2-3 new Vue 发生了什么-文档

## **new Vue 发生了什么**

# **new Vue 发生了什么**

从入口代码开始分析，我们先来分析 new Vue 背后发生了哪些事情。我们都知道，new 关键字在 Javascript 语言中代表实例化是一个对象，而 Vue 实际上是一个类，类在 Javascript 中是用 Function 来实现的，来看一下源码，在src/core/instance/index.js 中。

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)

}

可以看到 Vue 只能通过 new 关键字初始化，然后会调用 this.\_init 方法， 该方法在 src/core/instance/init.js 中定义。

Vue.prototype.\_init = function (options?: Object) {

const vm: Component = this

// a uid

vm.\_uid = uid++

let startTag, endTag

/\* istanbul ignore if \*/

if (process.env.NODE\_ENV !== 'production' && config.performance && mark) {

startTag = `vue-perf-start:${vm.\_uid}`

endTag = `vue-perf-end:${vm.\_uid}`

mark(startTag)

}

// a flag to avoid this being observed

vm.\_isVue = true

// merge options

if (options && options.\_isComponent) {

// optimize internal component instantiation

// since dynamic options merging is pretty slow, and none of the

// internal component options needs special treatment.

initInternalComponent(vm, options)

} else {

vm.$options = mergeOptions(

resolveConstructorOptions(vm.constructor),

options || {},

vm

)

}

/\* istanbul ignore else \*/

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

initProxy(vm)

} else {

vm.\_renderProxy = vm

}

// expose real self

vm.\_self = vm

initLifecycle(vm)

initEvents(vm)

initRender(vm)

callHook(vm, 'beforeCreate')

initInjections(vm) // resolve injections before data/props

initState(vm)

initProvide(vm) // resolve provide after data/props

callHook(vm, 'created')

/\* istanbul ignore if \*/

if (process.env.NODE\_ENV !== 'production' && config.performance && mark) {

vm.\_name = formatComponentName(vm, false)

mark(endTag)

measure(`vue ${vm.\_name} init`, startTag, endTag)

}

if (vm.$options.el) {

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

}

}

Vue 初始化主要就干了几件事情，合并配置，初始化生命周期，初始化事件中心，初始化渲染，初始化 data、props、computed、watcher 等等。

## **总结**

Vue 的初始化逻辑写的非常清楚，把不同的功能逻辑拆成一些单独的函数执行，让主线逻辑一目了然，这样的编程思想是非常值得借鉴和学习的。

由于我们这一章的目标是弄清楚模板和数据如何渲染成最终的 DOM，所以各种初始化逻辑我们先不看。在初始化的最后，检测到如果有 el 属性，则调用 vm.$mount 方法挂载 vm，挂载的目标就是把模板渲染成最终的 DOM，那么接下来我们来分析 Vue 的挂载过程。

****任务****

请仔细阅读本文档，为下面学习视频内容做准备。