
interface Empty { }
type Empty = { }
```

3. 对象类型是类类型的子类型
4. 结构类型的根是Empty


“子类型-》父类型”
“子类-》父类”
字面类型
s
=》
原始类型
x
=》
包装类型
X
=》
Object
（祖先类）
r
无限层级
任意子类
o
=》
父类
p
=》
Empty { }
（空接口/空白对象）
obj
'abc'
{ ... }
string
object
String
Object
Object
Empty
Empty

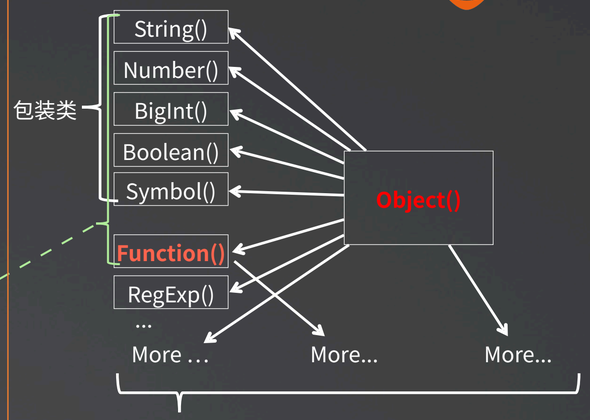
结构类型兼容

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)

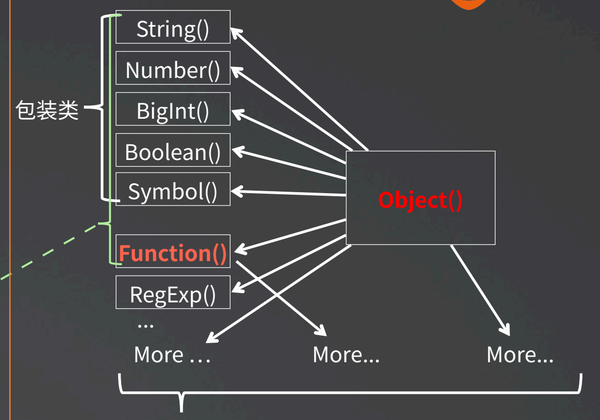
Interface define instance and/or class for instance, using one or more definitions.

JS类型 => TS类型

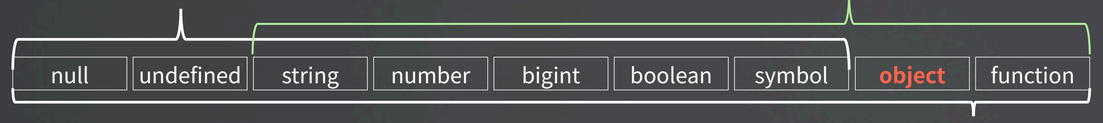
Literal style

Construct style
(with/without package classes)

极客时间



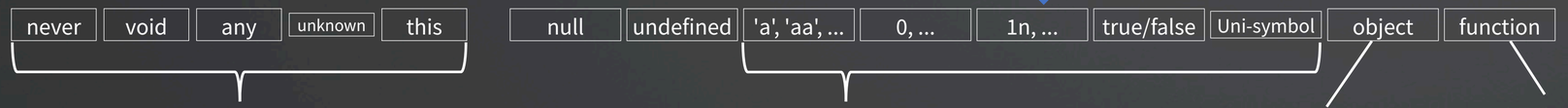
原始值类型(同es)
(Primitive types)



Above is collective type of literal types

基础类型系统
(typeof)

类/构造类型
(Class/Construct type)



特殊类型
(Special types)

字面类型
(Literal types)

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

对象类型
(object type)

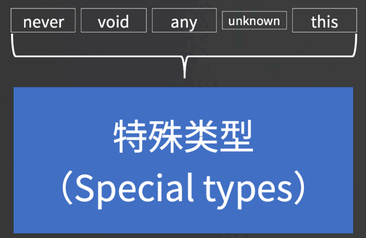
箭头函数: () => ...
具名函数: function f() { ... }
匿名函数: function () { ... }
...

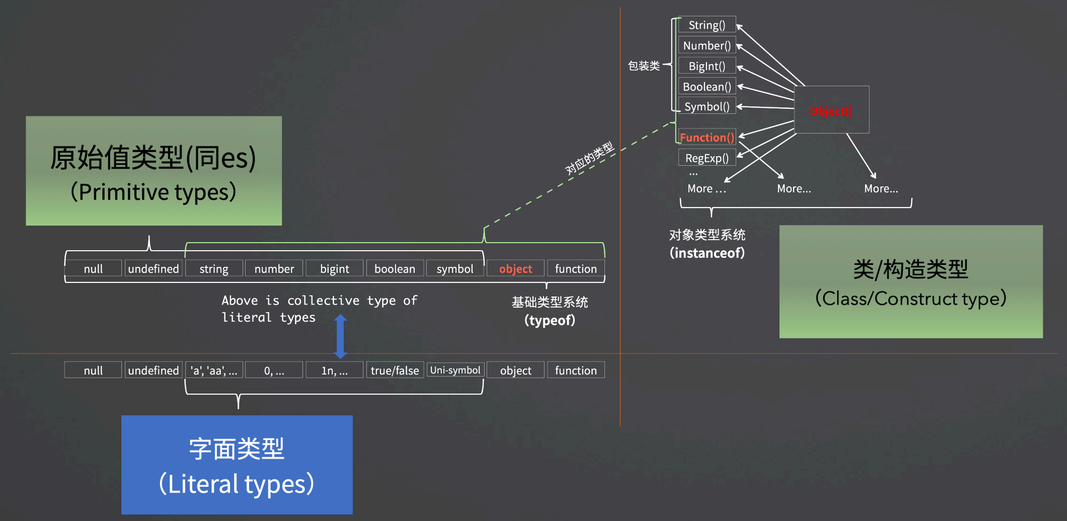
函数类型
(function type)

接口类型
(Interface type)

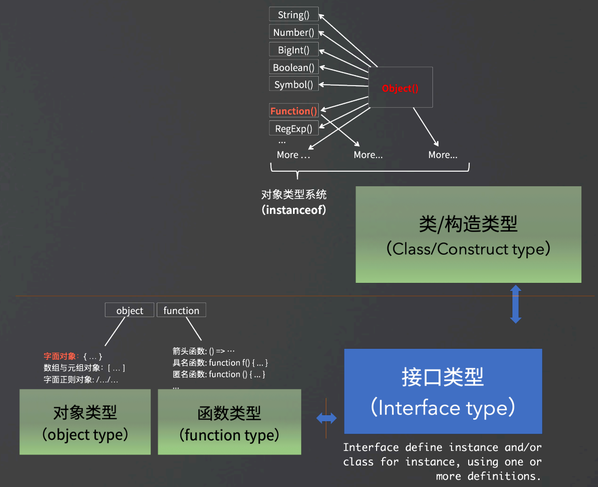
Interface define instance and/or class for instance, using one or more definitions.

赋值兼容





子类型兼容



结构类型兼容

名词、术语

字面类型、包装类型、包装类：Literal types, Wrapper types/Boxed types, Wrapper/Boxed class

原始值类型、对象类型、子面对象类型：Primitive types, Object types, Literal object types

值类型、引用类型：Value types, Reference types

子类、父类、祖先类：Child class(Subclass, derived class), Parent class(Superclass, base class), Ancestor class(root class)

子类型、父类型/超类型、子类型兼容：Subtype、Supertype、Subtype compatibility

结构类型、结构类型兼容：Structural types 、Structural type compatibility

接口、接口类型：Interface、Interface types

概念

- 在 JavaScript 中可以使用基于类的和基于原型的面向对象系统（*class-based and prototype-based object-oriented system*）。
- 在JavaScript中的类，实际上是基于原型的构造器的语法糖（*classes are really just syntactic sugar to prototype-based constructor functions*）。

  @ *https://www.typescriptlang.org/docs/handbook/decorators.html#class-decorators*
- 子类型是一种通过某些可替换性概念与其它数据类型（超类型）关联（*A subtype is a datatype that is related to another datatype (the supertype) by some notion of substitutability*）的数据类型，它是类型多态性的一种形式（*is therefore a form of type polymorphism*）。

  @ *https://en.wikipedia.org/wiki/Subtyping*
- 在TypeScript中的类型兼容性是基于结构子类型的（*is based on structural subtyping*）；结构类型兼容背后的思想是，如果多个类型的“成员类型是兼容的”，则他们是兼容的。

  @ *https://www.typescriptlang.org/docs/handbook/type-compatibility.html*

  @ *https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html#structural-type-system*

  @ *https://jkchao.github.io/typescript-book-chinese/faqs/type-system-behavior.html*
- 赋值兼容扩展了子类型兼容，以支持从“any赋值”、“赋值给any”（*assignment to and from `any`*），以及数字枚举成员与一般数值之间的赋值（*enum with corresponding numeric values*）。

  @ *https://www.typescriptlang.org/docs/handbook/type-compatibility.html#subtype-vs-assignment*

Q&A

Q：在哪里可以了解JavaScript的基本语法

A ： @see MDN https://developer.mozilla.org/zh-CN/docs/Web/JavaScript
@see ECMAScript Sepc., https://tc39.es/ecma262

Q：表达式、名字和值等等基础概念为什么与常见的语法书有所不同

A ： 这里主要参考的是ECMAScript中对这些基础概念的解释，与一般的语法书确实有些差别。另外也可以参考《JavaScript语言精髓与编程实践》之“4.2.2 表达式”来重新理解这些概念。

Q：为什么TypeScript手册中说只有两种兼容（子类型和赋值）

A ： 在手册中的“子类型”包含了结构子类型（Structural subtyping）和名义子类型（Nominal subtyping）两种。我们在本讲中讲的“子类型”是指名义子类型，即显式声明关联关系的类型系统。在这一语义下：1、子类型兼容包含了“字面类型与其对应的类型”、“子类与父类”、“接口与其父级接口”这三个大的类型间的兼容关系，此外也包含了“any与其它类型”，以及“枚举类型与枚举成员”之间的子类型兼容关系；2、结构类型兼容则特指那些不存在此类显式“父-子类型”声明时的兼容关系处理。

Q：结构类型与名义类型有什么不同?

A ： 在比较学术的讨论环境中，所谓结构类型（Structural typing）是直接比较类型之间的子成员的，亦即是直接考察类型的结构来解释多个类型之间的关联（Relationships）；而名义类型（Nominal typing）是考虑类型名之间是否通过某些规则来建立了关联，这种情况下，“用规则建立关联”是显性的，例如“声明子类A1派生自父类A”。基于上述定义，所谓的结构的或名义的子类型系统（Structural, or Nominal subtyping system），则是指强调以“父-子类型”作为上述“关联”的类型系统。

课后作业

1. 试解释如下代码为何是正确的

```
// class MyString extends String { }
let x: object = new MyString;
```

2. 说明如下两种声明有什么异同

```
const x: 'abc' = 'abc';
let x: 'abc' = 'abc';
```


极客时间
THANKS