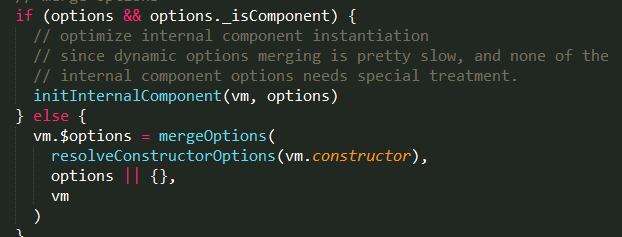
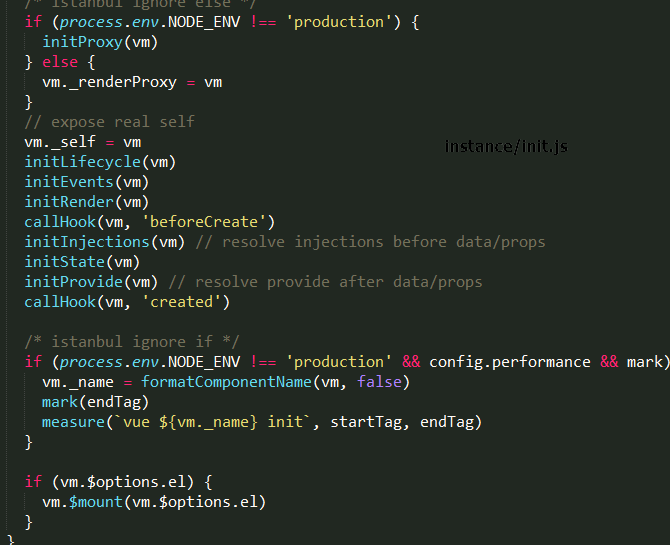
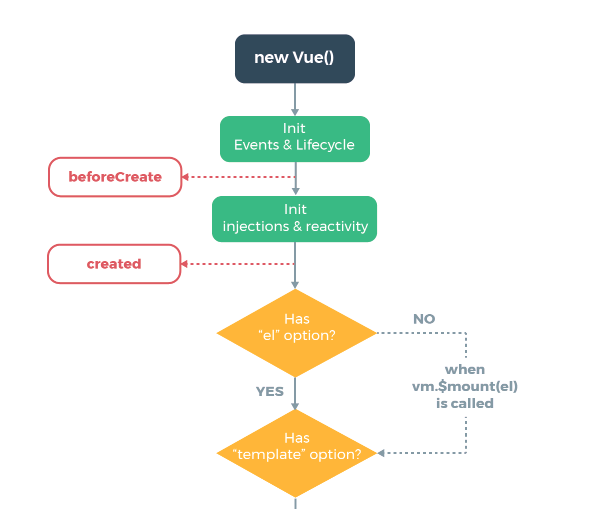
学习vue的源码的记录，肯定有很多不足之处和错误的地方，慢慢学习。

vue的Instance的initMixin

绑定vm.$options

上图对应的就是官网生命周期的



在instance的lifecycle.js中：

1.initLifecycle:主要在vue对象上初始化一些属性

2.mountComponent:注册beforeMount事件，把虚拟节点挂载到页面中吗，注册mount事件

2.lifecycleMixin，在index.js中被调用，在数据改变之间注册一个beforeUpdate事件

虚拟节点重新渲染和页面编译。updated事件在observer的scheduler.js中？？？

3.activateChildComponent是注册activated事件

4.deactivateChildComponent是注册deactivated事件

5.Vue.prototype.$destroy是注册beforeDestroy事件，执行销毁，注册destroyed

在instance的events.js中：

1.initEvents:如果是有\_parentListeners，加listeners？？？

2.eventsMixin:绑定四个方法，因为是绑定在Vue的原型上，必须是vue实例才能调用

2.1$on:(vm.\_events[event] || (vm.\_events[event] = [])).push(fn)

2.2$once:function on () {

vm.$off(event, on)

fn.apply(vm, arguments)

}

on.fn = fn

vm.$on(event, on)

2.3$off:vm.\_events(是一个数组) ，vm.\_events[event].splice(i, 1)

2.4$emit:toArray(vm.\_events[event])[i].apply(vm, args)

在instance的render.js

1.initRender

1.1vm.$slots = resolveSlots(options.\_renderChildren, renderContext) 获取 VNodes 列表中的静态内容

1.2vm.$scopedSlots = emptyObject获得能用作函数的作用域插槽，这个函数返回 VNodes

1.3defineReactive(vm, '$attrs', parentData && parentData.attrs || emptyObject, null, true)

1.4vm.$createElement = (a, b, c, d) => createElement(vm, a, b, c, d, true)

1.5defineReactive(vm, '$listeners', options.\_parentListeners || emptyObject, null, true)

$listener是包含了父作用域中的 (不含 .native 修饰器的) v-on 事件监听器。它可以通过 v-on="$listeners" 传入内部组件——在创建更高层次的组件时非常有用 $attrs可以通过 v-bind="$attrs" 传入内部组件——在创建透明的包裹组件时非常有用。 一起用就可以A→b->c，（子组件）直接传值

2.renderMixin

2.1installRenderHelpers(Vue.prototype)

2.2Vue.prototype.$nextTick

2.3Vue.prototype.\_render

在instance的inject.js

1.initProvide

2.initInjections resolve injections before data/props？？？

3.resolveInject

在instance的state.js

1.initState

2.initProps

3.initData

4.getData

5.initComputed

6.defineComputed

7.stateMixin

7.1Vue.prototype.$set = set

7.2Vue.prototype.$delete = del

7.3Vue.prototype.$watch

8.initWatch

9.initMethods

10.vm.$mount(vm.$options.el)