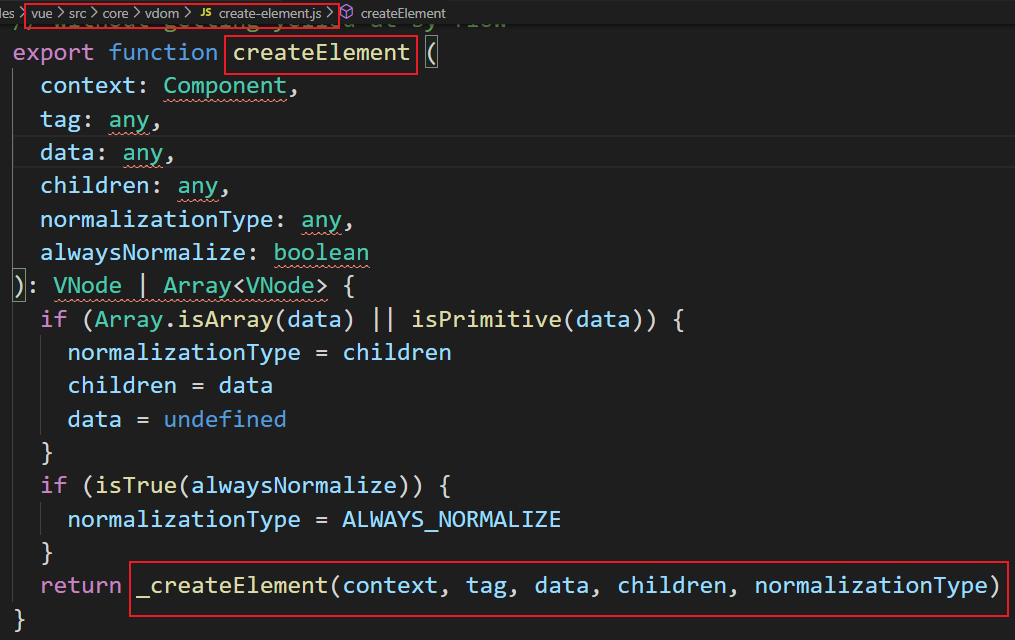
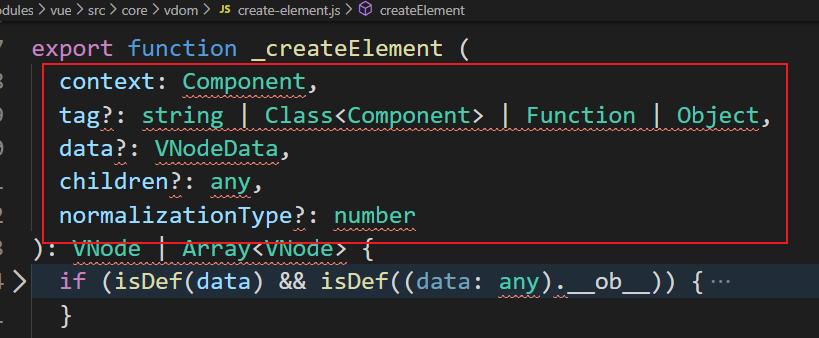
**createElemet**



对\_createElement方法的封装，它允许传入的参数更加灵活.当data没有（是数组就相当于没有传而是直接传了children）进行参数重载。

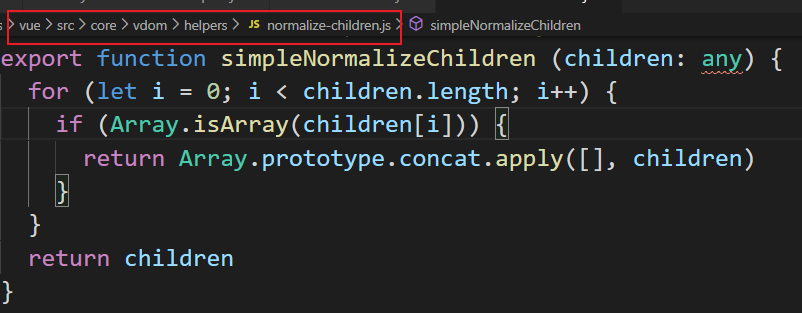
**\_createElemet**



\_createElement 方法有 5 个参数，contex表示 VNode 的上下文环境，它是 Component 类型；tag表示标签，它可以是一个字符串，也可以是一个 Component；data 表示 VNode 的数据，它是一个 VNodeData 类型，可以在 flow/vnode.js 中找到它的定义，这里先不展开说；children 表示当前 VNode 的子节点，它是任意类型的，它接下来需要被规范为标准的 VNode 数组；normalizationType 表示子节点规范的类型，类型不同规范的方法也就不一样，它主要是参考 render 函数是编译生成的还是用户手写的。

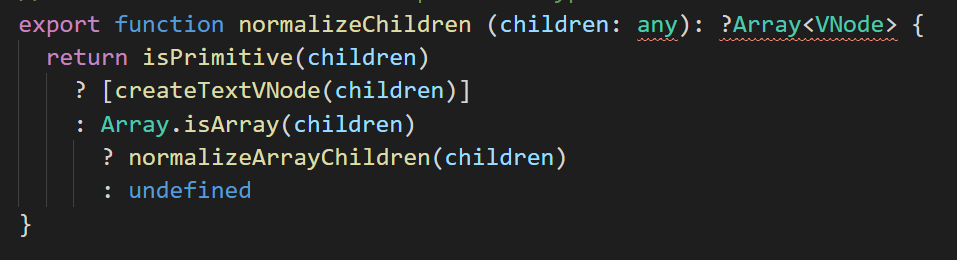
**规范化children**

**1. simpleNormalizeChildren**



调用场景是 render 函数当函数是编译生成的

**2.** **normalizeChildren**



一个场景是 render 函数是用户手写的，当 children 只有一个节点的时候，Vue.js 从接口层面允许用户把 children 写成基础类型用来创建单个简单的文本节点，这种情况会调用 createTextVNode 创建一个文本节点的 VNode；另一个场景是当编译 slot、v-for 的时候会产生嵌套数组的情况，会调用 normalizeArrayChildren 方法

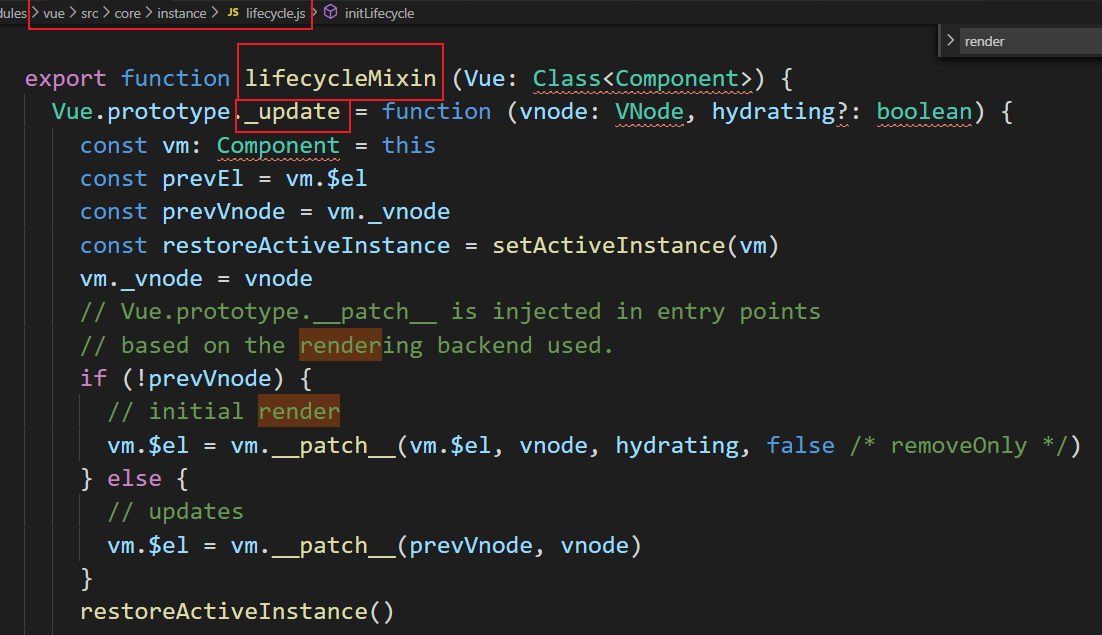
**normalizeArrayChildren最终递归得到一个children数组**

**然后通过createElement获得Vnode**

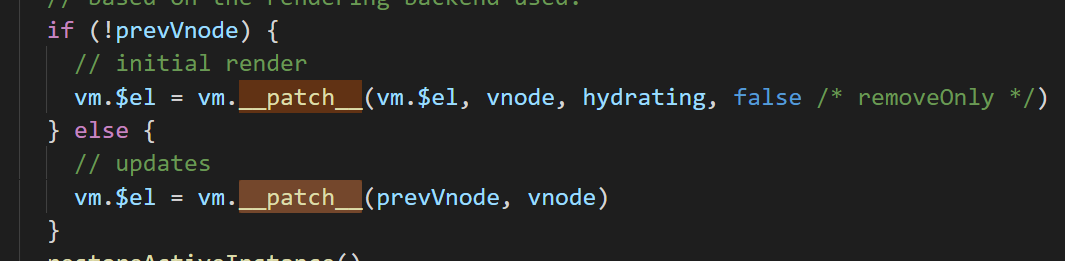
**\_update()**

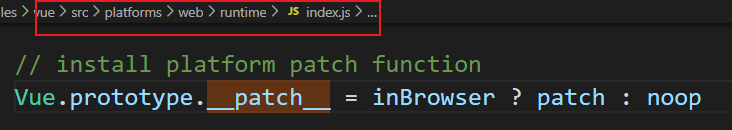
在vue构造函数那lifecycleMixin(Vue)原型上挂载了\_update

Vue 的 \_update 是实例的一个私有方法，它被调用的时机有 2 个，一个是首次渲染，一个是数据更新的时候；由于我们这一章节只分析首次渲染部分，数据更新部分会在之后分析响应式原理的时候涉及。\_update 方法的作用是把 VNode 渲染成真实的 DOM。

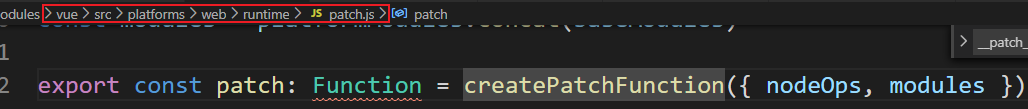


**核心调用\_\_patch\_\_**

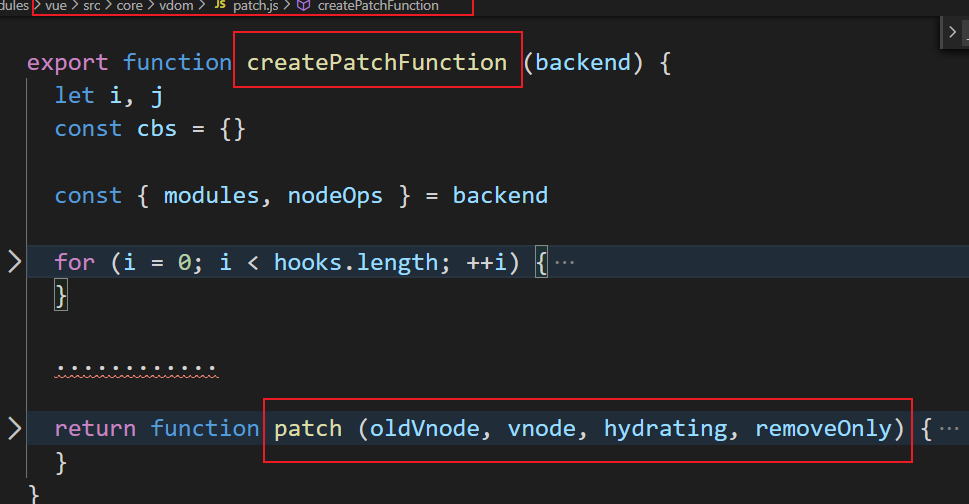




因为在服务端渲染中，没有真实的浏览器 DOM 环境，所以不需要把 VNode 最终转换成 DOM，因此是一个空函数，而在浏览器端渲染中，它指向了 patch 方法



nodeOps 封装了一系列 DOM 操作的方法，modules 定义了一些模块的钩子函数的实现



createPatchFunction 内部定义了一系列的辅助方法，最终返回了一个 patch 方法，这个方法就赋值给了 vm.\_update 函数里调用的 vm.\_\_patch\_\_。它接收 4个参数，oldVnode 表示旧的 VNode 节点，它也可以不存在或者是一个 DOM 对象；vnode 表示执行 \_render 后返回的 VNode 的节点；hydrating 表示是否是服务端渲染；removeOnly 是给 transition-group 用的

**createElm (vnode挂载到真实dom上)**