双11返场进行中！优惠Go on，快来捡漏！ 02 天 15 时 09 分 19 秒 Go >>>

Sunday • 更新于 2022-10-19

上一节 13：框架实现：... 15：总结 下一节

14：reactive 函数的局限性

目前我们已经成功的完成了一个相对完善的 `reactive` 函数，通过它配合 `effect` 函数可以实现对应的响应性渲染。

但是对于 `reactive` 而言，它其实是具备一些局限性的。那么具体都有哪些局限性呢？

我们来思考以下两个问题：

1. `reactive` 可以对简单数据类型使用吗？比如：`reactive('张三')`
2. 当我们对 `reactive` 返回的响应性数据进行解构时，解构之后的属性还会具备响应性吗？

那么下面我们就对这两个问题，一个一个进行解释：

`reactive` 可以对简单数据类型使用吗？

我们知道，对于 `reactive` 函数而言，它会把传入的 **object** 作为 `proxy` 的 `target` 参数，而对于 `proxy` 而言，他只能代理 **对象**，而不能代理简单数据类型，所以说：**我们不可以使用 `reactive` 函数，构建简单数据类型的响应性。**

当我们对 `reactive` 返回的响应性数据进行解构时，解构之后的属性还会具备响应性吗？

一个数据是否具备响应性的关键在于：**是否可以监听它的 `getter` 和 `setter` **。而根据我们的代码可知，只有 `proxy` 类型的 **代理对象** 才可以被监听 `getter` 和 `setter` ，而一旦解构，对应的属性将不再是 `proxy` 类型的对象，所以：**解构之后的属性，将不具备响应性。**

总结与新的问题

那么到现在我们知道了， `reactive` 不可以对 **简单数据类型使用**，并且 **不可以解构**。那么如果我们期望 **简单数据类型也具备响应性**，那么我们又应该如何做呢？

熟悉 `vue 3` 的同学，肯定知道，此时我们可以使用 `ref` 函数来进行实现。

那么 `ref` 函数它又是因为什么可以构建简单数据类型的响应性，又为什么必须要通过 `.value` 访问数据呢？

想要知道以上问题，那么我们就继续往下看~~~~

13：框架实现：构建 Dep 模块，处理一对多... 上一节 下一节 15：总结

我要提出意见反馈

Copyright © 2022 imooc.com All Rights Reserved | 京ICP备 12003892号-11 京公网安备11010802030151号