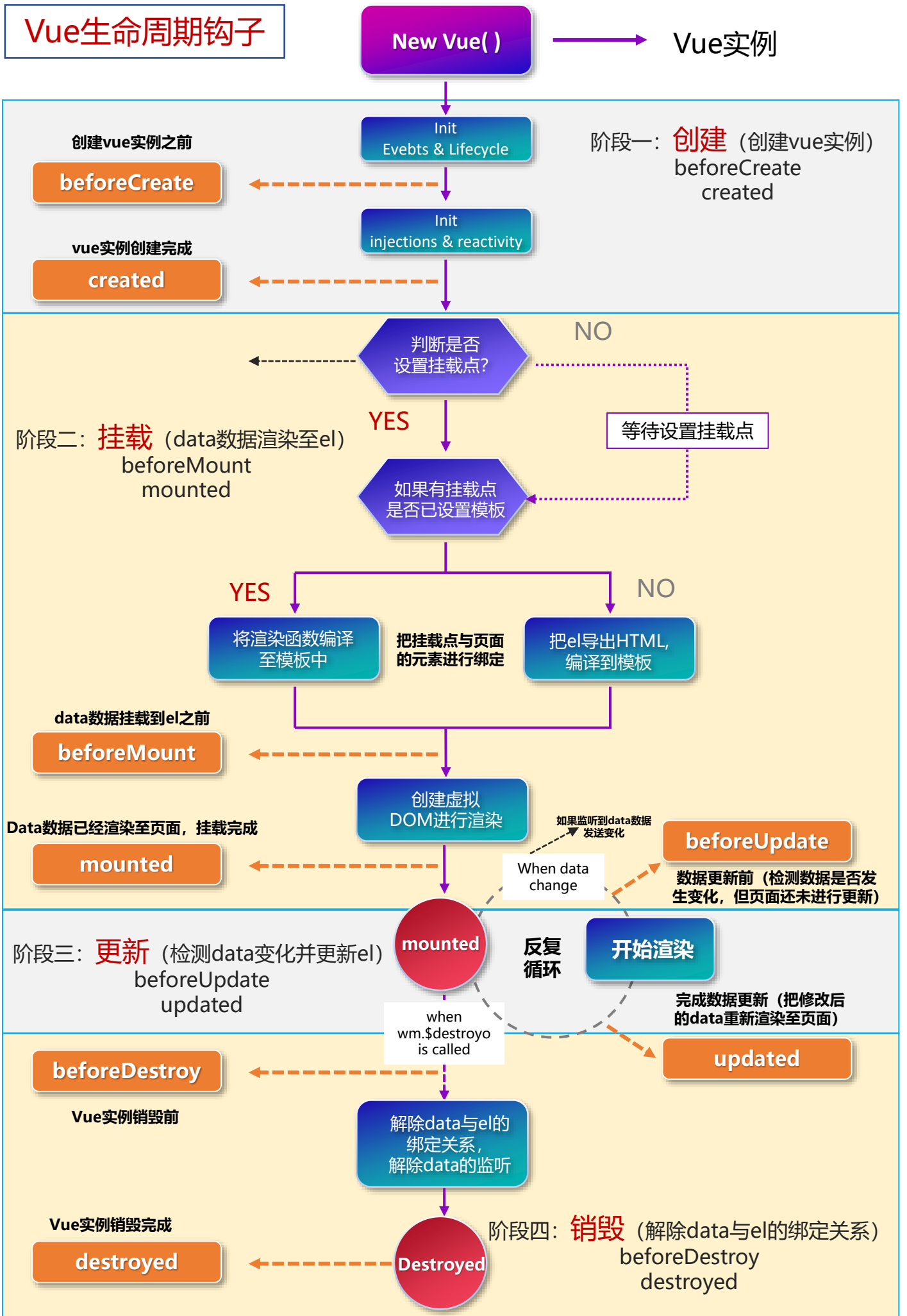
Vue生命周期钩子


New Vue( )
Vue实例
Init
Evebts & Lifecycle
阶段一：创建（创建vue实例）
beforeCreate
created
创建vue实例之前
beforeCreate
Init
injections & reactivity
vue实例创建完成
created
判断是否
设置挂载点?
NO
YES
等待设置挂载点
阶段二：挂载（data数据渲染至el）
beforeMount
mounted
如果有挂载点
是否已设置模板
YES
NO
将渲染函数编译
至模板中
把挂载点与页面
的元素进行绑定
把el导出HTML,
编译到模板
data数据挂载到el之前
beforeMount
创建虚拟
DOM进行渲染
Data数据已经渲染至页面，挂载完成
mounted
如果监听到data数据
发送变化
beforeUpdate
When data
change
数据更新前（检测数据是否发
生变化，但页面还未进行更新）
mounted
反复
循环
开始渲染
阶段三：更新（检测data变化并更新el）
beforeUpdate
updated
完成数据更新（把修改后
的data重新渲染至页面）
when
wm.$destroyo
is called
updated
beforeDestroy
Vue实例销毁前
解除data与el的
绑定关系，
解除data的监听
Vue实例销毁完成
destroyed
Destroyed
阶段四：销毁（解除data与el的绑定关系）
beforeDestroy
destroyed

事件循环 Event Loop

01、同步任务

02、异步任务

03、异步微任务

3.1、同步

3.2、异步微任务

3.3、异步宏任务

04、异步宏任务

4.1、同步

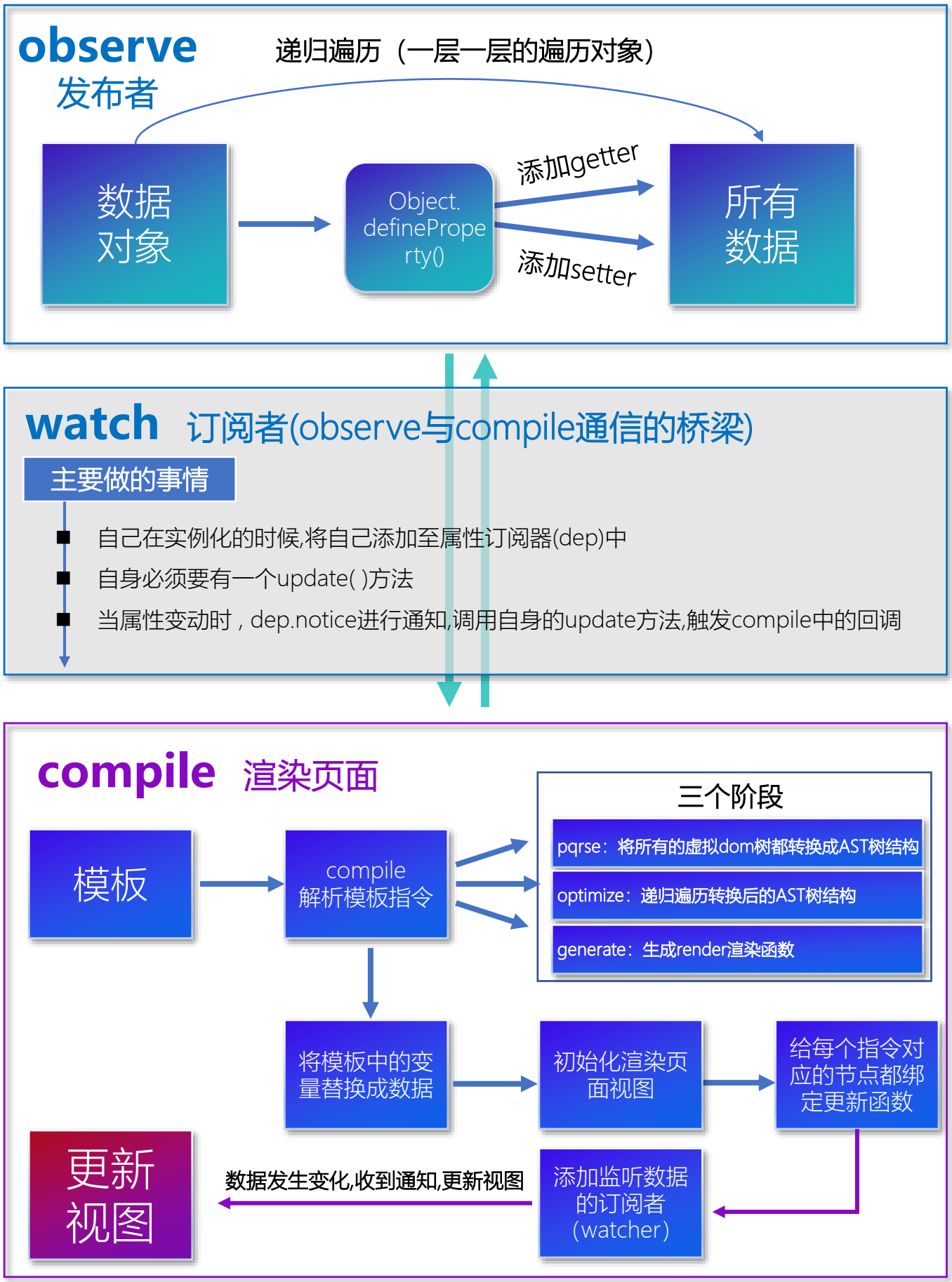
4.2、异步微任务

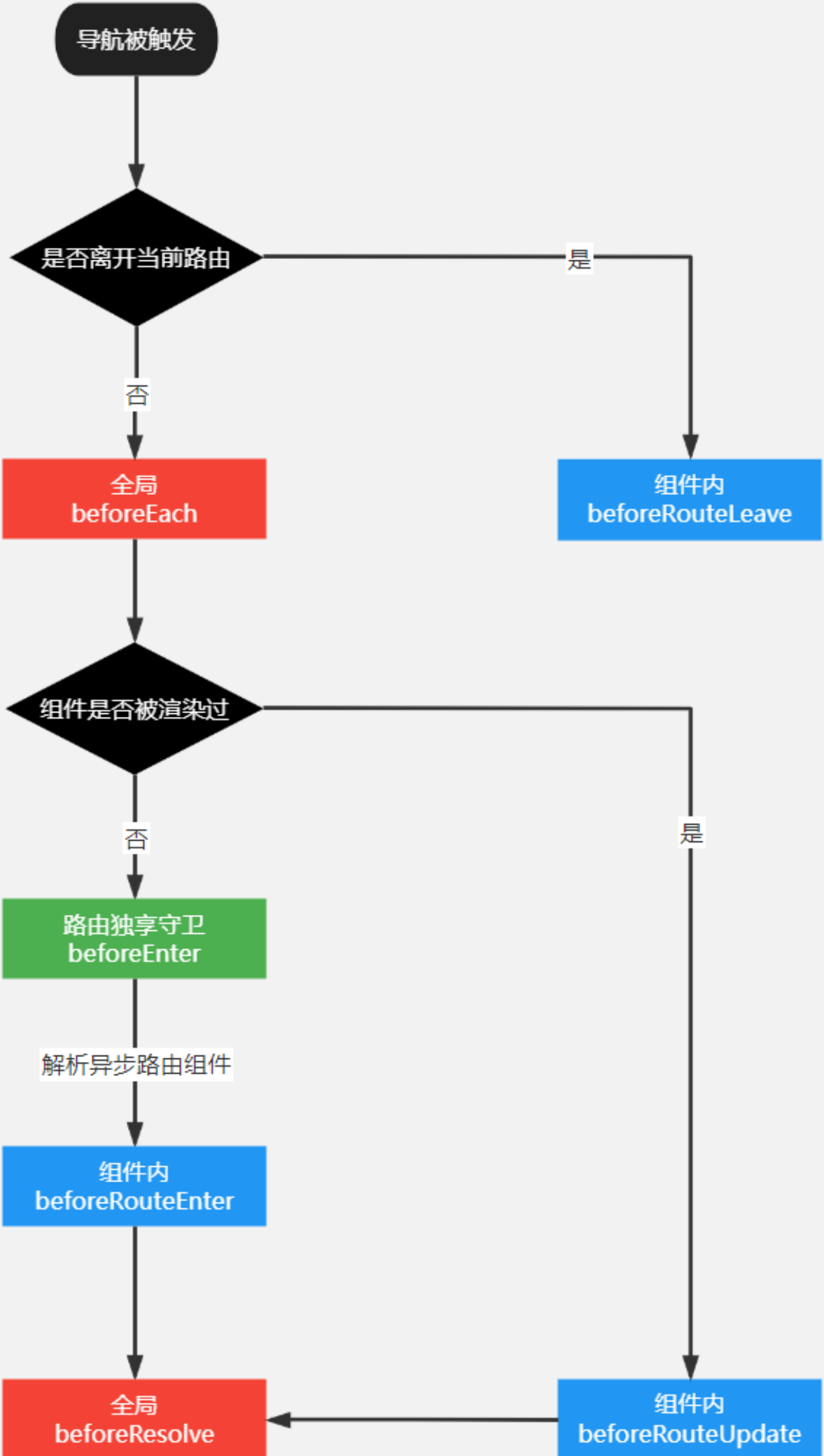
4.3、异步宏任务

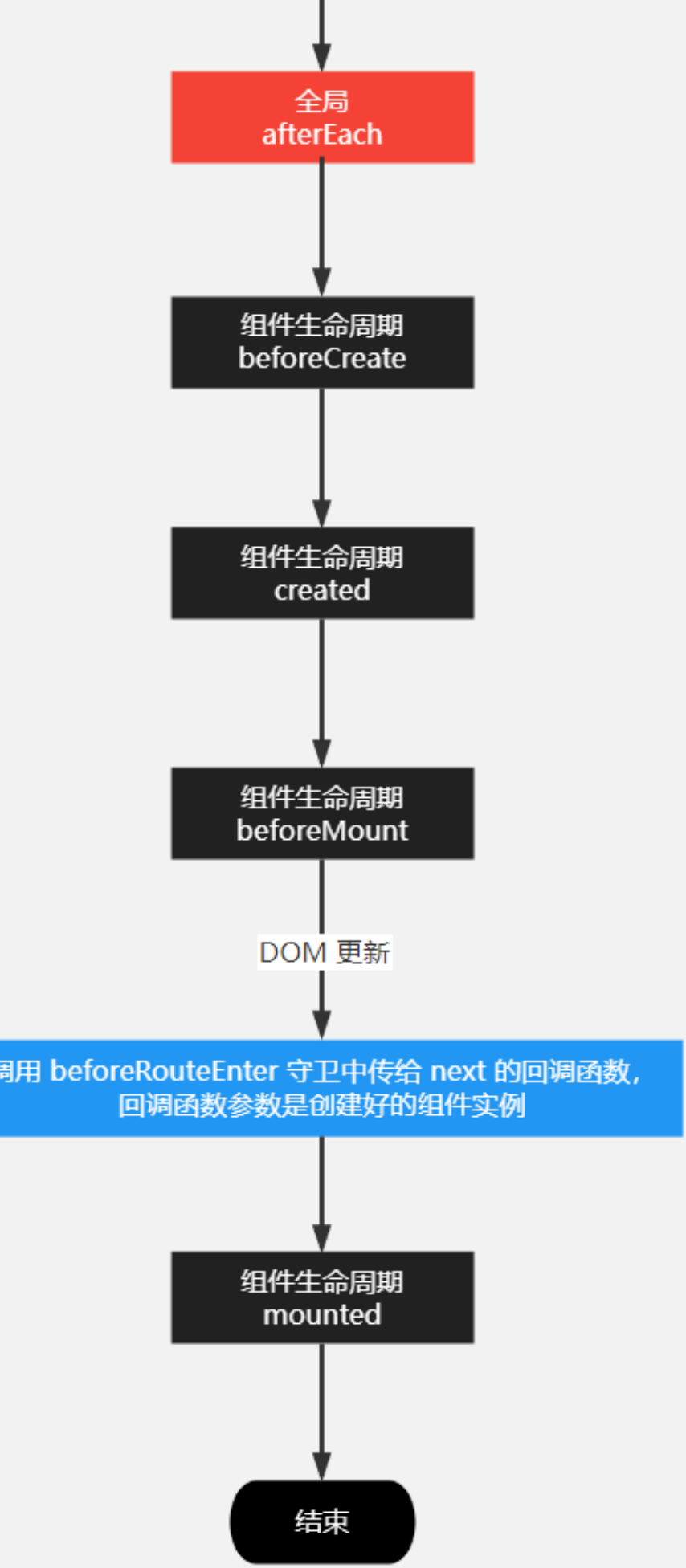
```
// 1.设置基地址
const BASE_URL='https://api-hmugo-
web.itheima.net'

async function request({url, header = {},
method, data}){
  // 添加loading提示\
  uni.showLoading({
    title:'loading...',
    mask:true
  })
  let [error,res] = await uni.request({
    url: BASE_URL + url,
    header,
    method,
    data
  })
    // 3.关闭loading
  uni.hideLoading()
    if (!error) {
    // 4.返回更加友好的结果格式
          let _data = { msg: res.data.meta,
data: res.data.message }
        return _data
    } else {
      return new Error(error)
    }
}
// vue 插件形式
// 1. 定义插件
const MyRequest = {
  install (Vue, opts) {
    Vue.prototype.request = request
  }
}
export default MyRequest
```

VUE2响应式原理


observe
发布者
递归遍历（一层一层的遍历对象）
数据对象
Object.defineProperty()
添加getter
添加setter
所有数据
watch
订阅者(observe与compile通信的桥梁)
主要做的事情
自己在实例化的时候,将自己添加至属性订阅器(dep)中
自身必须要有一个update( )方法
当属性变动时，dep.notice进行通知,调用自身的update方法,触发compile中的回调
compile
渲染页面
模板
compile
解析模板指令
三个阶段
pqrse：将所有的虚拟dom树都转换成AST树结构
optimize：递归遍历转换后的AST树结构
generate：生成render渲染函数
将模板中的变量替换成数据
初始化渲染页面视图
给每个指令对应的节点都绑定更新函数
更新视图
数据发生变化,收到通知,更新视图
添加监听数据的订阅者(watcher)


导航被触发
是否离开当前路由
是
否
全局
beforeEach
组件内
beforeRouteLeave
组件是否被渲染过
是
否
路由独享守卫
beforeEnter
解析异步路由组件
组件内
beforeRouteEnter
全局
beforeResolve
组件内
beforeRouteUpdate


全局
afterEach
组件生命周期
beforeCreate
组件生命周期
created
组件生命周期
beforeMount
DOM 更新
调用 beforeRouteEnter 守卫中传给 next 的回调函数，
回调函数参数是创建好的组件实例
组件生命周期
mounted
结束

state

储存全局使用的数据

一般存储token和用户信息

mutations

放置同步任务

用来改变state里的数据

Mutations里的函数第一个参数就是vuex中的state

actions

放置异步任务

改变state的数据

需要context.commit(‘事件名’,传递的参数)

Actions里的函数第一个参数context指向的就是整个vuex

Action里的函数在其他部分调用，可以使用this.$store.dispatch(‘事件名’

getters

modules

Generate