Sunday • 更新于 2022-10-19

上一节 08：框架实现：... 10：框架实现：... 下一节

09：源码阅读：场景四：旧节点多于新节点时的 diff 比对

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` ：

代码块

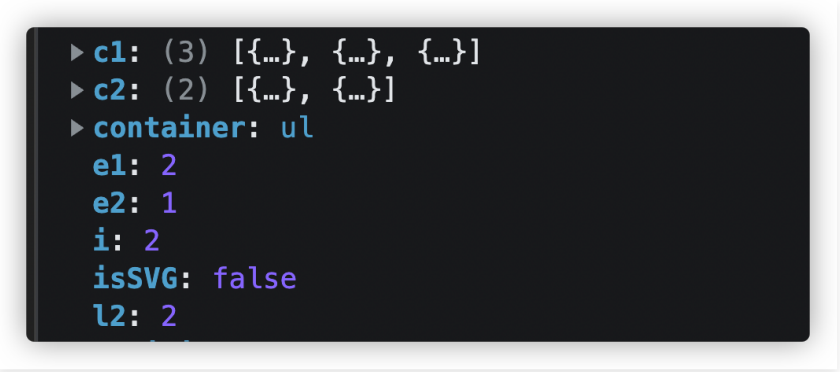
```
<script>
  const { h, render } = Vue

  const vnode = h('ul', [
    h('li', {
      key: 1
    }, 'a'),
    h('li', {
      key: 2
    }, 'b'),
    h('li', {
      key: 3
    }, 'c'),
  ])
  // 挂载
  render(vnode, document.querySelector('#app'))

  // 延迟两秒，生成新的 vnode，进行更新操作
  setTimeout(() => {
    const vnode2 = h('ul', [
      h('li', {
        key: 1
      }, 'a'),
      h('li', {
        key: 2
      }, 'b'),
    ])
    render(vnode2, document.querySelector('#app'))
  }, 2000);
</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：



2. 满足 `i > e2`
3. 满足 `while` 循环 `i <= 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