## **render**

# **render**

Vue 的 \_render 方法是实例的一个私有方法，它用来把实例渲染成一个虚拟 Node。它的定义在 src/core/instance/render.js 文件中：

Vue.prototype.\_render = function (): VNode {

const vm: Component = this

const { render, \_parentVnode } = vm.$options

// reset \_rendered flag on slots for duplicate slot check

if (process.env.NODE\_ENV !== 'production') {

for (const key in vm.$slots) {

// $flow-disable-line

vm.$slots[key].\_rendered = false

}

}

if (\_parentVnode) {

vm.$scopedSlots = \_parentVnode.data.scopedSlots || emptyObject

}

// set parent vnode. this allows render functions to have access

// to the data on the placeholder node.

vm.$vnode = \_parentVnode

// render self

let vnode

try {

vnode = render.call(vm.\_renderProxy, vm.$createElement)

} catch (e) {

handleError(e, vm, `render`)

// return error render result,

// or previous vnode to prevent render error causing blank component

/\* istanbul ignore else \*/

if (process.env.NODE\_ENV !== 'production') {

if (vm.$options.renderError) {

try {

vnode = vm.$options.renderError.call(vm.\_renderProxy, vm.$createElement, e)

} catch (e) {

handleError(e, vm, `renderError`)

vnode = vm.\_vnode

}

} else {

vnode = vm.\_vnode

}

} else {

vnode = vm.\_vnode

}

}

// return empty vnode in case the render function errored out

if (!(vnode instanceof VNode)) {

if (process.env.NODE\_ENV !== 'production' && Array.isArray(vnode)) {

warn(

'Multiple root nodes returned from render function. Render function ' +

'should return a single root node.',

vm

)

}

vnode = createEmptyVNode()

}

// set parent

vnode.parent = \_parentVnode

return vnode

}

这段代码最关键的是 render 方法的调用，我们在平时的开发工作中手写 render 方法的场景比较少，而写的比较多的是 template 模板，在之前的 mounted 方法的实现中，会把 template 编译成 render 方法，但这个编译过程是非常复杂的，我们不打算在这里展开讲，之后会专门花一个章节来分析 Vue 的编译过程。

在 Vue 的官方文档中介绍了 render 函数的第一个参数是 createElement，那么结合之前的例子：

<div id="app">

{{ message }}

</div>

相当于我们编写如下 render 函数：

render: function (createElement) {

return createElement('div', {

attrs: {

id: 'app'

},

}, this.message)

}

再回到 \_render 函数中的 render 方法的调用：

vnode = render.call(vm.\_renderProxy, vm.$createElement)

可以看到，render 函数中的 createElement 方法就是 vm.$createElement 方法：

export function initRender (vm: Component) {

// ...

// bind the createElement fn to this instance

// so that we get proper render context inside it.

// args order: tag, data, children, normalizationType, alwaysNormalize

// internal version is used by render functions compiled from templates

vm.\_c = (a, b, c, d) => createElement(vm, a, b, c, d, false)

// normalization is always applied for the public version, used in

// user-written render functions.

vm.$createElement = (a, b, c, d) => createElement(vm, a, b, c, d, true)

}

实际上，vm.$createElement 方法定义是在执行 initRender 方法的时候，可以看到除了 vm.$createElement 方法，还有一个 vm.\_c 方法，它是被模板编译成的 render 函数使用，而 vm.$createElement 是用户手写 render 方法使用的， 这俩个方法支持的参数相同，并且内部都调用了 createElement 方法。

## **总结**

vm.\_render 最终是通过执行 createElement 方法并返回的是 vnode，它是一个虚拟 Node。Vue 2.0 相比 Vue 1.0 最大的升级就是利用了 Virtual DOM。因此在分析 createElement 的实现前，我们先了解一下 Virtual DOM 的概念。

****任务****

请仔细阅读本文档，为下面学习视频内容做准备。