Sunday • 更新于 2022-10-19

上一节 14：源码阅读：... 16：总结 下一节

15：框架实现：场景五：乱序下的 diff 比对

那么明确好了对应的概念之后，我们下面就可以实现对应逻辑了，因为咱们要实现的代码，本质上和 vue 中的代码一致，所以我们没有必要再重头写一遍了，我们只需要**复制 vue 中的源码，然后修改一下变量名** 即可

1. 在 patchKeyedChildren 中，添加场景五乱序逻辑：

代码块

```
// 5. 乱序的 diff 比对
else {
    const oldStartIndex = i
    const newStartIndex = i
    const keyToNewIndexMap = new Map()
    for (i = newStartIndex; i <= newChildrenEnd; i++) {
        const nextChild = normalizeVNode(newChildren[i])
        if (nextChild.key != null) {
            keyToNewIndexMap.set(nextChild.key, i)
        }
    }

    let j
    let patched = 0
    const toBePatched = newChildrenEnd - newStartIndex + 1
    let moved = false
    let maxNewIndexSoFar = 0
    const newIndexToOldIndexMap = new Array(toBePatched)
    for (i = 0; i < toBePatched; i++) newIndexToOldIndexMap[i] = 0
    for (i = oldStartIndex; i <= oldChildrenEnd; i++) {
        const prevChild = oldChildren[i]
        if (patched >= toBePatched) {
            unmount(prevChild)
            continue
        }
        let newIndex
        if (prevChild.key != null) {
            newIndex = keyToNewIndexMap.get(prevChild.key)
        }

        if (newIndex === undefined) {
            unmount(prevChild)
        }
        else {
            newIndexToOldIndexMap[newIndex - newStartIndex] = i + 1
            if (newIndex >= maxNewIndexSoFar) {
                maxNewIndexSoFar = newIndex
            } else {
                moved = true
            }
            patch(prevChild, newChildren[newIndex], container, null)
            patched++
        }
    }

    const increasingNewIndexSequence = moved
```

```
        j = increasingNewIndexSequence.length - 1
        for (i = toBePatched - 1; i >= 0; i--) {
            const nextIndex = newStartIndex + i
            const nextChild = newChildren[nextIndex]
            const anchor =
                nextIndex + 1 < newChildrenLength
                    ? newChildren[nextIndex + 1].el
                    : parentAnchor
            if (newIndexToOldIndexMap[i] === 0) {
                patch(null, nextChild, container, anchor)
            } else if (moved) {
                if (j < 0 || i !== increasingNewIndexSequence[j]) {
                    move(nextChild, container, anchor)
                } else {
                    j--
                }
            }
        }
    }
```

2. 新增 `move` 方法：

代码块

```
    /**
     * 移动节点到指定位置
     */
    const move = (vnode, container, anchor) => {
        const { el } = vnode
        hostInsert(el!, container, anchor)
    }
```

至此，场景五的逻辑完成。

可以创建对应测试实例 `packages/vue/examples/imooc/runtime/render-element-diff-5.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'),
    h('li', {
      key: 4
    }, 'd'),
    h('li', {
      key: 5
    }, 'e')
  ])
  // 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
      }, 'new-a'),
      h('li', {
        key: 3
      }, 'new-c'),
      h('li', {
```

```
35          }, 'new-b'),
36          h('li', {
37            key: 6
38          }, 'new-f'),
39          h('li', {
40            key: 5
41          }, 'new-e'),
42        ])
43        render(vnode2, document.querySelector('#app'))
44      }, 2000);
45    </script>
```

测试成功

14：源码阅读：场景五：乱序下的 diff 比对 ◂ 上一节　　下一节 ▸ 16：总结