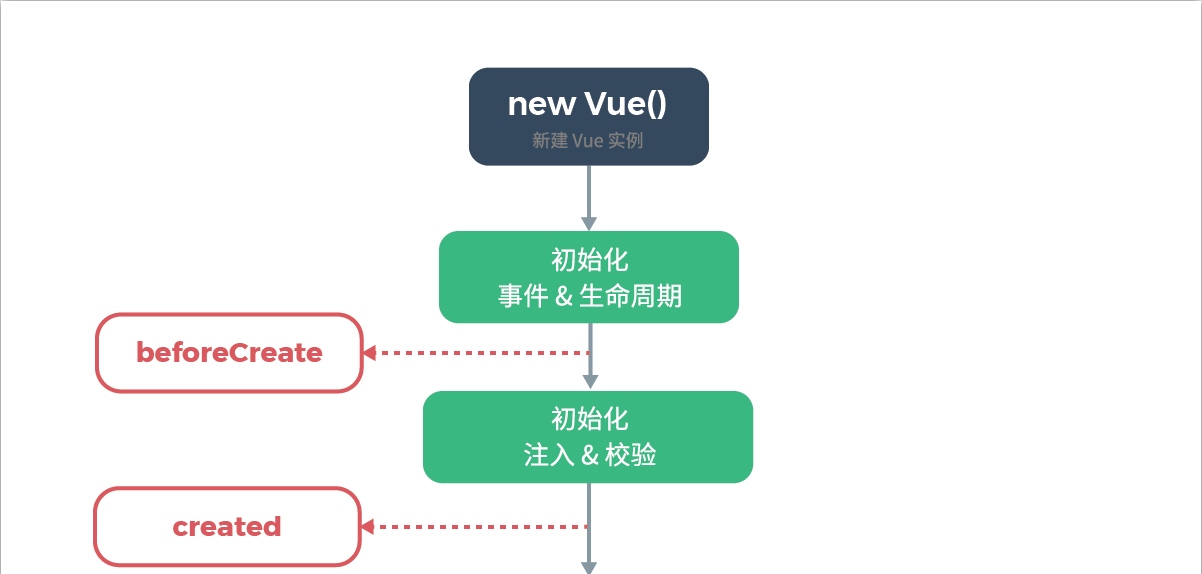
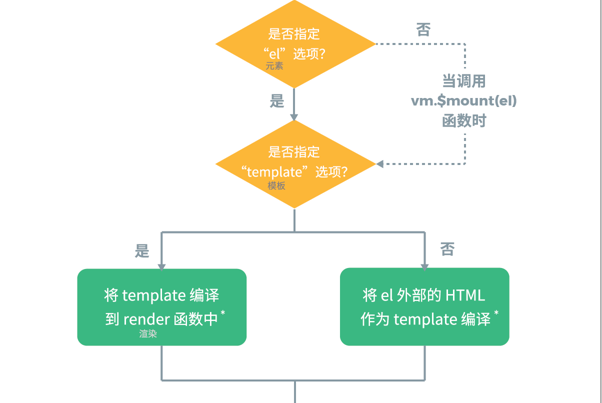
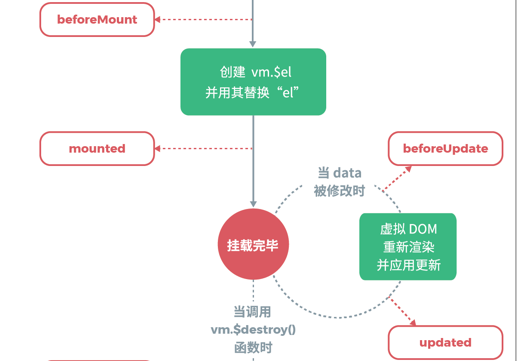
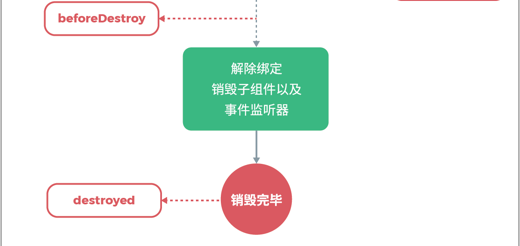
# Vue生命周期

### vue生命周期

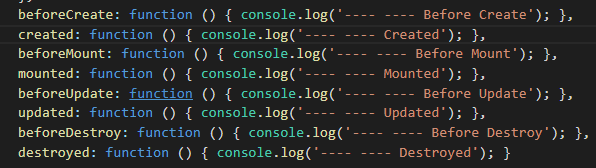








一个 vue对象从创建到销毁，提供了8个不同钩子函数，而这几个钩子函数降vue的生命周期划分了几个部分。



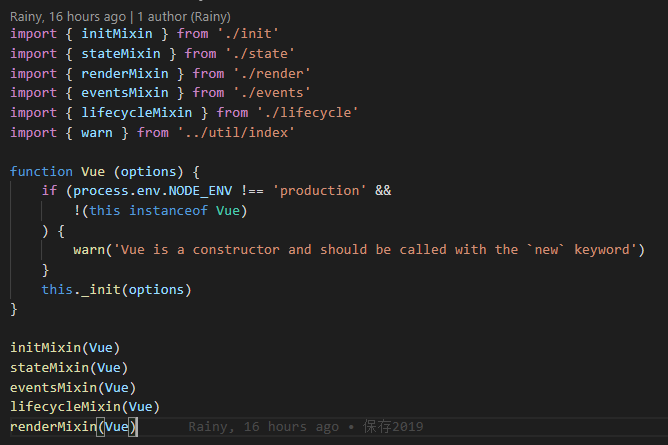
其中在前两个阶段是准备阶段

中间两个阶段是挂载编译阶段 ====》 实现view层和moldel层绑定

数据更新的两个钩子会在数据变化的两个阶段调用。

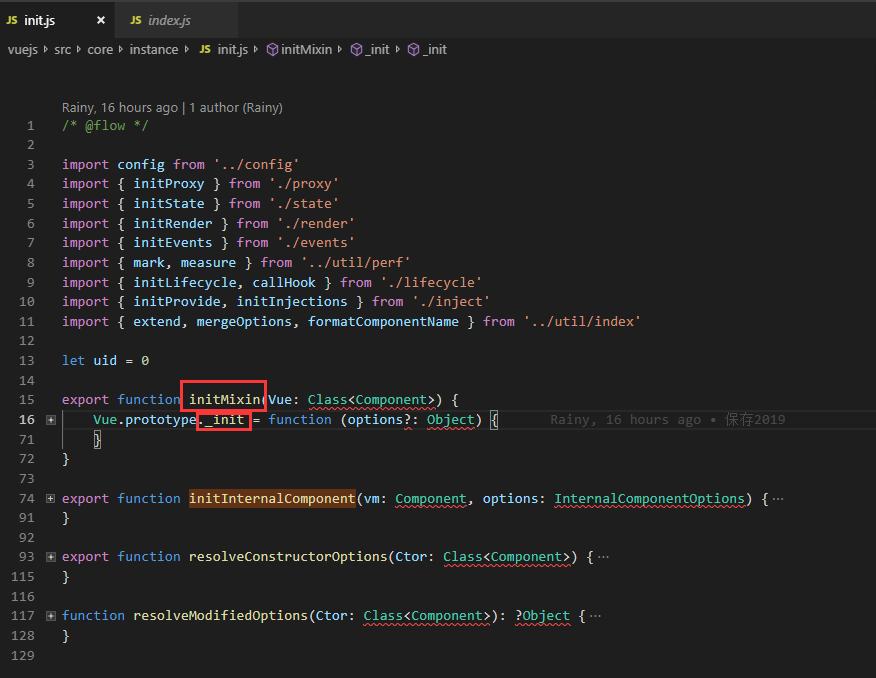
最后两个阶段会在vue实例销毁的时候调用。

### Vue 构造函数



简单分析一下各部分的作用。

InitMIxin (Vue)



给Vue构造函数的原型对象增加了一个\_init();//具体干了什么后面再看。

stateMixin(Vue):



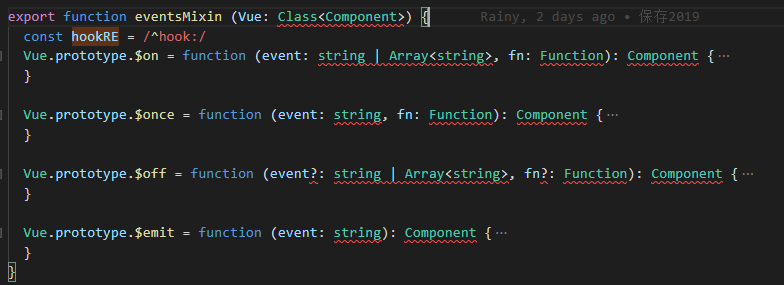
给Vue的原型对象

增加 $data,$props两个属性，通过设置了get分别从实例的\_data,\_props取值。

增加 $set $delete 动态往data上增肌和删除双向绑定监听的数据。

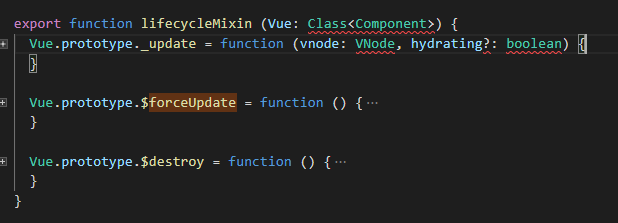
增加 $watch 动态设置监听数据变化的回调。

eventsMixin(Vue):



同样给Vue的原型增加四个方法，$on,$once,$off,$emit,分别对应绑定自顶替事件，触发一次的绑定自定义事件，解除自定义事件绑定，和发送（触发）自定义事件

lifecycleMixin(Vue):

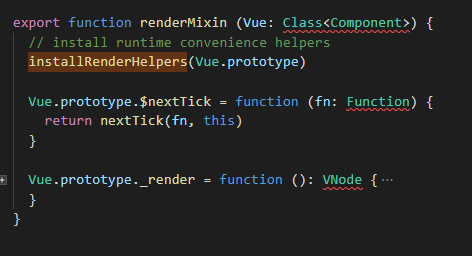


同样也给Vue的原型加了上方法，其中\_update,vue中约定\_开头的方法是私有方法，不是暴露出来的方法我们暂时不看，后面用到看这部分。

$forceUpate:强制按最新的数据更新视图。例：显示一个数据列表，改变数组第一个元素的某个属性，vue可能监听不到变化，官方建议通过$set改变，但是如果我们还是arr[0].X=’X’去改变，之后使用这个函数，强制刷新视图也可以达到效果。并触发beforeUpdate和updated钩子函数。

$destroy:删除当前实例，并触发beforeDestroy和dedestroyed 钩子函数。

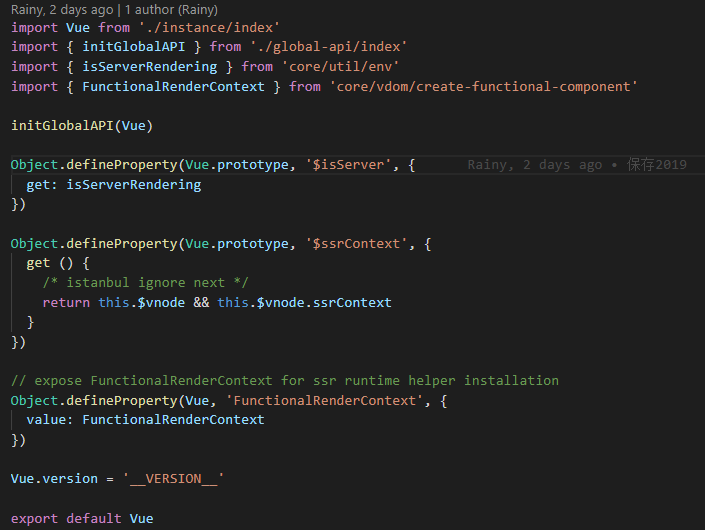
renderMixin(Vue):

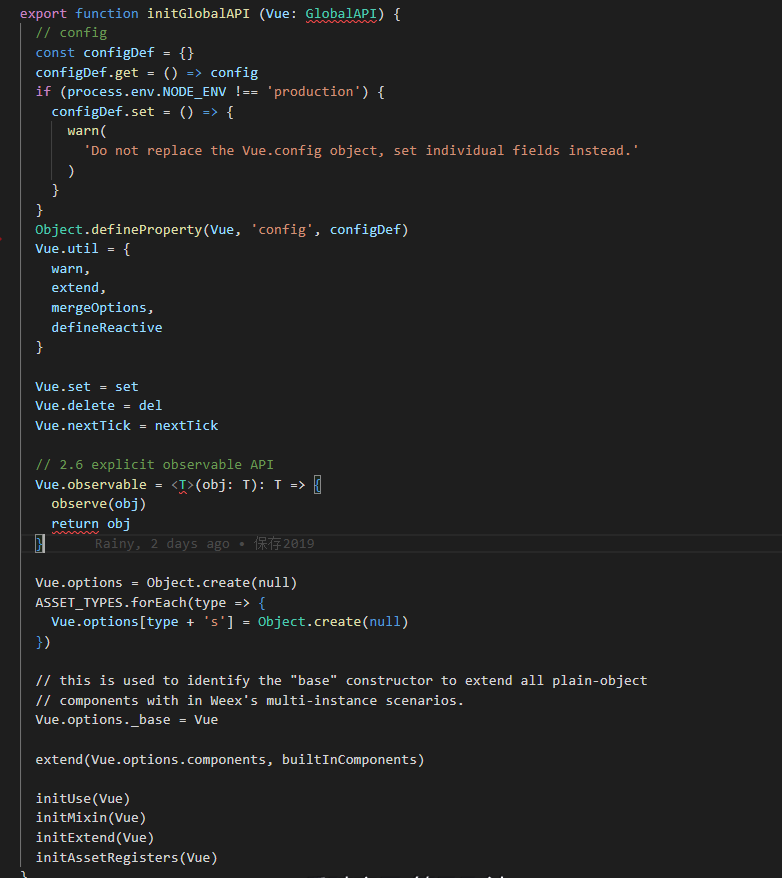


\_render 和 installRenderHelpers函数都是向Vue原型增加的内部方法。

$nextTick:将回调延迟到下次 DOM 更新循环之后执行，如果有异步操作，这个函数会在所有同步操作之后 异步之前（Updated之后）调用。

最后 将Vue exprot出去的时候，





initGlobalAPI(Vue):给Vue构造函数上增加了一些静态属性和静态方法。

Config: 一些配置（dev or product）

Set，del,设置（删除u）对应属性值并监听变化。

Nextick，observable，

初始化 options

并将属性components,directives,filters 初始化为null，

然后将options .components 默认加上 KeepAlive

并将options .\_base 设置为Vue.

给Vue加上静态方法 use，mixin，extend，component,directive,filter.

$isServer:判断代码是否运行在服务器上；

$ssrContext:

### vue实例的准备阶段，即 beforeCreate和created。

现在我们按照官方提供的生命周期图示看看从vue实例创建到销毁中间都做了那些事情。

首先是通过new来调用Vue函数 到beforeCreated钩子被调用这中间，

初始化事件和生命周期。

Vue构造函数：

function Vue (options) {

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

!(this instanceof Vue)

) {

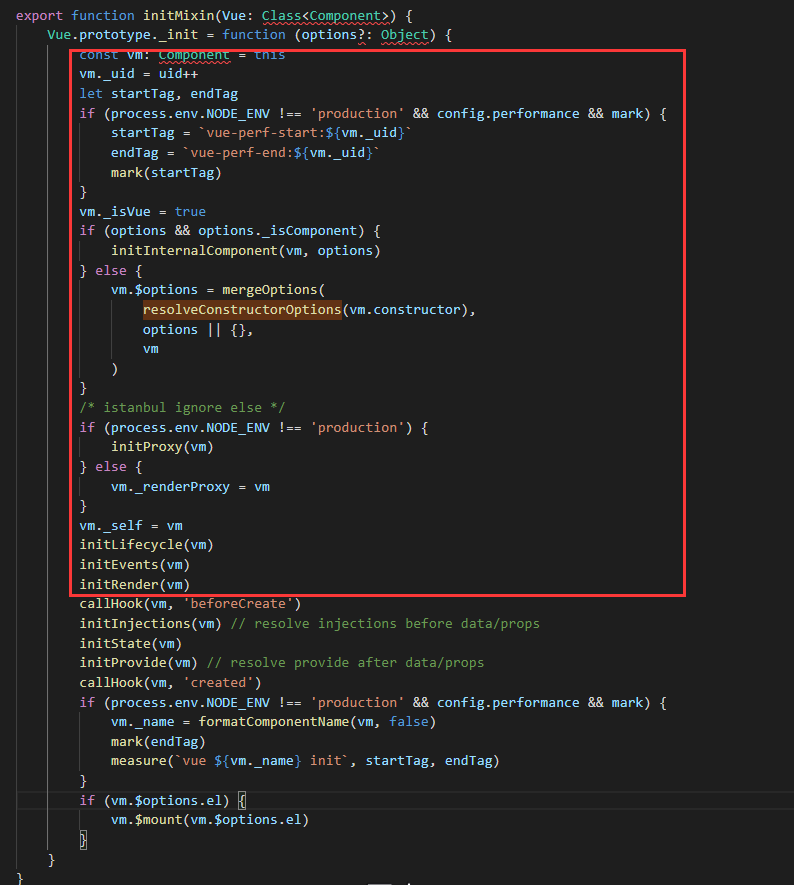
warn('Vue is a constructor and should be called with the `new` keyword')

}

this.\_init(options)

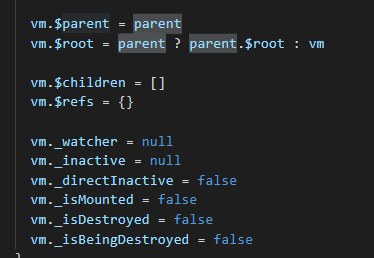
}

在new Vue(opts) 这个过程中，会首先调用\_init方法，并将接受的参数传入进去。



在\_init中，除了参数和环境的判断之后，执行了三个initLifeCyle(vm),initEvent(vm),initRender(vm),按照字面理解，分别是初始化生命周期，初始化事件和初始化渲染器。

initLifeCyle(vm)：



初始化生命周期相关的参数。

$parent 指定已创建的实例之父实例，在两者之间建立父子关系。子实例可以用 this.访问父实例，子实例被推入父实例的children 数组中。

$root 当前组件树的根 Vue 实例。如果当前实例没有父实例，此实例将会是其自己。

$children 当前实例的直接子组件。需要注意 $children 并不保证顺序，也不是响应式的。

$refs 一个对象，持有已注册过 ref 的所有子组件。

\_watcher 组件实例相应的 watcher 实例对象。

\_inactive 表示keep-alive中组件状态，如被激活，该值为false,反之为true。

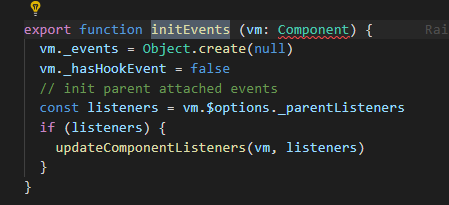
\_directInactive 也是表示keep-alive中组件状态的属性。

\_isMounted 当前实例是否完成挂载(对应生命周期图示中的mounted)。

\_isDestroyed 当前实例是否已经被销毁(对应生命周期图示中的destroyed)。

\_isBeingDestroyed 当前实例是否正在被销毁,还没有销毁完成(介于生命周期图示中deforeDestroy和destroyed之间)。

InitEvents(vm):



初始化事件参数。

\_event: 储存调用@绑定的时间和当前实例使用on绑定的时间。

\_hasHookEvent: 是否有@hook：created 这种方式调用生命周期钩子函数。