Vue全局API原理

- 获取全局配置 config
- 不允许通过 Vue.config = config 直接赋值
  将 config 代理到 Vue 实例上，可以通过 this.config 来访问
- 向 Vue 实例暴露出一些工具方法

  //日志
  warn,
  //将 A 对象上的属性 复制到 B 对象上
  extend,
  //合并选项
  mergeOptions,
  //给对象设置 getter setter，涉及到依赖收集 更新 触发 通知
  defineReactive
- Vue.set
  - //处理数组的情况
    判断是否为数组
    利用数组的 splice 方法实现动态插入并且返回 val 值
  - //处理对象的情况
    直接将值添加到要添加的对象上
  - // 对新属性设置 getter setter 读取时 收集依赖 更新时触发依赖通知更新
    defineReactive(ob.value, key, val)
  - // 进行 依赖通知
    ob.dep.notify()
    返回 val 值
- Vue.del
  - //处理数组的情况
    处理数组的情况还是用数组的 splice 方法实现
  - //处理对象的情况
    用 Object 修饰符 delete 实现
  - 进行依赖通知
- Vue.nextTick — 参见前文：Vue异步更新
- 向外暴露出响应式方法 observe
- 给 Vue 全局配置上的 component directive filter 选项
  将 Vue 构造函数赋值给 Vue.options._base — 遍历 ASSET_TYPES ，为 Vue 实例添加对应的属性
- 将 keep-alive 组件放到每个组件上
- initUse
  - 缓存插件
    判断 plugin 是否是一个对象，如果是执行 plugin.install
    判断 plugin 是否是一个函数，如果是直接执行函数
    将完成 install 的插件添加到 installedPlugins 保证同一个插件不会重复注册
- initMixin — 利用 mergeOptions 合并选项
- initExtend
  - 获取被继承组件的，以及被继承组件的 id

    // 用同一个配置项 多次调用 Vue.extend 时会调用,第二次调用开始时就会使用缓存
    const cachedCtors = extendOptions._Ctor || (extendOptions._Ctor = {})

    验证组件的名称

    定义一个 Vue 子类 [过程和初始化 Vue 一致 Vue._init]

    设置子类的原型对象，构造函数，和子类 uid

    将被继承组件放到子类的 super 属性上

    初始化子类的 prop 和 computed

    // 让子类支持继续向下拓展
    Sub.extend = Super.extend
    Sub.mixin = Super.mixin
    Sub.use = Super.use

    // 给子类设置 component filter directive

    //组件自调用核心原理
    if (name) {
    Sub.options.components[name] = Sub
    }
    // 给子类设置基类的选项
    Sub.superOptions = Super.options
    Sub.extendOptions = extendOptions
    Sub.sealedOptions = extend({}, Sub.options)

    缓存子类并且返回子类
- initAssetRegisters
  - 判断是否有组件构造函数或者配置对象，如果不是
    return this.options[type + 's'][id]
    如果是

    type === components
    设置组件的 name 属性，有就用 name 没有就用 id
    基于 definition 去拓展一个新的组件子类，直接 new definition() 实例化一个组件

    type === directive
    definition = { bind: definition, update: definition }

    返回 definition