

07 | 类的声明与使用

周爱民 (Aimingoo)

目录

- 1 JavaScript 中的类的写法
- 2 TypeScript 中类的继承、实现与属性修饰字，以及成员的初始化
- 3 TypeScript 中的类、抽象类和工厂方法类
- 4 总结

ECMAScript / JavaScript			TypeScript 2				
			private	protected	public	abstract	override
公开的 (默认)	构造方法	constructor()	Y 4	Y 5	Y	X(ts1042) 6	X(ts1089)
	存取器方法	get x() { ... }	Y	Y	Y	Y	Y
	方法	foo() { ... }	Y	Y	Y	Y	Y
	自动存取器	accessor x = 100	Y	Y	Y	Y	Y
	字段	x = 100	Y+	Y+	Y+ 3	Y	Y
私有的 (#) 1	存取器方法	get #x() { ... }	X	X	X	X	Y
	方法	#foo() { ... }	X	X	X	X	Y
	自动存取器	accessor #x = 100	X	X	X	X	Y
	字段	#x = 100	X	X	X	X	Y

字段 vs. 属性

1. 到现在为止，我们所讲的 TypeScript 语言特性都不涉及类的原型属性。


 2. “公开字段”与“使用 ``public`` 修饰字的公开属性”在语义和实现上是完全一致的。




 ▶ 可见性修饰字只能作用于字段与方法，不能作用于私有字段

3. “私有字段”与“私有属性”是不同的。

私有字段 vs. 私有属性

1. 私有字段是 ES/JS 原生支持的，不会被侵入

 2. 私有属性（等可见性）是 TS 在属性的基础上实现的，只有语法检查的意义，因此可被篡改。但在实现某些 TS 特性时不可或缺，例如：

-   属性参数
-  私有构造方法

总结

1. 继承 (`extends`)、实现 (`implements`)、覆盖 (`override`)

- ▶ 构造类型兼容 (实现) 与子类型兼容 (继承)
- ▶ 覆盖主要用于限制父类中必须存在同名方法

2. 可见性 (`private`、`protected`、`public`)

- ▶ 私有字段 vs. 私有属性
- ▶ 实例字段 vs. 原型存取器属性 (包括 `accessor` 关键字) vs. 原型属性 (下一讲补充)
- ▶ 构造参数 (constructor) 和私有构造方法 (`private` constructor)
- ▶ 抽象类与抽象方法 (`abstract`)、重载 (overload)

3. 其它修饰字与特殊类型 (`readonly`、`optional`、`this`)

- ▶ JS原生支持: `static`、`get`、`set`、`accessor`、`async`、`*`

名词/概念

名词、术语

可见性、继承性、多态性、封装: Visibility, Inheritance, Polymorphism, Encapsulation
成员、属性、字段、方法: Members, Properties, Fields, Methods
静态的、只读的、可选的、抽象的: Static, Readonly, Optional, Abstract
存取器（方法）、构造器（方法）: Accessor(get/setter), Constructor(construct)
私有字段、私有元素、私有名字: Private Fields, Private Element, Private Names
参数属性: Parameter Properties

概念

- JavaScript中的对象，是一个属性的集合（属性包，*a collection of properties*），每个属性要么是数据属性（data property），要么是存取器属性（accessor property）。私有字段是类声明或类表达式所在的环境中的私有名字，私有名字表达的是类的私有元素的键名（*represents the key of a private class element*），包括私有的字段、方法与存取器。需要强调的是，私有字段（以及所有的私有元素）不是对象的属性，它们通过名字关联到类的私有环境（Private Environment of class definition or class expression），后者用于存放这些私有名字。
@see: <https://tc39.es/ecma262/#sec-object-type>
@see: <https://tc39.es/ecma262/#sec-privateenvironment-records>
@see: <https://tc39.es/ecma262/#sec-private-names>
- 可见性（Visibility）是对象的性质在其继承关系上的表现的具体方法。
@see: 《程序原本》之“10.4 可见性同样也是多余的：它是对继承性的补充与展现”
@see: https://en.wikipedia.org/wiki/Object-oriented_programming
@see: <https://roadmap.sh/software-design-architecture/object-oriented-programming/paradigm-features/scope-visibility>
- 类成员缺省的可见性是“公开的（public）”，公开成员可以在任何地方访问（*be accessed anywhere*）。使用private声明的私有属性是软私有，不提供严格的隐私保护（*are soft private and don't strictly enforce privacy*），而使用#的私有字段是硬私有，在编译后仍然是私有的、不可被外部访问的（*remain private after compilation and ... making them hard private*）。
@see: <https://www.typescriptlang.org/docs/handbook/2/classes.html#member-visibility>
- 静态成员不与该类的任何特定实例关联（*aren't associated with a particular instance of the class*）。
@see: https://en.wikipedia.org/wiki/Type_system

Q&A

Q: 为什么在TypeScript中 `override` 能用于一般字段或属性?

A: 在一般的OOP概念中, `override`用于在子类中重写方法的接口, 这对子类中灵活设计方法的调用参数是很有用的。但是在TypeScript中, 子类与父类的兼容性是通过特定方法来检查的, 因此 `overrdie` 没有“改写接口”的意义。因而这个修饰字只用作“检查父类中是否存在同名成员”, 显然, 这对一般字段和属性来说也是适用的。

THANKS