Sunday • 更新于 2022-10-19

上一节 02：HTML DO… 04：h 函数 与 r… 下一节

03：挂载与更新

这一小节，我们将通过一个极简的案例，来了解两个比较重要的概念：

1. 挂载：`mount`
2. 更新：`patch`

挂载：mount

首先我们先来构建这个案例（该案例在 第二章第七小节 《运行时》进行过大致的讲解）：

代码块

```
<script>
  const VNode = {
    type: 'div',
    children: 'hello render'
  }

  // 创建 render 渲染函数
  function render(oldVNode, newVNode, container) {
    if (!oldVNode) {
      mount(newVNode, container)
    }
  }

  // 挂载函数
  function mount(vnode, container) {
    // 根据 type 生成 element
    const ele = document.createElement(vnode.type)
    // 把 children 赋值给 ele 的 innerText
    ele.innerText = vnode.children
    // 把 ele 作为子节点插入 body 中
    container.appendChild(ele)
  }

  render(null, VNode, document.querySelector('#app'))
</script>
```

在当前案例中，我们首先创建了一个 `render` 渲染函数，该函数接收三个参数：

1. `oldVNode`：旧的 `VNode`
2. `newVNode`：新的 `VNode`
3. `container`：容器

当 `oldVNode` 不存在时，那么我们就认为这是一个全新的渲染，也就是 **挂载。**

所以以上的 `mount` 方法，我们就可以把它称为是一个 **挂载方法。**

更新：patch

旧的视图不可能被一直展示，它会在未来某一个时刻被更新为全新的视图。

比如：

代码块

```
<script>
  const VNode = {
    type: 'div',
    children: 'hello render'
  }

  const VNode2 = {
    type: 'div',
    children: 'patch render'
  }

  // 创建 render 渲染函数
  function render(oldVNode, newVNode, container) {
    if (!oldVNode) {
      mount(newVNode, container)
    } else {
      patch(oldVNode, newVNode, container)
    }
  }

  // 挂载函数
  function mount(vnode, container) {
    // 根据 type 生成 element
    const ele = document.createElement(vnode.type)
    // 把 children 赋值给 ele 的 innerText
    ele.innerText = vnode.children
    // 把 ele 作为子节点插入 body 中
    container.appendChild(ele)
  }

  // 取消挂载
  function unmount(container) {
    container.innerHTML = ''
  }

  // 更新函数
  function patch(oldVNode, newVNode, container) {
    unmount(container)

    // 根据 type 生成 element
    const ele = document.createElement(newVNode.type)
    // 把 children 赋值给 ele 的 innerText
    ele.innerText = newVNode.children
    // 把 ele 作为子节点插入 body 中
    container.appendChild(ele)
  }

  render(null, VNode, document.querySelector('#app'))

  setTimeout(() => {
    render(VNode, VNode2, document.querySelector('#app'))
  }, 2000);
</script>
```

我们在原有的代码中去新增了一部分逻辑，新增了 `patch` 函数。

在 `patch` 函数中，我们先 **删除了旧的** `VNode` **，然后创建了一个新的** `VNode` **。** 这样的一个流程，我们就把它叫做 **挂载** `patch`

总结

本小节我们通过一个简单的例子讲解了 **挂载** `mount` 和 **更新** `patch` 的概念，这两个概念 Vue 3 官方文档 也对此进行了详细的介绍：

1. **挂载**：运行时渲染器调用渲染函数，遍历返回的虚拟 DOM 树，并基于它创建实际的 DOM 节点。
2. **更新**：当一个依赖发生变化后，副作用会重新运行，这时候会创建一个更新后的虚拟 DOM 树。运行时渲染器遍历这棵新树，将它与旧树进行比较，然后将必要的更新应用到真实 DOM 上去。

这两个概念在我们后面去实现 `renderer` 渲染器的时候还会经常的使用到。

02：HTML DOM 节点树与虚拟 DOM 树 ◂ 上一节　　下一节 ▸ 04：h 函数 与 render 函数

我要提出意见反馈