Instance/init.js中

vm.$mount(vm.$options.el) //如果 Vue 实例在实例化时没有收到 el 选项，则它处于“未挂载”状态，没有关联的 DOM 元素。可以使用 vm.$mount() 手动地挂载一个未挂载的实例。

Platforms/web/entry-runtime-with-compiler.js

const mount = Vue.prototype.$mount

Vue.prototype.$mount = function (

el?: string | Element,

hydrating?: boolean

): Component {

el = el && query(el)

//获取模板（template）

const options = this.$options

// resolve template/el and convert to render function

if (!options.render) {

let template = options.template

if (template) {

if (typeof template === 'string') {

if (template.charAt(0) === '#') {

template = idToTemplate(template)

/\* istanbul ignore if \*/

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

warn(

`Template element not found or is empty: ${options.template}`,

this

)

}

}

} else if (template.nodeType) {

template = template.innerHTML

} else {

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

warn('invalid template option:' + template, this)

}

return this

}

} else if (el) {

template = getOuterHTML(el)

}

if (template) {

const { render, staticRenderFns } = compileToFunctions(template, {

shouldDecodeNewlines,

shouldDecodeNewlinesForHref,

delimiters: options.delimiters,

comments: options.comments

}, this)

options.render = render

options.staticRenderFns = staticRenderFns

/\* istanbul ignore if \*/

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

mark('compile end')

measure(`vue ${this.\_name} compile`, 'compile', 'compile end')

}

}

}

return mount.call(this, el, hydrating)

}

compileToFunctions

export const createCompiler = createCompilerCreator(function baseCompile (

template: string,

options: CompilerOptions

): CompiledResult {

const ast = parse(template.trim(), options)

if (options.optimize !== false) {

optimize(ast, options)

}

const code = generate(ast, options)

return {

ast,

render: code.render,

staticRenderFns: code.staticRenderFns

}

})

1.合并options 在create-compiler.js

const finalOptions = Object.create(baseOptions)

const errors = []

const tips = []

finalOptions.warn = (msg, tip) => {

(tip ? tips : errors).push(msg)

}

if (options) {

// merge custom modules

if (options.modules) {

finalOptions.modules =

(baseOptions.modules || []).concat(options.modules)

}

// merge custom directives

if (options.directives) {

finalOptions.directives = extend(

Object.create(baseOptions.directives || null),

options.directives

)

}

// copy other options

for (const key in options) {

if (key !== 'modules' && key !== 'directives') {

finalOptions[key] = options[key]

}

}

}

const compiled = baseCompile(template, finalOptions)

baseCompile是 在compiler/index.js

function baseCompile (

template: string,

options: CompilerOptions

): CompiledResult {

const ast = parse(template.trim(), options)

if (options.optimize !== false) {

optimize(ast, options)

}

const code = generate(ast, options)

return {

ast,

render: code.render,

staticRenderFns: code.staticRenderFns

}

}

2.构建AST树 const ast = parse(template.trim(), options)

parse在compiler/parser中

3.optimize(优化) optimize(ast, options)

optimize在compiler/optimizer.js

4.generate（生成with语句） const code = generate(ast, options)

在compiler/codegen/index

5.返回对象

return {

ast,

render: code.render,

staticRenderFns: code.staticRenderFns

}

6.mount.call(this, el, hydrating)

mount是 platform/web/runtime/index.js

Vue.prototype.$mount = function (

el?: string | Element,

hydrating?: boolean

): Component {

el = el && inBrowser ? query(el) : undefined

return mountComponent(this, el, hydrating)

}

7.mountComponent 在lifeCycle.js中 就是挂载的过程

updateComponent = () => {

vm.\_update(vm.\_render(), hydrating)

}

new Watcher(vm, updateComponent, noop, null, true /\* isRenderWatcher \*/)

8.vm.\_render()是with语句生成虚拟节点（vnode）的过程

在render.js中

Vue.prototype.\_render = function (): VNode {

const vm: Component = this

const { render, \_parentVnode } = vm.$options

// reset \_rendered flag on slots for duplicate slot check

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

for (const key in vm.$slots) {

// $flow-disable-line

vm.$slots[key].\_rendered = false

}

}

if (\_parentVnode) {

vm.$scopedSlots = \_parentVnode.data.scopedSlots || emptyObject

}

// set parent vnode. this allows render functions to have access

// to the data on the placeholder node.

vm.$vnode = \_parentVnode

// render self

let vnode

try {

vnode = render.call(vm.\_renderProxy, vm.$createElement)

} catch (e) {

handleError(e, vm, `render`)

// return error render result,

// or previous vnode to prevent render error causing blank component

/\* istanbul ignore else \*/

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

if (vm.$options.renderError) {

try {

vnode = vm.$options.renderError.call(vm.\_renderProxy, vm.$createElement, e)

} catch (e) {

handleError(e, vm, `renderError`)

vnode = vm.\_vnode

}

} else {

vnode = vm.\_vnode

}

} else {

vnode = vm.\_vnode

}

}

// return empty vnode in case the render function errored out

if (!(vnode instanceof VNode)) {

if (process.env.NODE\_ENV !== 'production' && Array.isArray(vnode)) {

warn(

'Multiple root nodes returned from render function. Render function ' +

'should return a single root node.',

vm

)

}

vnode = createEmptyVNode()

}

// set parent

vnode.parent = \_parentVnode

return vnode

}

9.new Watcher(vm, updateComponent, noop, null, true /\* isRenderWatcher \*/)

在observe/watcher.js

是一个class

this.getter = updateComponent

get()方法会触发this.getter()

10. vm.\_update() lifecycle

if (!prevVnode) {

// initial render

vm.$el = vm.\_\_patch\_\_(

vm.$el, vnode, hydrating, false /\* removeOnly \*/,

vm.$options.\_parentElm,

vm.$options.\_refElm

)

// no need for the ref nodes after initial patch

// this prevents keeping a detached DOM tree in memory (#5851)

vm.$options.\_parentElm = vm.$options.\_refElm = null

} else {

// updates

vm.$el = vm.\_\_patch\_\_(prevVnode, vnode)

}

11.vm.\_\_patch\_\_()

Vue.prototype.\_\_patch\_\_ = inBrowser ? patch : noop 在platforms/web/runtime/index.js

12 patch 在platforms/web/runtime/patch.js

export const patch = createPatchFunction({ nodeOps, modules })

13. createPatchFunction 在core/vdom/patch.js 把vnode生成dom节点 appendChild到body上

return patch() //684行