双11返场进行中！优惠Go on，快来捡漏！ 02 天 14 时 44 分 12 秒 Go >>>

Sunday • 更新于 2022-10-19

上一节 02：源码阅读：... 04：源码阅读：... 下一节

03：框架实现：完成无状态基础组件的挂载逻辑

明确好了源码的无状态组件挂载之后，那么接下来我们来进行一下对应实现。

1. 在 `packages/runtime-core/src/renderer.ts` 的 `patch` 方法中，创建 `processComponent` 的触发：

代码块

```
else if (shapeFlag & ShapeFlags.COMPONENT) {
  // 组件
  processComponent(oldVNode, newVNode, container, anchor)
}
```

2. 创建 `processComponent` 函数：

代码块

```
/**
 * 组件的打补丁操作
 */
const processComponent = (oldVNode, newVNode, container, anchor) => {
    if (oldVNode == null) {
        // 挂载
        mountComponent(newVNode, container, anchor)
    }
}
```

3. 创建 `mountComponent` 方法：

代码块

```
const mountComponent = (initialVNode, container, anchor) => {
    // 生成组件实例
    initialVNode.component = createComponentInstance(initialVNode)
    // 浅拷贝，绑定同一块内存空间
    const instance = initialVNode.component

    // 标准化组件实例数据
    setupComponent(instance)

    // 设置组件渲染
    setupRenderEffect(instance, initialVNode, container, anchor)
}
```

4. 创建 `packages/runtime-core/src/component.ts` 模块，构建 `createComponentInstance` 函数逻辑：

代码块

```
let uid = 0

/**
 * 创建组件实例
 */
export function createComponentInstance(vnode) {
    const type = vnode.type
```

```
        uid: uid++, // 唯一标记
        vnode, // 虚拟节点
        type, // 组件类型
        subTree: null!, // render 函数的返回值
        effect: null!, // ReactiveEffect 实例
        update: null!, // update 函数，触发 effect.run
        render: null // 组件内的 render 函数
    }

    return instance
}
```

5. 在 `packages/runtime-core/src/component.ts` 模块，创建 `setupComponent` 函数逻辑：

代码块

```
/**
 * 规范化组件实例数据
 */
export function setupComponent(instance) {
    // 为 render 赋值
    const setupResult = setupStatefulComponent(instance)
    return setupResult
}

function setupStatefulComponent(instance) {
    finishComponentSetup(instance)
}

export function finishComponentSetup(instance) {
    const Component = instance.type

    instance.render = Component.render
}
```

6. 在 `packages/runtime-core/src/renderer.ts` 中，创建 `setupRenderEffect` 函数：

代码块

```
/**
 * 设置组件渲染
 */
const setupRenderEffect = (instance, initialVNode, container, anchor) => {
    // 组件挂载和更新的方法
    const componentUpdateFn = () => {
        // 当前处于 mounted 之前，即执行 挂载 逻辑
        if (!instance.isMounted) {
            // 从 render 中获取需要渲染的内容
            const subTree = (instance.subTree = renderComponentRoot(instance))

            // 通过 patch 对 subTree，进行打补丁。即：渲染组件
            patch(null, subTree, container, anchor)

            // 把组件根节点的 el，作为组件的 el
            initialVNode.el = subTree.el
        } else {
        }
    }

    // 创建包含 scheduler 的 effect 实例
    const effect = (instance.effect = new ReactiveEffect(
        componentUpdateFn,
        () => queuePreFlushCb(update)
    ))

    // 生成 update 函数
    const update = (instance.update = () => effect.run())

    // 触发 update 函数，本质上触发的是 componentUpdateFn
    update()
}
```

意见反馈 收藏教程 标记书签

7. 创建 `packages/runtime-core/src/componentRenderUtils.ts` 模块，构建 `renderComponentRoot` 函数：

代码块

```
import { ShapeFlags } from 'packages/shared/src/shapeFlags'

/**
 * 解析 render 函数的返回值
 */
export function renderComponentRoot(instance) {
    const { vnode, render } = instance

    let result
    try {
        // 解析到状态组件
        if (vnode.shapeFlag & ShapeFlags.STATEFUL_COMPONENT) {
            // 获取到 result 返回值
            result = normalizeVNode(render!())
        }
    } catch (err) {
        console.error(err)
    }

    return result
}

/**
 * 标准化 VNode
 */
export function normalizeVNode(child) {
    if (typeof child === 'object') {
        return cloneIfMounted(child)
    }
}

/**
 * clone VNode
 */
export function cloneIfMounted(child) {
    return child
}
```

至此代码完成。

创建 `packages/vue/examples/runtime/render-component.html` 测试实例：

代码块

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

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

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

此时，组件渲染完成。

02：源码阅读：无状态基础组件挂载逻辑 ◂ 上一节　　下一节 ▸ 04：源码阅读：无状态基础组件更新逻辑

我要提出意见反馈

意见反馈　收藏教程　标记书签

Copyright © 2022 imooc.com All Rights Reserved | 京ICP备 12003892号-11 京公网安备11010802030151号

意见反馈 收藏教程 标记书签