慕课网首页 免费课 体系课 慕课教程 专栏 手记 企业服务 实战课

我的课程

Q

?

 \odot

从所有教程的词条中查询…

首页 > 慕课教程 > Vue3源码分析与构建方案 > 04: 总结: ref 复杂数据类型的响应性

全部开发者教程 Ξ

01: 前言

02: 源码阅读: ref 复杂数据 类型的响应性

03: 框架实现: ref 函数 - 构 建复杂数据类型的响应性

04: 总结: ref 复杂数据类型 的响应性

05: 源码阅读: ref 简单数据 类型的响应性

06: 框架实现: ref 函数 - 构 建简单数据类型的响应性

07: 总结: ref 简单数据类型 响应性

08: 总结

第七章:响应系统 computed && watch

01: 开篇

02: 源码阅读: computed 的 响应

03: 框架实现: 构建

Sunday • 更新于 2022-10-19

◆ 上一节 03: 框架实现: ... 05: 源码阅读: ... 下一节 →

索引目录

04: 总结: ref 复

04: 总结: ref 复杂数据类型的响应性

根据以上代码实现我们知道,针对于 ref 的复杂数据类型而言,它的响应性本身,其实是 利用 reacti ve 函数 进行的实现,即:

```
<>代码块
     const obj = ref({
         name: '张三'
2
3
     const obj = reactive({
         name: '张三'
7
```

本质上的实现方案其实是完全相同的,都是利用 reactive 函数,返回一个 proxy 实例,监听 proxy 的 getter 和 setter 进行的依赖收集和依赖触发。

但是它们之间也存在一些不同的地方,比如:

1. ref:

- 1. ref 的返回值是一个 RefImpl 类型的实例对象
- 2. 想要访问 ref 的真实数据,需要通过 .value 来触发 get value 函数,得到被 toReactive 标 记之后的 this._value 数据,即: proxy 实例

2. reactive:

1. reactive 会直接返回一个 proxy 的实例对象,不需要通过 .value 属性得到

同时我们也知道,对于 reactive 而言,它是不具备简单数据类型的响应性呢,但是 ref 是具备的。

那么 ref 是如何处理 简单数据类型 的响应性的呢? 我们继续来往下看~~~

03: 框架实现: ref 函数 - 构建复杂数据类型... ← 上一节 下一节 ▶ 05: 源码阅读: ref 简单数据类型的响应性

▶ 我要提出意见反馈

企业服务 网站地图 网站首页 关于我们 联系我们 讲师招募 帮助中心 意见反馈 代码托管

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





