

全部开发者教程

☰

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

10：源码阅读：生命回调钩子中访问响应性数据

11：框架实现：生命回调钩子中访问响应性数据

12：源码阅读：响应性数据改变，触发组件的响应性变化

13：框架实现：响应性数据改变，触发组件的响应性变化

14：源码阅读：composition API，setup 函数挂载逻辑



Sunday • 更新于 2022-10-19

◀ 上一节 10：源码阅读：生命回调钩子中访问响应性数据

12：源码阅读：响应性数据改变，触发组件的响应性变化

下一节 ▶

11：框架实现：生命回调钩子中访问响应性数据

根据上一小节的描述，我们只需要 改变生命周期钩子的 this 指向即可

1. 在 packages/runtime-core/src/component.ts 中为 callHook 方法增加参数，以此来改变 this 指向：

<> 代码块

```
1  /**
2   * 触发 hooks
3   */
4   function callHook(hook: Function, proxy) {
5     hook.bind(proxy)()
6   }
```

2. 在 applyOptions 方法中为 callHook 的调用，传递第二个参数：

<> 代码块

```
1  // hooks
2  if (beforeCreate) {
3    callHook(beforeCreate, instance.data)
4  }
5
6  ...
7
8  // hooks
9  if (created) {
10    callHook(created, instance.data)
11  }
```

3. 在 registerLifecycleHook 中，为 hook 修改 this 指向

<> 代码块

```
1  function registerLifecycleHook(register: Function, hook?: Function) {
2    register(hook?.bind(instance.data), instance)
3  }
```

至此，代码完成。

创建对应测试实例 packages/vue/examples/runtime/redner-component-hook-data.html：

<> 代码块

```
1  <script>
2    const { h, render } = Vue
3
4    const component = {
5      data() {
6        return {
7          msg: 'hello component'
8        }
9      },
10     render() {
```

📝 意见反馈

📖 收藏教程

🔖 标记书签



```
13      // 组件实例处理完所有与状态相关的选项之后
14      created() {
15          console.log('created', this.msg);
16      },
17      // 组件被挂载之后
18      mounted() {
19          console.log('mounted', this.msg);
20      },
21  }
22
23  const vnode = h(component)
24  // 挂载
25  render(vnode, document.querySelector('#app'))
26  </script>
```

数据访问成功

10: 源码阅读: 生命回调钩子中访问响应性... ◀ 上一节      下一节 ▶ 12: 源码阅读: 响应性数据改变, 触发组件...

 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



 意见反馈

 收藏教程

 标记书签