

 Sunday • 更新于 2022-10-19

接下来我们来看场景四 **旧节点多于新节点时**，根据场景三的经验，其实我们也可以明确，对于旧节点多于新节点时，对应的场景也可以细分为两种：

1. 多出的旧节点位于 **尾部**
2. 多出的旧节点位于 **头部**

那么我们先创建第一种场景下的测试实例 `packages/vue/examples/imooc/runtime/render-element-diff-4.html` :

```
1 <script>
2   const { h, render } = Vue
3
4   const vnode = h('ul', [
5     h('li', {
6       key: 1
7     }, 'a'),
8     h('li', {
9       key: 2
10    }, 'b'),
11    h('li', {
12      key: 3
13    }, 'c'),
14  ])
15  // 挂载
16  render(vnode, document.querySelector('#app'))
17
18  // 延迟两秒，生成新的 vnode，进行更新操作
19  setTimeout(() => {
20    const vnode2 = h('ul', [
21      h('li', {
22        key: 1
23      }, 'a'),
24      h('li', {
25        key: 2
26      }, 'b'),
27    ])
28    render(vnode2, document.querySelector('#app'))
29  }, 2000);
30 </script>
```

跟踪代码，直接进入场景四即可：

1. 代码进入场景四，此时的参数为：

```
▶ c1: (3) [{...}, {...}, {...}]
▶ c2: (2) [{...}, {...}]
▶ container: ul
  e1: 2
  e2: 1
  i: 2
  isSVG: false
  l2: 2
```

2. 满足  $i > e2$

3. 满足 while 循环  $i \leq e1$  的判断

1. 直接执行 `unmount(c1[i], parentComponent, parentSuspense, true)` 方法，对旧节点进行 **卸载** 操作

2. 执行 `i++`

以上代码比较简单，对于 多出的旧节点位于 **头部** 的场景，同样执行该逻辑。

由以上代码可知：

1. 旧节点多于新节点时，整体的处理比较简单，只需要 **卸载旧节点即可**

08: 框架实现：场景三：新节点多余旧节点... ◀ 上一节      下一节 ▶ 10: 框架实现：场景四：旧节点多于新节点...

 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