# **1.生命周期函数**

### **1.1 什么是生命周期函数**

我理解的生命周期函数就是 vue 实例从创建到销毁的过程中，到了某一个特定的时间节点会自动触发的钩子函数 常用的钩子函数一共有八个 beforeCreate 创建前 实例还未创建 created 创建后 这个第一个能拿到 date 和 methods 来用的钩子函数 beforeMount 挂载前 挂载的是节点 mounted 挂载后 以上四个是创建阶段的钩子函数 这些钩子函数只会执行 1 次

beforeUpdate 数据变了视图未更新 updated 数据变了 视图更新 更新阶段的钩子函数会执行 0 次或者无数次

beforeDestory 销毁之前 最后一个能使用 data 和 methods 的钩子函数 destoryed 销毁之后 销毁阶段的钩子函数 执行 1 次

除了这八个还有三个 actived 激活缓存组件 deactived 停用缓存组件 errorCaptured 子组件出错的时候触发

### **1.2 在项目开发过程中哪些钩子函数用的比较多**

在写代码的过程中，我用的比较多的钩子函数有 cretaed 和 mounted 在 created 中我经常调用 axios 请求数据 或者获取本地存储里的数据 在 mounted 中我经常获取 dom 节点

### **1.3 父组件和子组件的生命周期函数的执行顺序**

父组件的创建前 父组件的创建后 父组件的挂载前 子组件的创建前 子组件的创建后 子组件的挂载前 子组件的挂载后 父组件的挂载后

# **2.nextTick**

在 dom 更新之后执行的延迟回调 ，在 vue 中 dom 的更新是异步的，我理解的 nextTick 是把本身同步的代码变成异步执行，排在 dom 更新之后，所以能拿到 dom 节点

# **3.vuex**

### **3.1 什么是 vuex 能解决什么问题**

vuex 是 vue 的状态管理工具，能解决组件之间数据共享的问题

### **3.2 五大核心以及怎么调用**

state 存放数据的地方 this.$store.state.xxx mutations 修改state的地方 this.$store.commit() getters 计算 state 的地方 this.$store.getters.xxx actions 发送异步请求的地方 this.$store.dispatch() modules 模块化 把数据分模块管理

### **3.3 vuex 的语法糖辅助函数**

mapState mapMutations mapActions mapGetters

先引入 然后展开使用 mapState 和 mapGetters 在计算属性下展开 mapActions 和 mapMutations 在 methods 下展开 展开的时候用扩展运算符...展开

### **3.4 vuex 的执行机制**

如果要修改 state，先通过 dispatch 调用一个 action 再在 action 里通过 commit 调用一个 mutations 修改 state

### **3.5 vuex 的弊端**

不过 vuex 也有⼀些弊端,⽐如浏览器刷新的时候,vuex 的数据会丢失,我们⼀般结合本地存储来解决,当我们在 mutations ⾥⾯改变 state 的时候在通过 localStorage 或者 sessionStorage 或者持久化插件存储到本地,然后在 state 的状态的属性值那块写⼀个三元表达式,如果本地存 在数据的话则读取本地存储的数据,否则就赋值为 null

### **3.6 怎么开启严格模式**

跟 state 同级的地方 设置 strict: false, 如果开启了严格模式 只能在 mutations 中修改 state 否则就会报错

### **3.7 modules**

当在 vuex 中存储的数据很多的时候 需要用 modules 来分模块管理 首先新建一个模块的 js 文件 里面有四大核心 state mutations actions getters 并且要开启一个命名空间 namespaced:true 然后在 index.js 中引入 这个 js 并在 modules 中注册模块 用的时候 this.$store.state.a.arr 这就是调用模块a下面的arr this.$store.commit("a/add") 这个就是调用模块 a 下面的 add 方法

# **4. vue 路由有⼏种模式?有什么区别?原理是什么?(必问) 讲解**

⾯试官您好,接下来我给您介绍⼀下 vue 的路由模式,vue 的路由模式⼀共有两种,分别是哈希和 history. 他们的区别是 hash 模式不 会包含在 http 请求当中,并且 hash 不会重新加载⻚⾯,⽽使⽤ history 模式的话,如果前端的 url 和后端发起请求的 url 不⼀致的话,会 报 404 错误,所以使⽤ history 模块的话我们需要和后端进⾏配合. history 的原理就是利⽤ html5 新增的两个特性⽅法,分别是 pushState 和 replaceState 来完成的. 以上就是我对 vue 路由模式的理解.

# **5.路由导航守卫**

### **5.1 什么是导航守卫**

路由的导航守卫 又叫做路由的钩子函数 或者路由的生命周期函数 是在页面跳转的过程中到了某一个特定的时间节点会触发的钩子函数 有三种七个 全局守卫 beforeEach 全局前置守卫 beforeResolve 路由解析之前 afterEach 全局后置守卫

组件级守卫 beforeRouteEnter 路由进入之前 beforeRouteUpdate 路由更新之前 beforeRouteLeave 路由离开之前

单个路由独享的守卫 beforeEnter 路由进入之前

这些导航守卫有三个参数 to from next to 代表到哪儿去 from 代表从哪儿来 next 代表执行下一步

### **5.2 导航守卫的使用场景**

在写代码的过程中我一般用导航守卫进行页面的鉴权处理，在用户登陆成功的时候，我们会把 token 和用户信息存到 vuex 或者本地存储中，代表了用户的登陆状态，在访问某个页面的时候判断是否存的有 token，就能正常访问这个页面，如果没有 token，那么就用 next 跳转到登陆页面

### **5.3 导航守卫的执行顺序**

全局前置 单个路由规则的 组件前置守卫 全局解析守卫 后置守卫

### **5.4 beforeEach 和 beforeResolve 的区别**

这两个都是全局守卫 都是在路由跳转之前就会执行的守卫 beforeEach 比 beforeResolve 执行的时间要更早

# **6. v-if 与 v-show 的区别?(90%)**

⾯试官您好,接下来我给您介绍⼀下 v-if 和 v-show 的区别? ⾸先 v-if 和 v-show 都是控制元素的显示与隐藏, 不过 v-if 控制元素的显 示和隐藏的时候会删除对⽤的 dom 元素,当每⼀个显示的时候,都会重新创建 dom 和渲染. ⽽ v-show 则是通过 css 的 display:none 和 display:block 来控制元素的显示与隐藏. v-if ⽐较耗费性能,所以我们涉及到频繁的显示隐藏操作我们建议使⽤ v-show,如果不 是频繁操作的话,我们可以 v-if 在项⽬中我会经常使⽤ v-if 和 v-show,⽐如我们在搜索功能的时候,他有⼀个历史记录,这个时候我们根据是否有搜索的结果来判断历史记录的显示与隐藏,这块我就可以使⽤ v-if ,当然⽤ v-show 也可以. 以上就是我对 v-if 和 v-show 的理解.

# **7. v-for 与 v-if 的优先级那个⾼?如果同时使⽤ v-for 和 v-if 怎么解决?(90%) 讲解**

v-for 的优先级⾼. 因为 v-for 的时候我们才开始渲染 dom 元素,这个 v-if 还⽆法进⾏判断. v-for 和 v-if 不能同时使⽤,我们可以通过标签,⽐如 div 或者 template 标签来进⾏包裹,把 v-if 写到包裹的标签上⾯(写到 v-for 外⾯) 还有就是用计算属性算出来想要的数据再去循环 在 vue3 中 v-if 的优先级更高

# **8.methods computed watch 的区别**

methods 是方法 各种事件放在这里 没有缓存功能

computed 是计算属性 需要 return 计算的结果 并且有缓存功能 只有跟他有关的值发生变化才会重新计算 否则就从缓存里拿 通常用来计算总数和总价 比如购物车功能就会用到计算属性

watch 监听属性 有三个参数 handler（执行函数） deep（深度监听） immediate（立即监听） 如果不需要 deep 和 immediate 的时候就会简写成一个函数 watch 可以监听 data 数据还能监听路由的变化 watch 里可以执行异步操作 但是计算属性不行 watch 的使用场景 存本地存储的时候 只要数据变了就要重新存一次 那么就可以用 watch 实现 watch 监听路由的变化 是在有 router-view 的页面 watch 监听 this.$route 也是两个参数 newVal oldVal 新值就是到哪儿去 旧值就是从哪儿儿来

# **9. vue 常⽤的指令有哪些?(50%)**

v-if v-show v-html v-text v-on v-bind v-model v-for

# **10. vue 常⽤的修饰符有哪些?(50%)**

.trim 去除⾸尾多余的空格 .stop 阻⽌事件冒泡 .once 只渲染⼀次 .self 事件只作⽤在元素本身 .number 将值转化为 number 类型 .capter 组件之间捕获 .prevent 阻⽌元素的默认⾏为 .native 事件穿透,让我们可以在⾃定义组件上定义事件和⽅法

# **11.组件通信**

父传子 首先在子组件标签上绑定一个自定义属性，然后在子组件里通过 props 接收这个属性，props 接收有两种方式 数组和对象的形式 父传子的使用场景 封装列表的时候 把数据传到子组件渲染

子传父 在子组件的标签上绑定一个自定义事件，然后在子组件里通过$emit 调用这个自定义事件，调用这个事件的同时把要传的数据作为参数传到父组件 子传父的使用场景：在子组件中有删除等方法的时候要在子组件把下标传到父组件操作数据

兄弟组件传值 中央事件总线 eventbus 在 main.js 定义一个空的 vue 实例并且挂载在原型上，通过$emit传数据，在要接收的子组件的created钩子函数下通过$on 接收

vuex 把公共数据存在 vuex 里 这样就能实现组件通信

本地存储 把要传递的数据存在本地存储里 在要接收的组件接收 也能实现组件通信

v-model v-model 是一个语法糖 是:value 和 @input 的合写 所以说能实现组件通信

$attr 和 $listener 主要是祖孙之间传值 父组件下面有子组件a 子组件a下面又有 子组件b 在父组件里还是用自定义属性和自定义事件传 在子组件a下面绑定v-bind="$attrs" v-on="$listeners" 在子组件b下面通过$attrs 拿数据 通过 $emit 调用方法

provide 和 inject 不管嵌套了几层都能拿到数据 把数据定义在 provide 里 在子组件通过 inject 拿到数据 缺点是不响应

children 和 parent 在子组件里通过$parent 拿到父组件的数据和方法 在父组件里通过$children 拿到子组件的数据和方法

# **12.什么是 mvvm 模式？**

vue 是一个 mvvm 模式的框架 m 就是模型 model 在 vue 中指数据 v 就是 view 视图 vm 是 view-model 控制器 在 vue 中就是 vue 实例 视图变了数据会变 数据变了视图会变 这中间就是 vm 在控制

除了 mvvm 模式之外 还有 mvp 和 mvc 模式

MVVM 模式的优点： 1、低耦合： 视图（View）可以独⽴于 Model 变化和修改，⼀个 ViewModel 可以绑定到不同的"View"上，当 View 变化的时候 Model 可以不变，当 Model 变化的时候 View 也可以不变。 2、可重⽤性： 你可以把⼀些视图逻辑放在⼀个 ViewModel ⾥⾯，让很多 view 重⽤这段视图逻辑。 3、独⽴开发： 开发⼈员可以专注于业务逻辑和数据的开发（ViewModel），设计⼈员可以专注于⻚⾯设计。 4、可测试： 界⾯素来是⽐较难于测试的，⽽现在测试可以针对 ViewModel 来写。

MVVM 和 MVC 的区别： mvc 和 mvvm 其实区别并不⼤。都是⼀种设计思想。 主要区别 mvc 中 Controller 演变成 mvvm 中的 viewModel， mvvm 通过数据来显示视图层⽽不是节点操作。 mvvm 主要解决了: mvc 中⼤量的 DOM 操作使⻚⾯渲染性能降低，加载速度变慢，影响⽤户体验。

# **13. vue 双向数据绑定原理?(必问) 过⼀下**

vue.js 则是采⽤ 数据劫持 结合 发布者-订阅者 模式的⽅式， 通过 Object.defineProperty() 来劫持各个属性的 setter ， getter ， 在数据变动时发布消息给订阅者，触发相应的监听回调。 这个时候就可以实现数据的双向绑定 Object.defineProperty 是对象的一个方法，当添加或者修改属性的时候会触发，第一个参数是目标对象，第二个参数是要修改的属性，第三个参数是配置项，可以是 value 和 writeable 等选项，也可以是 get 和 set 函数 get 是获取值的时候触发 需要 return 一个值就是这个属性的值 set 是修改值的时候触发 有一个默认参数就是修改后的值

Object.defineProperty 的缺点 是 对于新添加的属性检测不到 解决的方法 this.$set

# **14. vue 如何封装可复⽤的组件?以及请举例说明你封装过的组件?(50%) 讲解**

根据项目需求合理规划子组件（是封装一个页面还是一个页面多个子组件 具体情况具体分析）

定义 components 文件夹 定义子组件

引入 注册 作为标签使用

合理使用插槽和组件通信 保证 子组件的封闭性和开发性（有些东西是固定的，有些东西是动态的就需要插槽或者组件通信设置

举例： 封装头部子组件 标题部分用插槽 封装列表子组件 商品数据通过父传子传递 导航子组件 数据通过父传子传 点击事件通过子传父实现

# **15. vue 中 key 的作⽤是什么**

在用 for 循环的时候用 key 值，为了避免元素重复渲染，在设置 key 值的时候一般设置为 id 等唯一的值 比如说循环 1-5 在 3 的后面插入一个 a 如果没有 key 值 那么从 a 开始 a 4 5 都要重新渲染 如果有 key 值 只会渲染 a 所以有 key 值能提高循环效率

# **16.说⼀下 vue 和 jquery 的区别?(50%)**

⾸先呢 jquery 他是⽤ js 封装的⼀个类库,主要是为了⽅便操作 dom 元素,⽽ vue 他是⼀个框架,并且呢,他会从真实 dom 构建出⼀个 虚拟的 dom 树,通过 di!算法渲染只发⽣改变的 dom 元素,其他的相同的 dom 元素不⽤在重新渲染. ⽽使⽤ jquery 去改变 dom 元素 的时候,即使有相同的 dom 元素也会重新渲染,以上就是我对 vue 和 jquery 区别的理解.

# **17. vue 中 data 发⽣变化,视图不更新如何解决?(必问) 过⼀下**

给对象或者数组新添加的属性不是响应式的，是因为 vue 的双向数据绑定的原理是 Object.defineProperty，由于这个方法的限制无法检测到属性的新增和删除，不会响应到视图上 解决方法就是用 Vue.set/this.$set 在组件里就用this.$set 在 js 文件里用 Vue.set 第一个参数 目标对象，第二个参数是要添加的属性，第三个是初始值

# **18. 为什么 vue 中 data 必须是⼀个函数?(必问) 过⼀下**

如果 data 是⼀个函数的话，这样每复⽤⼀次组件，就会返回⼀份新的 data，类似于给每个组件实例创建⼀个私有的数据空 间，让各个组件实例维护各⾃的数据。⽽单纯的写成对象形式，就使得所有组件实例共⽤了⼀份 data，就会造成⼀个变了全 都会变的结果。 所以说 vue 组件的 data 必须是函数。这都是因为 js 的特性带来的，跟 vue 本身设计⽆关。

# **19.说⼀下什么是 vue 过滤器? 有⼏种?项⽬中如何使⽤,请举例说明?(60%)**

所谓的 vue 过滤器就是将数据进⾏⼆次处理,得到我们想要的结果数据 vue 的过滤器分为两种,第⼀种是全局过滤器,通过 vue.filet 来进⾏定义,第⼆种是局部过滤器,需要定义在组件内部 项⽬中我们通过过滤器将后台返回的状态 0 和 1 转化为⽀付或者未⽀付

# **20. 怎样理解 Vue 的单向数据流?**

数据总是从⽗组件传到⼦组件，⼦组件没有权利修改⽗组件传过来的数据，只能请求⽗组件对原始数据进⾏修改。这样会防⽌从 ⼦组件意外改变⽗级组件的状态，从⽽导致你的应⽤的数据流向难以理解。 注意：在⼦组件直接⽤ v-model 绑定⽗组件传过来的 prop 这样是不规范的写法 开发环境会报警告 如果实在要改变⽗组件的 prop 值 可以再 data ⾥⾯定义⼀个变量 并⽤ prop 的值初始化它 之后⽤$emit 通知⽗组件去修改

还有一个方法修饰符 sync 可以在子组件里用 emit 修改父组件传来的值 在父组件传值的时候加上这个修饰符就行了 这个修饰符 就是:num @update:num 的简写

# **21.虚拟 dom**

操作真实 dom 的代价是很昂贵的，所以在 vue 中采取虚拟 dom 提高渲染效率，虚拟 dom 就是根据真实的 dom 所生成的一个 js 对象，这个对象里包含了这个 dom 节点的标签属性子元素等信息

优点: 保证性能下限： ⽐起粗暴的 DOM 操作性能要好很多 ⽆需⼿动操作 DOM： 我们不再需要⼿动去操作 DOM，我们可以以可预期的⽅式更新视图，极⼤提⾼我们的开发效率； 跨平台： 虚拟 DOM 本质上是 JavaScript 对象,所以虚拟 DOM 可以进⾏更⽅便地跨平台操作 缺点: ⽆法进⾏极致优化： 虽然虚拟 DOM + 合理的优化，⾜以应对绝⼤部分应⽤的性能需求，但在⼀些性能要求极⾼的应⽤中虚拟 DOM ⽆法进⾏针对性的极致优化。 ⾸次渲染⼤量 DOM 时，由于多了⼀层虚拟 DOM 的计算，会⽐ innerHTML 插⼊慢

（ 不用背 参考虚拟 dom 具体长啥样

# **22. Vue 的 diff 算法原理是什么？**

diff 算法就是把新生成的虚拟 dom 和旧的虚拟 dom 进行对比的过程就是 diff 算法 Vue 的 diff 算法是平级⽐较，不考虑跨级⽐较的情况。内部采⽤深度递归的⽅式+双指针⽅式⽐较 先⽐较两个节点是不是相同节点 相同节点⽐较属性，复⽤⽼节点 先⽐较⼉⼦节点，考虑⽼节点和新节点⼉⼦的情况 优化⽐较：头头、尾尾、头尾、尾头 ⽐对查找，进⾏复⽤

# **23.说⼀下你对 keep-alive 的理解?以及在项⽬中如何使⽤?**

keep-alive 是一个内置组件，作用是缓存不活动的组件不被销毁，下次访问这个组件的时候不会重新加载而是从缓存里拿。 比如说刚填好的表单需要缓存 还有保持滚动条的距离 keep-alive 的黑白名单：可以用 include 和 exclude 规定缓存谁不缓存谁 也可以 使用路由的元信息定义

跟他相关的两个钩子函数的 actived 和 deactived 举例：保持滚动条的距离： 给 router-view 外层套一个 keep-alive 在想要保持滚动条的页面 的 beforeRouteLeave 的时候拿到当前滚动条饿的距离 在 actived 里面 赋值给页面的滚动条 拿滚动条或者赋值滚动条都是 document.documentElement.scrollTop

# **24.插槽**

插槽分为 匿名插槽 具名插槽 和 作用域插槽 匿名插槽指的是不起名字的插槽 具名插槽是起了具体名字的插槽 v-slot 也可以简写为#起名字 用 slot 的时候加上 name 属性区分名字 作用域插槽是把子组件里的数据拿到父组件的插槽位置那里使用 在子组件通过自定义属性传 在父组件插槽那里通过 v-slot=“”起个名字接收

匿名插槽和具名都用的很多 比如在封装头部子组件的时候 中间的标题每个页面都不一样 就可以用插槽表示 在封装弹框子组件的时候 弹框里的内容也可以用插槽表示

# **25.axios 的封装**

先建一个 utils 文件夹 里面放 request.js 文件

在 request.js 文件用 create 方法创建 axios 实例，并且设置基本路径和超时时间

设置请求拦截和响应拦截

在请求拦截的成功的回调函数里 设置 token 以及 全局的 loading

在响应拦截的成功回调函数里 清除 loading 以及根据 返回的编码做一个处理

在响应拦截的失败的回调里 做错误编码处理

再创建一个 api.js 统一管理我们的接口

# **26. 拦截器**

⾸先呢,axios 拦截器是 axios 给我们提供的两个⽅法,通过这两个⽅法我们可以对请求发送之前以及响应之后进⾏逻辑的再次处 理(拦截). 这两个拦截器不需要⼿动触发,只要发送 http 请求的时候就会⾃动触发. 我在项⽬中经常通过拦截器发送 token, 对 token 进⾏过期处理,以及进⾏其他的⼀些操作

# **27.vue 本地的跨域 代理跨域 proxy**

本地跨域是通过在 vue.config.js ⽂件⾥⾯的 devServer 属性⾥⾯的 proxy 属性⾥⾯配置,⼀共配置三个属性,分别是代理名称 代理 地址 开启跨域 重写路径

# **28. Vue.use 是⼲什么的？**

vue.use 是⽤来使⽤插件的。use 可以注册组件库，use 方法会调用 install 方法，install 方法里有一个参数 Vue，所以不需要再引入 Vue，可以理解为 Vue.component 是注册一个组件，但是用 use 方法可以实现一次注册多个组件 除了可以注册组件之外还能注册全局的方法。

# **29.组件写 name 有啥好处？**

增加 name 属性，会在 components 属性中增加组件本身，实现组件的递归调⽤。

可以表示组件的具体名称，⽅便调试和查找对应的组件 比如 keep-alive 缓存组件设置黑白名单的时候 就可以通过 name 设置

# **30.mixin**

mixin 混入 把公共的选项（生命周期函数，计算属性，watch 等）提到一个 js 文件里，哪个组件用就在哪个组件引入 混入有全局混入和局部混入 局部混入是只在这个组件里能用 全局混入是这个项目的任何组件都能使用 如果混入和组件都有这个选项的时候混入里的先触发

# **31.路由跳转方式 路由配置 嵌套和动态区别**

路由跳转： 1.1router-link 标签跳转 1.2.编程式导航 this.$router.push() 1.3.this.$router.replace（{path：‘/’ } ） （replace 跟 push 类似，只不过跳转之后不存历史记录）

传参是传参 跳转是跳转 传参就是我们之前背的 query params 动态路由传参 在路由跳转的时候可以携带参数 可以在这里说我们之前背的路由的三种传参 query 可以通过 name 和 path 都能跳转 接收通过 query 接收 就相当于是 get 传输参数 因为参数会被拼接到 url 上刷新数据不丢失 params 只能通过 name 跳转 接收通过 params 接收 相当于是 post 传参 刷新数据丢失 解决方法：采取动态路由传参 还有存本地存储里用的时候拿 动态路由 通过 params 接收 还可以开启 props 传参 通过 props 接收 注意路由后面要拼上:xx 参数 刷新数据不丢失

路由的配置： 在 router/index.js 中配置路由规则 路由规则中的 path 属性代表路径，name 属性是名字，component 属性代表在这个路径下加载什么组件，还可以用 children 属性配置子路由。还可以用 meta 定义路由的元信息。

动态路由： 动态路由是指路由器能够自动的建立自己的路由表，并且能够根据实际情况的变化实时地进行调整。用：开头，：后面跟的值是不确定的。这个值是我们要传递的参数

嵌套路由： vue 项目中，界面通常由多个嵌套的组件构成，用 children 实现嵌套路由

# **32.自定义指令和自定义过滤器的使用场景和作用**

自定义指令： 使用位置：写在在标签里面 以 v-开头

自定义指令的钩子函数 一个指令定义对象可以提供如下几个钩子函数 (均为可选)：

bind：指令第一次绑定到元素时调用。

inserted：被绑定元素插入父节点时调用

update：所在组件的 虚拟 dom 更新时调用，

componentUpdated：指令所在组件的 VNode 及其子 VNode 全部更新后调用。

unbind：只调用一次，指令与元素解绑时调用。

钩子函数的参数 即 el、binding el 指令所绑定的元素 binding：一个对象，包含这个自定义指令的一些信息

在很多时候，你可能想在 bind 和 update 时触发相同行为，而不关心其它的钩子就可以简写为一个函数

过滤器： 使用位置:双花括号插值 和 v-bind 表达式 使用场景 ：处理数据格式的 比如说 我们可以用过滤器过滤时间格式

# **33.多环境变量**

我们在 vue 项目开发中,项目在运行时会根据启动的指令来运行不同的环境,在不同的环境中,我们配置对应所需的变量来满足我们的开发需求,称为多环境变量。 在项目根目录定义两个文件 .env.dev 里面写上 NODE\_ENV="development" .env.prod 里面写上 NODE\_ENV="production" 在封装 axios 的文件里放一个 baseURL.js 里面判断环境 不同的环境 定义不同的 baseURl 再把这个文件引入 封装的 axios 文件中使用 （代码详见 10.21-day5-demo 文件夹下）

# **34.路由信息对象和实例对象**

this.$route 路由信息对象 包含的有参数和路径等信息 路由传参接收的时候 this.$route.query.params this.$router 路由实例对象 包含了 push 等方法

# **35.vue3 和 vue2 生命周期函数的对比**

beforeCreate -> setup() 开始创建组件之前，创建的是 data 和 method created -> setup() beforeMount -> onBeforeMount 组件挂载到节点上之前执行的函数。 mounted -> onMounted 组件挂载完成后执行的函数 beforeUpdate -> onBeforeUpdate 组件更新之前执行的函数。 updated -> onUpdated 组件更新完成之后执行的函数。 beforeDestroy -> onBeforeUnmount 组件挂载到节点上之前执行的函数。 destroyed -> onUnmounted 组件卸载之前执行的函数。 activated -> onActivated 组件卸载完成后执行的函数 deactivated -> onDeactivated 组件停用