《前端高频面试100题系列》

一、['1', '2', '3'].map(parseInt) 输出什么?

1、parseInt(str, radix)

- 解析一个字符串，并返回**十进制**整数
- 第一个参数 str，即要解析的字符串
- 第二个参数radix，基数（进制），范围 2-36

2、未传递 radix 参数

- 当 str 以 '0x' 开头，则按照 16 进制处理
- 当 str 以 '0' 开头，则按照 8 进制处理（但 ES5 取消了！！！）
- 其他情况按 10 进制处理

3、分析代码

```
['1',  '2', '3'].map(parseInt)
// 相当于
['1', '2', '3'].map(function(item, index) {
  return parseInt(item, index)
  // parseInt('1', 0) // 注意，这里是按照 radix 参数不存在处理的。
  // parseInt('2', 1)
  // parseInt('3', 2)
})
// 因此结果为：[1, NaN, NaN]
```

二、说出下面代码的执行结果

```
var a = 1
{
  function a() {}
  a = 2
  function a() {}
  a = 3
  function a() {}
  a = 4
  console.log('内部a:', a)
}

console.log('外部a:', a)
```

这个问题可以转化为：书写在**块语句**中的代码是如何预解析的?

我们都知道 JS 中，在 ES6 之前作用域分为 *全局作用域* 和 *函数作用域* ，函数作用域也是 *块作用域* 的一种，虽然这种情况我们平时很少用到，官方API文档也没有说明,但是在实际的过程当中能够发现以下规律：

> 这里所讲的**块作用域**的情况是指 `{ }` 单独应用的时候。

1、当在块语句内引用的时候

- 将函数写在块语句中，命名函数只会预解析，不会预赋值。只有在执行块语句的时候，赋值函数。
- 如果块语句中出现变量和函数名相同的情况时，执行块语句，最后打印的是正常顺序赋值的结果。
- 使用ES6语法 let 和 const 的时候遵从 ES6 对应规则的变量访问机制，定义机制，赋值机制。

2、当在块语句外引用的时候

- 得到的变量，是最后一个**重名函数上面的赋值变量**的结果，如果上面没有重名的赋值变量，那么得到的就是这个函数。
- 在块语句中，不管有几个同名函数，都会被最后一个覆盖掉。

> **总结一句话：**这个只需要记住一点，变量与函数同名时，块作用域中最后一个同名函数上面的变量赋值将会被挤出块外。

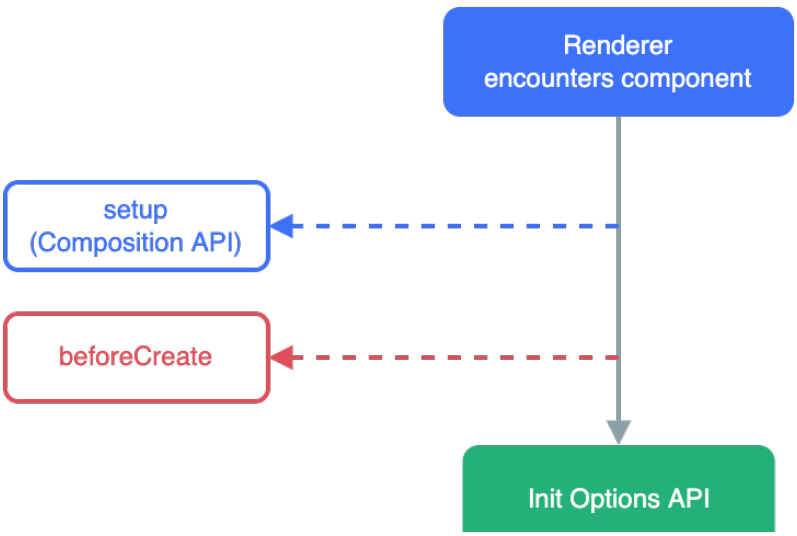
3、分析代码

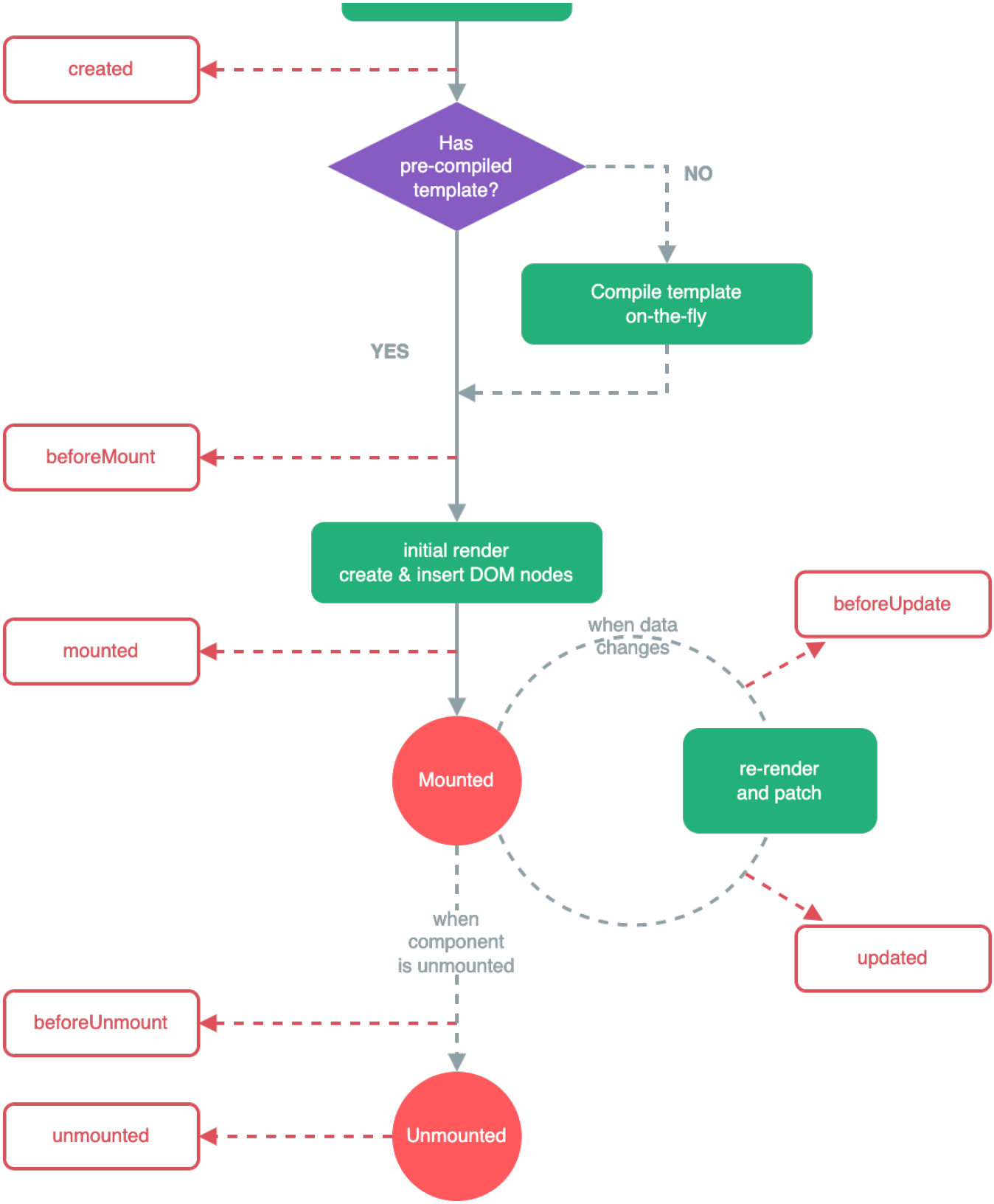
根据以上规律，上述代码的执行结果为：

```
内部a: 4
外部a: 3
```

三、Vue 每个声明周期都做了什么?

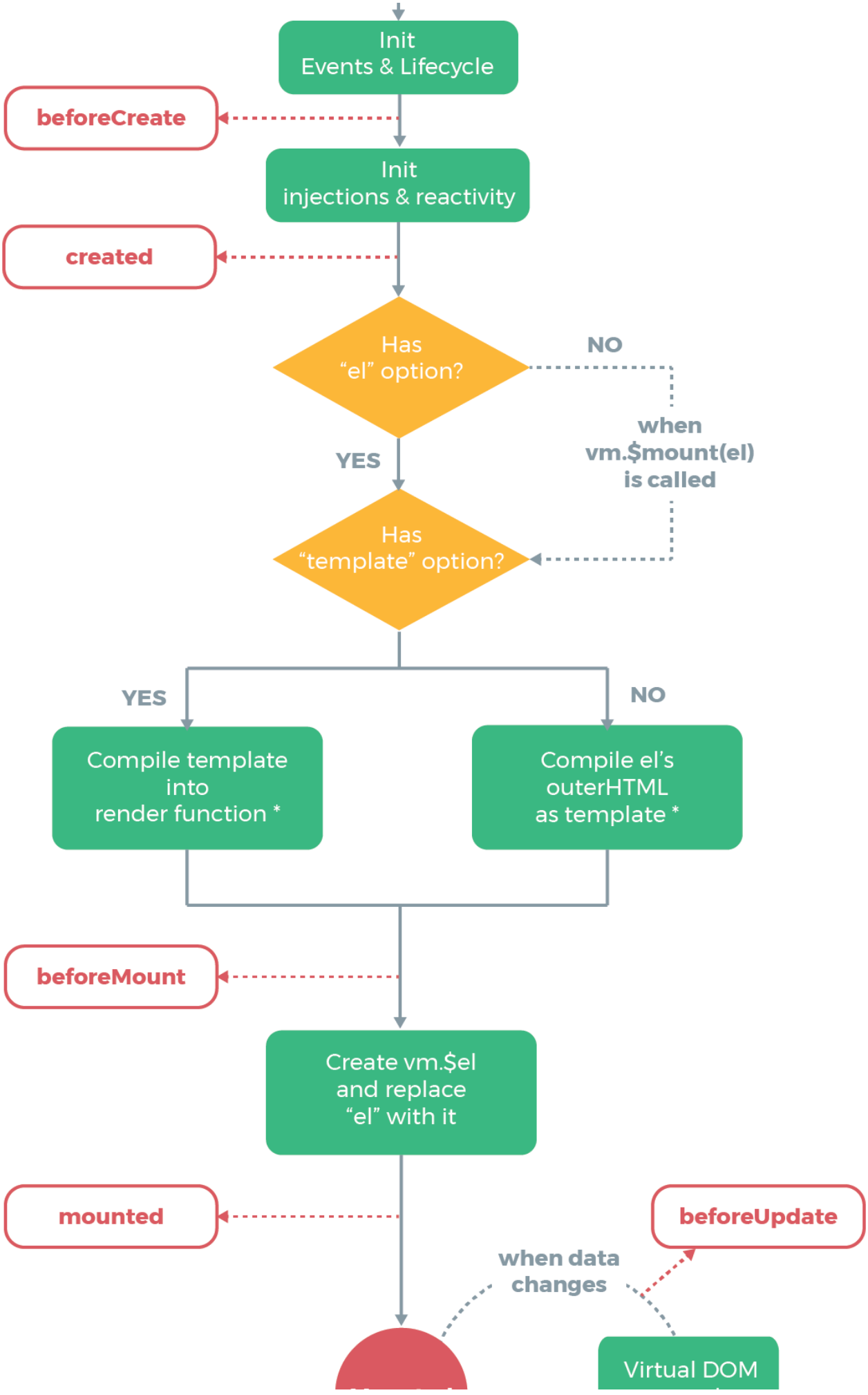
1、Vue3 生命周期

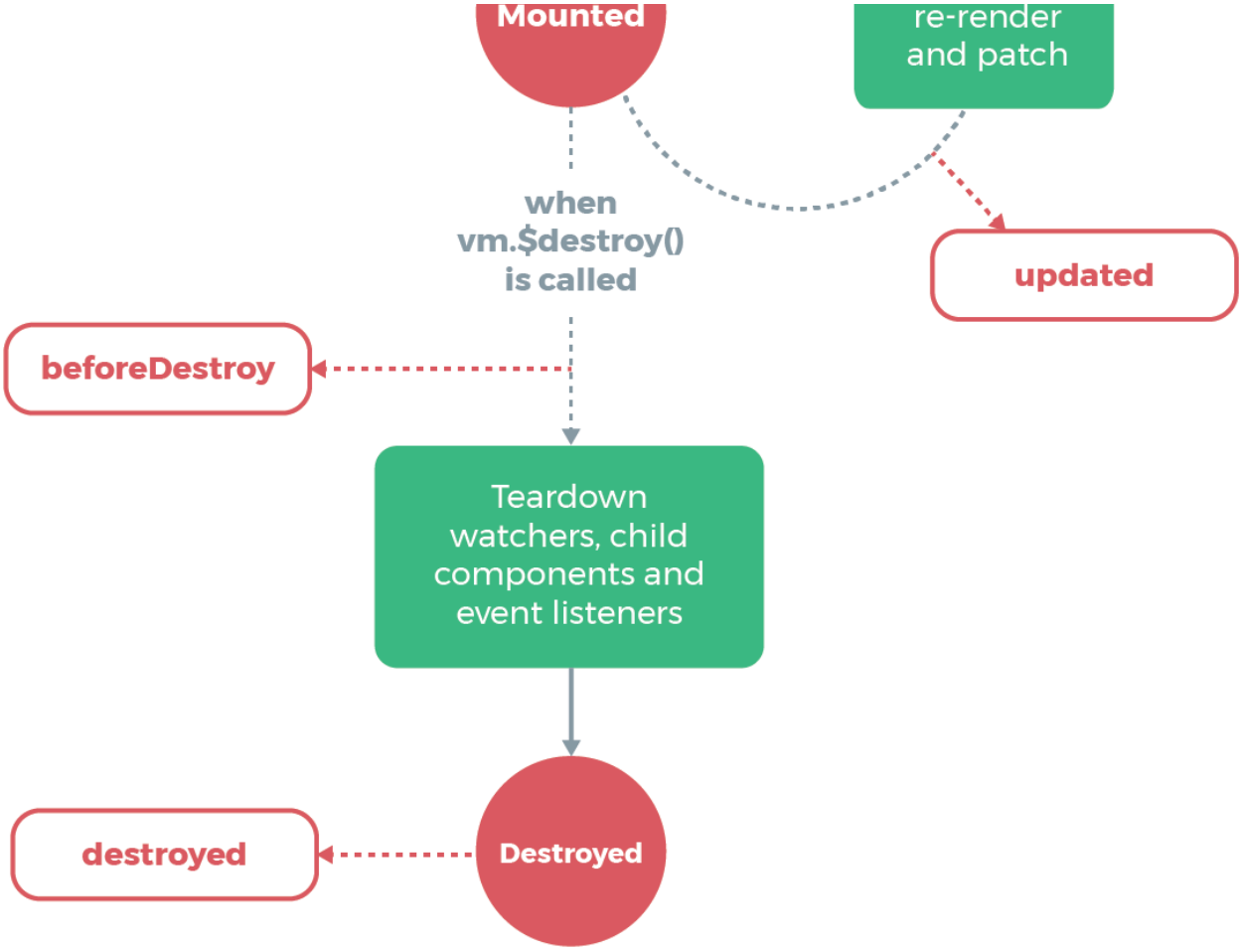



created
Has pre-compiled template?
NO
Compile template on-the-fly
YES
beforeMount
initial render create & insert DOM nodes
mounted
when data changes
beforeUpdate
Mounted
re-render and patch
updated
when component is unmounted
beforeUnmount
unmounted
Unmounted

2、Vue2 生命周期


new Vue()


Init
Events & Lifecycle
beforeCreate
Init
injections & reactivity
created
Has
“el” option?
NO
when
vm.$mount(el)
is called
YES
Has
“template” option?
YES
NO
Compile template
into
render function *
Compile el’s
outerHTML
as template *
beforeMount
Create vm.$el
and replace
“el” with it
mounted
beforeUpdate
when data
changes
Virtual DOM



* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

3、普通钩子说明

（1）beforeCreate

- 创建空白的 Vue 实例
- data、methods 尚未被初始化，不可使用

（2）created

- Vue实例初始化完成，完成响应式绑定
- data、methods 都已经初始化完成，可调用
- 尚未开始渲染模板

（3）beforeMount

- 编译模板，调用 render 生成 vdom
- 还没有开始渲染 DOM

（4）mounted

- 完成 DOM 渲染
- 组件创建完成
- 开始由“创建阶段”进入“运行阶段”

（5）beforeUpdate

- data 发生了变化之后
- 准备更新DOM（尚未更新 DOM）

（6）updated

- data 发生变化，且 DOM 更新完成
- （不要在 updated 中修改 data，可能会导致死循环）

（7）beforeUnmount(beforeDestroy)

- 组件进入销毁阶段（尚未销毁，可正常使用）
- 可移除、解绑一些全局事件、自定义事件

（8）unmounted(destroyed)

- 组件被销毁了
- 所有的子组件也都被销毁了

4、keep-alive 组件钩子说明

- **onActivated(activated)** 缓存组件被激活
- **onDeactivated(deactivated)** 缓存组件被隐藏

四、Vue 什么时候操作 DOM 比较合适?

- 在 mounted 和 updated 都不能保证子组件全部挂载完成
- 使用 $nextTick 渲染 DOM

```
mounted() {
  this.$nextTick(function(){
    // 仅在整个视图被渲染之后才会运行的代码
  })
}
// 或
async mounted() {
  await this.$nextTick()
  // 仅在整个视图被渲染之后才会运行的代码
}
```

五、Ajax 应该在哪个生命周期?

- 有两个选择：created 和 mounted
- 推荐 mounted

六、Vue3 Composition API 生命周期有何区别?

- 用 setup 代替了 beforeCreate 和 created
- 使用 Hooks 函数的形式，如 mounted 改为 onMounted()