Sunday • 更新于 2022-10-19

上一节 03：框架实现：... 05：局部总结：... 下一节

04：源码阅读：无状态基础组件更新逻辑

此时我们的无状态组件挂载已经完成，接下来我们来看一下 **无状态组件更新** 的处理逻辑。

创建如下测视案例 `packages/vue/examples/imooc/runtime/render-component-update.html` ，

代码块

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

  const component = {
    render() {
      return h('div', 'hello component')
    }
  }

  const vnode = h(component)

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

  setTimeout(() => {
    const component2 = {
      render() {
        return h('div', 'update component')
      }
    }

    const vnode2 = h(component2)

    render(vnode2, document.querySelector('#app'))
  }, 2000);
</script>
```

在 `render` 中进入 `debugger`：

1. **第一次** 进入 `render` ，执行 **组件挂载** 逻辑
    1. **第一次** 触发 `patch` 函数，执行组件挂载
        1. 进入 `patch` 函数
        2. 因为当前是组件挂载，所以会触发 `processComponent` 方法
            1. 进入 `processComponent`
            2. 触发 `mountComponent`
                1. 进入 `mountComponent`
                2. 生成组件实例 `instance` 和 `initialVNode.component`
                3. 执行 `setupComponent(instance)` ，为 `instance.render` 赋值
                4. 执行 `setupRenderEffect` 方法
                    1. 进入 `setupRenderEffect` 方法
                    2. 生成 `effect` 实例，绑定 `fn` 为 `componentUpdateFn`

4. 执行 `update` 方法，从而触发 `componentUpdateFn`

    1. 进入 `componentUpdateFn`
    2. 通过 `renderComponentRoot` 方法，触发 `render` 拿到 `subTree`
    3. 通过 `patch` 方法进行挂载

2. **第二次** 触发 `patch`，此时为 `component` 的 `render` 渲染

    1. 因为 `render` 为 `ELEMENT` 的渲染操作
    2. 所以会触发 `processElement`
    3. ...

2. 此时第一次 `component` 的挂载操作完成

3. 延迟两秒之后，再次进入 `render`，此时是第二个 `component` 的挂载，即：**更新**

    1. 同样进入 `patch`，此时的参数为：

    ```
    const patch: PatchFn = (
      n1,   n1 = {__v_isVNode: true, __v_skip: true, type: {…}, props: null
      n2,   n2 = {__v_isVNode: true, __v_skip: true, type: {…}, props: null
      container,   container = div#app {_vnode: {…}, align: '', title: '',
      anchor = null,   anchor = null
      parentComponent = null,   parentComponent = null
      parentSuspense = null,   parentSuspense = null
      isSVG = false,   isSVG = false
      slotScopeIds = null,   slotScopeIds = null
      optimized = __DEV__ && isHmrUpdating ? false : !!n2.dynamicChildren
    ) => {
    ```

    2. 此时存在两个不同的 `VNode`，所以 `if (n1 && !isSameVNodeType(n1, n2))` 判断为 `true`，此时将执行 **卸载旧的** `VNode` **逻辑**

    3. 执行 `unmount(n1, parentComponent, parentSuspense, true)`，触发 **卸载逻辑**

    4. 代码继续执行，经过 `switch`，再次执行 `processComponent`，因为 **旧的** `VNode` **已经被卸载，所以此时** `n1 = null`

        1. 进入 `processComponent` 方法，此时的参数为：

        ```
        const processComponent = (
          n1: VNode | null,   n1 = null
          n2: VNode,   n2 = {__v_isVNode: true, __v_skip: true, type: {…}, props: null, key:
          container: RendererElement,   container = div#app {_vnode: {…}, align: '', title: '
          anchor: RendererNode | null,   anchor = null
          parentComponent: ComponentInternalInstance | null,   parentComponent = null
          parentSuspense: SuspenseBoundary | null,   parentSuspense = null
          isSVG: boolean,   isSVG = false
          slotScopeIds: string[] | null,   slotScopeIds = null
          optimized: boolean   optimized = false
        ) => {
        ```

        2. 代码继续执行，发现 **再次触发** `mountComponent`，执行 **挂载操作**
        3. 后续省略...

至此，组件更新完成。

由以上代码可知：

1. 所谓的组件更新，其实本质上就是一个 **卸载、挂载** 的逻辑

    1. 对于这样的卸载逻辑，我们之前已经完成过。
    2. 所以，目前我们的代码 **支持** 组件的更新操作。

可以在 `vue-next-mini` 中 直接通过测试实例进行测试 `packages/vue/examples/imooc/runtime/render-component-update.html`：

代码块

```
1  <script>
2    const { h, render } = Vue
3
4    const component = {
```

```
      return h('div', 'hello component')
    }
  }

  const vnode = h(component)
  // 挂载
  render(vnode, document.querySelector('#app'))

  setTimeout(() => {
    const component2 = {
      render() {
        return h('div', '你好，世界')
      }
    }

    const vnode2 = h(component2)
    // 挂载
    render(vnode2, document.querySelector('#app'))
  }, 2000);
</script>
```

03：框架实现：完成无状态基础组件的挂载 ◂ 上一节　　下一节 ▸ 05：局部总结：无状态组件的挂载、更新、...

我要提出意见反馈

企业服务　网站地图　网站首页　关于我们　联系我们　讲师招募　帮助中心　意见反馈　代码托管

意见反馈　收藏教程　标记书签