

全部开发者教程

三

载、更新行力

26: 总结

第十一章：runtime 运行时 - 组件的设计原理与渲染方案

01: 前言

02: 源码阅读：无状态基础组件挂载逻辑

03: 框架实现：完成无状态基础组件的挂载

04: 源码阅读：无状态基础组件更新逻辑

05: 局部总结：无状态组件的挂载、更新、卸载总结

06: 源码阅读：有状态的响应性组件挂载逻辑

07: 框架实现：有状态的响应性组件挂载逻辑

08: 源码阅读：组件生命周期回调处理逻辑

09: 框架实现：组件生命周期回调处理逻辑



Sunday • 更新于 2022-10-19

◀ 上一节 04：源码阅读：无状态基础组件更新逻辑 06：源码阅读：有状态的响应性组件挂载逻辑 下一节 ▶

05：局部总结：无状态组件的挂载、更新、卸载总结

那么到现在我们已经完成了 无状态组件的挂载、更新、卸载 操作。

从以上的内容中我们可以发现：

1. 所谓组件渲染，本质上指的是 `render` 函数返回值的渲染

2. 组件渲染的过程中，会生成 `ReactiveEffect` 实例 `effect`

3. 额外还存在一个 `instance` 的实例，该实例表示 组件本身，同时 `vnode.component` 指向它

4. 组件本身额外提供了很多的状态，比如： `isMounted`

但是以上的内容，全部都是针对于 无状态 组件来看的。

在我们的实际开发中，组件通常是 有状态（即：存在 `data` 响应性数据）的，那么有状态的组件和无状态组件他们之间的渲染存在什么差异呢？让我们继续来往下看。

04：源码阅读：无状态基础组件更新逻辑 ◀ 上一节 下一节 ▶ 06：源码阅读：有状态的响应性组件挂载逻辑

✎ 我要提出意见反馈

索引目录

05：局部总结：无状态组件的挂载、更新、卸载总结

💬

?

📱

😊