

全部开发者教程

04：框架实现：场景一：自前向后的 diff 对比

05：源码阅读：场景二：自后向前的 diff 对比

06：框架实现：场景二：自后向前的 diff 对比

07：源码阅读：场景三：新节点多余旧节点时的 diff 比对

08：框架实现：场景三：新节点多余旧节点时的 diff 比对

09：源码阅读：场景四：旧节点多于新节点时的 diff 比对

10：框架实现：场景四：旧节点多于新节点时的 diff 比对

11：局部总结：前四种 diff 场景的总结与乱序场景

12：前置知识：场景五：最长递增子序列

13：源码逻辑：场景五：求解最长递增子序列



Sunday • 更新于 2022-10-19

◀ 上一节 07：源码阅读：... 09：源码阅读：... 下一节 ▶

## 08：框架实现：场景三：新节点多余旧节点时的 diff 比对

根据上一小节的分析，我们可以直接在 `packages/runtime-core/src/renderer.ts` 中的 `patchKeyedChildren` 方法下，实现如下代码：

<> 代码块

```
1 // 3. 新节点多余旧节点时的 diff 比对。
2 if (i > oldChildrenEnd) {
3   if (i <= newChildrenEnd) {
4     const nextPos = newChildrenEnd + 1
5     const anchor =
6       nextPos < newChildrenLength ? newChildren[nextPos].el : parentAnchor
7     while (i <= newChildrenEnd) {
8       patch(null, normalizeVNode(newChildren[i]), container, anchor)
9       i++
10    }
11  }
12 }
```

创建对应测试实例 `packages/vue/examples/runtime/render-element-diff-3.html`：

<> 代码块

```
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  ])
12   // 挂载
13   render(vnode, document.querySelector('#app'))
14
15   // 延迟两秒，生成新的 vnode，进行更新操作
16   setTimeout(() => {
17     const vnode2 = h('ul', [
18       h('li', {
19         key: 3
20       }, 'c'),
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>
```

测试成功

索引目录

08：框架实现：场



