### Vue的生命周期有哪些及每个生命周期做了什么

<https://www.bilibili.com/video/BV1JK411G7wh/?spm_id_from=333.337.search-card.all.click&vd_source=ceab44fb5c1365a19cb488ab650bab03>

生命周期函数是在某个时刻会自动执行的函数。

Vue2和Vue3钩子变化不大，beforeCreate 、created  两个钩子被setup()钩子来替代。

* Vue2的生命周期

1. beforeCreate是new Vue()之后触发的第一个钩子，在当前阶段data、methods、computed以及watch上的数据和方法都不能被访问。
2. created在实例创建完成后发生，当前阶段已经完成了数据观测，也就是可以使用数据，更改数据，在这里更改数据不会触发updated函数。可以做一些初始数据的获取，在当前阶段无法与Dom进行交互，如果非要想，可以通过vm.$nextTick来访问Dom。
3. beforeMount发生在挂载之前，在这之前template模板已导入渲染函数编译。而**当前阶段虚拟Dom已经创建完成**，即将开始渲染。在此时也可以对数据进行更改，不会触发updated。【模板渲染到页面之前】
4. mounted在挂载完成后发生，在当前阶段，**真实的Dom挂载完毕**，数据完成双向绑定，可以访问到Dom节点，使用$refs属性对Dom进行操作。【**模板渲染到页面完毕**】
5. beforeUpdate发生在更新之前，也就是响应式数据发生更新，虚拟dom重新渲染之前被触发，你可以在当前阶段进行更改数据，不会造成重渲染。
6. updated发生在更新完成之后，当前阶段组件Dom已完成更新。要注意的是避免在此期间更改数据，因为这可能会导致无限循环的更新。
7. beforeDestroy发生在实例销毁之前，在当前阶段实例完全可以被使用，我们可以在这时进行善后收尾工作，比如清除计时器。
8. destroyed发生在实例销毁之后，这个时候只剩下了dom空壳。组件已被拆解，数据绑定被卸除，监听被移出，子实例也统统被销毁。

有两个路由组件特有的周期函数，actived和deactivated。

### 一旦进入到页面（组件）会执行哪些生命周期

beforeCreate、created、beforeMount、mounted，创建前后&挂载前后

### 父组件引入子组件，生命周期执行顺序

先把父组件的一些数据处理了，因为有可能要给子组件。然后执行子组件的数据处理+子DOM，最后执行父组件的DOM

**顺序：**

父beforeCreate、created、beforeMount

子beforeCreate、created、beforeMount、mounted

父mounted

### 发送请求在created还是mounted

JS先执行同步后执行异步，生命周期函数是同步代码，我们的axios请求是一个异步的

因此不管axios放在（beforeCreate、created、beforeMount、mounted）哪里，都是最后执行的

### 在created中如何获取DOM

1. 只要写异步代码，获取DOM在异步代码就可以了。比如在created中发送axios请求或者setTimeout、Promise
2. 使用vue系统内置的nextTick

### 怎么获取DOM

Vue2中 ① this.$el

② 组件里添加ref <div ref=”haha”></div>，然后JS中 this.$refs.haha访问到

### 为什么发送请求不在beforeCreate里？beforeCreate和Created区别

如果在beforeCreate中使用 axios.get或者 async await axios都是可以的，但是不能在methods里定义函数然后调用，因为beforecreate中无法调用methods里的方法

Beforecreate中没有data; create有data

created可以拿到methods方法，前者不行

### Vue响应式原理是什么？ vue3响应式有何不同

<https://www.bilibili.com/video/BV1ve411k714/?spm_id_from=333.337.search-card.all.click>

**模板：vue2是怎么做的，vue2这种做法有什么问题，vue3又是怎么做的且怎么解决这个问题的。**

在vue2中使用了object.defineProperty来做的，在页面刚开始加载时，vue会遍历data中的所有属性，然后使用object.defineProperty把这些属性全部转为getter和setter。当用户访问或者设置某个属性时，会触发对应的getter或setter方法，然后会通知每个组件实例对应的一个watch方法，最后实现视图的更新。

但是object.defineProperty的缺点有① 对于复杂对象，需要深度监听，一次性监听到底，他的计算量很大性能不好，②对于对象的新增删除属性，这些操作无法监听，所以他需要使用vue.$set和vue.$detele作为辅助，③需要重写数组原生方法，来实现数组的监听。

所以在vue3中使用了proxy来代替defineProperty。他的优势在：①直接监听整个对象，而不需要遍历监听属性，性能有所提升②proxy可以直接监听数组的变化，而不需要重写数组原生的方法，便利性增加。③proxy有多达13种拦截方法，功能更加强大④作为一个新标准，会受到浏览器厂商重点持续的性能优化