Sunday • 更新于 2022-10-19

上一节 01：前言 03：框架实现：... 下一节

02：源码阅读：初见 render 函数，ELEMENT 节点的挂载操作

在上一小节，我们实现过一个这样的测试案例 `packages/vue/examples/imooc/runtime/h-element.html`：

代码块

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

  const vnode = h('div', {
    class: 'test'
  }, 'hello render')


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

这样我们可以得到一个对应的 `vnode`，我们可以使用 `render` 函数来去渲染它。

代码块

```
render(vnode, document.querySelector('#app'))
```

我们可以在 `packages/runtime-core/src/renderer.ts` 的第 `2327` 行，增加 `debugger`：

1. 进入 `render` 函数
2. `render` 函数接收三个参数：
    1. `vnode`：虚拟节点
    2. `container`：容器
    3. `isSVG`：是否是 `SVG`
3. 执行 `patch(container._vnode || null, vnode, container, null, null, null, isSVG)`
    1. 根据我们之前所说，我们知道 `patch` 表示 **更新** 节点。这里传递的参数我们主要关注 **前三个。**
    2. `container._vnode` 表示 **旧节点（`n1`）**，`vnode` 表示 **新节点（`n2`）**，`container` 表示 **容器**
    3. 执行 `switch`，`case` 到 `if (shapeFlag & ShapeFlags.ELEMENT)`：
        1. 我们知道此时 `shapeFlag` 的值是 `9`，转为为二进制：

        代码块

        ```
        00000000 00000000 00000000 00001001
        ```

        2. `ShapeFlags.ELEMENT` 的值是 `1`，转为二进制：

        代码块

        ```
        00000000 00000000 00000000 00000001
        ```

        3. 两者执行 按位与 (&)，得到的二进制结果为：

```
00000000 00000000 00000000 00000001 // 十进制 1
```

4. 即 `if (shapeFlag & ShapeFlags.ELEMENT) === if (1)`，等同于 `if(true)`
5. 所以会进入 `if`

4. 触发 `processElement` 方法：
    1. 进入 `processElement` 方法
    2. 因为当前为 **挂载操作**，所以 **没有旧节点**，即： `n1 === null`
    3. 触发 `mountElement` 方法，即 **挂载方法**：
        1. 进入 `mountElement` 方法
        2. 执行 `el = vnode.el = hostCreateElement(....)`，该方法为创建 `Element` 的方法
            1. 进入该方法，可以发现该方法指向 `packages/runtime-dom/src/nodeOps.ts` 中的 `createElement` 方法
            2. 不知道大家还记不记得，之前我们说过： `vue` 为了保持兼容性，把所有和浏览器相关的 `API` 封装到了 `runtime-dom` 中
            3. 在 `createElement` 中的代码非常简单就是通过 `document.createElement` 方法创建 `dom`，并返回
        3. 此时 `el` 和 `vnode.el` 的值为 `createElement` **生成的** `div` **实例**
        4. 接下来处理：**子节点**
        5. 执行 `if (shapeFlag & ShapeFlags.TEXT_CHILDREN)`，同样的 按位与 (&)，大家可以自己进行下二进制的转换
        6. 触发 `hostSetElementText` 方法
            1. 进入该方法，同样指向 `packages/runtime-dom/src/nodeOps.ts` 文件下的 `setElementText` 方法
            2. 里面的代码非常简单只有一行 `el.textContent = text`
        7. 那么至此 ** `div` 已经生成，并且 `textContent` 存在值 **，如果此时触发 `div` 的 `outerHTML` 方法，得到 `<div>hello render</div>`
        8. 那么此时，我们就 **只缺少** `class` **属性** 了，所以接下来将进入 `props` 的处理
        9. 执行 `for` 循环，进入 `hostPatchProp(...)` 方法，此时 `key = class` ， `props = {class: 'test'}`
            1. 进入 `hostPatchProp(...)` 方法，
            2. 该方法位于 `/packages/runtime-dom/src/patchProp.ts` 下的 `patchProp` 方法
            3. 此时 `key === class`，所以将触发 `patchClass`
                1. 进入 `patchClass`，我们可以看到它内部的代码也比较简单，主要分成了三种情况进行处理：

                代码块

                ```
                //  value 此时的值为 test（即：类名）
                if (value == null) {
                  el.removeAttribute('class')
                } else if (isSVG) {
                  el.setAttribute('class', value)
                } else {
                  el.className = value
                }
                ```

            2. 完成 `class` 设定

10. 当执行完成 `hostPatchProp` 之后， 如果此时触发 `div` 的 `outerHTML` 方法，得到 `<div class="test">hello render</div>`
11. 现在 `dom` 已经构建好了，最后就只剩下 **挂载** 操作了
12. 继续执行代码将进入 `hostInsert(el, container, anchor)` 方法：
    1. 进入 `hostInsert` 方法
    2. 该方法位于 `packages/runtime-dom/src/modules` 中 `insert` 方法
    3. 内部同样只有一行代码： `parent.insertBefore(child, anchor || null)`
    4. 我们知道 insertBefore 方法可以插入到 `dom` 到指定区域
13. 那么到这里，我们已经成功的把 `div` 插入到了 `dom` 树中，执行完成 `hostInsert` 方法之后，浏览器会出现对应的 `div`

4. 至此，整个 `patchElement` 执行完成

4. 执行 `container._vnode = vnode` ，为 **旧节点赋值**

由以上代码可知：

1. 整个挂载 `Element | Text_Children` 的过程分为以下步骤：
    1. 触发 `patch` 方法
    2. 根据 `shapeFlag` 的值，判定触发 `processElement` 方法
    3. 在 `processElement` 中，根据 **是否存在** 旧VNode 来判定触发 **挂载** 还是 **更新** 的操作
        1. 挂载中分成了4大步：
            1. 生成 `div`
            2. 处理 `textContent`
            3. 处理 `propx`
            4. 挂载 `dom`
    4. 通过 `container._vnode = vnode` 赋值 **旧 VNode**

01：前言 ‹ 上一节　　下一节 › 03：框架实现：构建 renderer 基本架构

我要提出意见反馈