Sunday • 更新于 2022-10-19

上一节 02：阅读源码：... 04：源码阅读：... 下一节

03：框架实现：构建 h 函数，处理 ELEMENT + TEXT_CHILDREN 场景

那么接下来我们依据刚才所查看的源码场景，处理自己的对应逻辑。

1. 创建 `packages/shared/src/shapeFlags.ts` ，写入所有的对应类型：

代码块

```
export const enum ShapeFlags {
    /**
     * type = Element
     */
    ELEMENT = 1,
    /**
     * 函数组件
     */
    FUNCTIONAL_COMPONENT = 1 << 1,
    /**
     * 有状态（响应数据）组件
     */
    STATEFUL_COMPONENT = 1 << 2,
    /**
     * children = Text
     */
    TEXT_CHILDREN = 1 << 3,
    /**
     * children = Array
     */
    ARRAY_CHILDREN = 1 << 4,
    /**
     * children = slot
     */
    SLOTS_CHILDREN = 1 << 5,
    /**
     * 组件：有状态（响应数据）组件 | 函数组件
     */
    COMPONENT = ShapeFlags.STATEFUL_COMPONENT | ShapeFlags.FUNCTIONAL_COMPONENT
}
```

2. 创建 `packages/runtime-core/src/h.ts` ，构建 `h` 函数：

代码块

```
import { isArray, isObject } from '@vue/shared'
import { createVNode, isVNode, VNode } from './vnode'

export function h(type: any, propsOrChildren?: any, children?: any): VNode {
    // 获取用户传递的参数数量
    const l = arguments.length
    // 如果用户只传递了两个参数，那么证明第二个参数可能是 props ，也可能是 children
    if (l === 2) {
        // 如果 第二个参数是对象，但不是数组。则第二个参数只有两种可能性：1. VNode 2.普通的
        if (isObject(propsOrChildren) && !isArray(propsOrChildren)) {
            // 如果是 VNode，则 第二个参数代表了 children
            if (isVNode(propsOrChildren)) {
                return createVNode(type, null, [propsOrChildren])
```

```
            return createVNode(type, propsOrChildren)
        }
        // 如果第二个参数不是单纯的 object，则 第二个参数代表了 props
        else {
            return createVNode(type, null, propsOrChildren)
        }
    }
    // 如果用户传递了三个或以上的参数，那么证明第二个参数一定代表了 props
    else {
        // 如果参数在三个以上，则从第二个参数开始，把后续所有参数都作为 children
        if (l > 3) {
            children = Array.prototype.slice.call(arguments, 2)
        }
        // 如果传递的参数只有三个，则 children 是单纯的 children
        else if (l === 3 && isVNode(children)) {
            children = [children]
        }
        // 触发 createVNode 方法，创建 VNode 实例
        return createVNode(type, propsOrChildren, children)
    }
}
```

3. 创建 `packages/runtime-core/src/vnode.ts`，处理 `VNode` 类型和 `isVNode` 函数：

代码块

```
export interface VNode {
    __v_isVNode: true
    type: any
    props: any
  children: any
    shapeFlag: number
}

export function isVNode(value: any): value is VNode {
    return value ? value.__v_isVNode === true : false
}
```

4. 在 `packages/runtime-core/src/vnode.ts` 中，构建 `createVNode` 函数：

代码块

```
/**
 * 生成一个 VNode 对象，并返回
 * @param type vnode.type
 * @param props 标签属性或自定义属性
 * @param children 子节点
 * @returns vnode 对象
 */
export function createVNode(type, props, children): VNode {
    // 通过 bit 位处理 shapeFlag 类型
    const shapeFlag = isString(type) ? ShapeFlags.ELEMENT : 0

    return createBaseVNode(type, props, children, shapeFlag)
}

/**
 * 构建基础 vnode
 */
function createBaseVNode(type, props, children, shapeFlag) {
    const vnode = {
        __v_isVNode: true,
        type,
        props,
        shapeFlag
    } as VNode

    normalizeChildren(vnode, children)

    return vnode
```

```
30
31  export function normalizeChildren(vnode: VNode, children: unknown) {
32      let type = 0
33      const { shapeFlag } = vnode
34      if (children == null) {
35          children = null
36      } else if (isArray(children)) {
37          // TODO: array
38      } else if (typeof children === 'object') {
39          // TODO: object
40      } else if (isFunction(children)) {
41          // TODO: function
42      } else {
43          // children 为 string
44          children = String(children)
45          // 为 type 指定 Flags
46          type = ShapeFlags.TEXT_CHILDREN
47      }
48      // 修改 vnode 的 chidlren
49      vnode.children = children
50      // 按位或赋值
51      vnode.shapeFlag |= type
52  }
```

5. 在 `index` 中导出 `h` 函数

至此 `h` 函数创建完成。

下面我们可以创建对应的测试实例， `packages/vue/examples/runtime/h-element.html`：

代码块

```
1   <script>
2     const { h } = Vue
3
4     const vnode = h('div', {
5       class: 'test'
6     }, 'hello render')
7
8     console.log(vnode);
9   </script>
```

最终打印的结果为：

代码块

```
1   children: "hello render"
2   props: {class: 'test'}
3   shapeFlag: 9
4   type: "div"
5   __v_isVNode: true
```

那么至此，我们就已经构建好了： `type = Element`， `children = Text` **的** `VNode` **对象**

02：阅读源码：初见 h 函数，跟踪 Vue 3 源... ◂ 上一节　　下一节 ▸ 04：源码阅读：h 函数，跟踪 ELEMENT + ...

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