Sunday • 更新于 2022-10-19



# 02：前置知识：VNode 虚拟节点 key 属性的作用

再去学习 `diff` 之前，我们需要先了解一个前置的知识点。

我们知道在 `v-for` 循环的时候，我们必须要指定一个 `key` 值。那么这个 `key` 值的作用是什么呢？

我们之前写过一个方法： `packages/runtime-core/src/vnode.ts` 中的 `isSameVNodeType` 方法。

该方法是用来判断两个 `VNode` 是否为相同 `VNode` 的方法：

代码块

```
/**
 * 根据 key || type 判断是否为相同类型节点
 */
export function isSameVNodeType(n1: VNode, n2: VNode): boolean {
    return n1.type === n2.type && n1.key === n2.key
}
```

在这个代码中， `type` 我们知道什么意思。但是这个 `key` 是干什么呢的？

这里的 `key` 是一个 `props` ，比如我们可以在 `vue` 源码中，利用 `h` 函数，构建这样的一个 `VNode` ：

代码块

```
const vnode = h('li', {
  key: 1,
}, '1')
```

`VNode` 的值为：

代码块

```
{
  "__v_isVNode": true,
  "type": "li",
  "props": { "key": 1 },
  "key": 1,
  "children": "1",
  "shapeFlag": 9,
  ....
}
```

此时我们可以发现， `VNode` 中多出了一个 `key` 的属性，值为 `1` 。

如果我们通过 `isSameVNodeType` 方法，判断两个 `VNode` 是否为 **相同的** `VNode` ，则只需要保证两个 `VNode` 的 `type` 和 `key` 相同即可：

代码块

```
const vnode = h('li', {
  key: 1,
}, '1')
const vnode2 = h('li', {
  key: 1,
}, '2')
isSameVNodeType(vnode, vnode2) // true
```

这个概念在 `diff` 中非常重要，它决定了 `ELEMENT` 的 **复用** 逻辑。具体怎么复用，我们会在后面进行详细讲解。

对于现在我们的代码而言，目前还不能增加 `key` 属性，所以我们需要为我们的代码增加 `key` 属性的挂载：

1. 在 `packages/runtime-core/src/vnode.ts` 的 `createBaseVNode` 中，增加 `key` 属性：

代码块

```
1     const vnode = {
2         __v_isVNode: true,
3         type,
4         props,
5         shapeFlag,
6     +   key: props?.key || null
7     } as VNode
```

这样，我们的 `vnode` 就可以具备 `key` 属性了。

01：前言 ◂ 上一节　　下一节 ▸ 03：源码阅读：场景一：自前向后的 diff 对比