

全部开发者教程

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 简单数据类型的响应性

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 07：总结：ref ... 01：开篇 下一节 ▶

08：总结

那么到这里我们就已经完成了 `ref` 响应性函数的构建，那么大家还记不记得开篇时所问的三个问题：

1. `ref` 函数是如何进行实现的呢？

2. `ref` 可以构建简单数据类型的响应性吗？

3. 为什么 `ref` 类型的数据，必须要通过 `.value` 访问值呢？

大家现在再次面对这三个问题，是否能够回答出来呢？

1. 问题一： `ref` 函数是如何进行实现的呢？

1. `ref` 函数本质上是生成了一个 `RefImpl` 类型的实例对象，通过 `get` 和 `set` 标记处理了 `value` 函数

2. 问题二： `ref` 可以构建简单数据类型的响应性吗？

1. 是的。 `ref` 可以构建简单数据类型的响应性

3. 问题三：为什么 `ref` 类型的数据，必须要通过 `.value` 访问值呢？

1. 因为 `ref` 需要处理简单数据类型的响应性，但是对于简单数据类型而言，它无法通过 `proxy` 建立代理。

2. 所以 `vue` 通过 `get value()` 和 `set value()` 定义了两个属性函数，通过 **主动** 触发这两个函数（属性调用）的形式来进行 **依赖收集** 和 **触发依赖**

3. 所以我们必须通过 `.value` 来保证响应性。

07：总结：ref 简单数据类型响应性 ◀ 上一节 下一节 ▶ 01：开篇

我要提出意见反馈

索引目录

08：总结

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

★ 微信 微博 抖音

意见反馈 收藏教程 标记书签