Sunday • 更新于 2022-10-19

上一节 04：源码阅读：... 06：源码阅读：... 下一节

05：框架实现：构建 h 函数，处理 ELEMENT + ARRAY_CHILDREN 场景

根据上一小节的源码阅读可知， `ELEMENT + ARRAY_CHILDREN` 场景下的处理，我们只需要在 `packages/runtime-core/src/vnode.ts` 中，处理 `isArray` 场景即可：

1. 在 `packages/runtime-core/src/vnode.ts` 中，找到 `normalizeChildren` 方法：

代码块

```
else if (isArray(children)) {
+       type = ShapeFlags.ARRAY_CHILDREN
    }
```

2. 创建测试实例 `packages/vue/examples/runtime/h-element-ArrayChildren.html` ：

代码块

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

  const vnode = h('div', {
    class: 'test'
  }, [
    h('p', 'p1'),
    h('p', 'p2'),
    h('p', 'p3')
  ])

  console.log(vnode);
</script>
```

可以得出同样的打印结果。（大家也可以直接把打印的 `vnode` 传递给 `vue 3` 的 `render` 函数，发现可以正常渲染）

局部总结

那么到现在我们可以先做一个局部的总结。

对于 `vnode` 而言，我们现在已经知道，它存在一个 `shapeFlag` 属性，该属性表示了当前 `VNode` 的 **“类型”**，这是一个非常关键的属性，在后面的 `render` 函数中，还会再次看到它。

`shapeFlag` 分成两部分：

1. `createVNode`：此处计算 “DOM” 类型，比如 `Element`
2. `createBaseVNode`：此处计算 “children” 类型，比如 `Text || Array`

04：源码阅读：h 函数，跟踪 ELEMENT + ... 上一节 下一节 06：源码阅读：h 函数，组件的本质与对应...