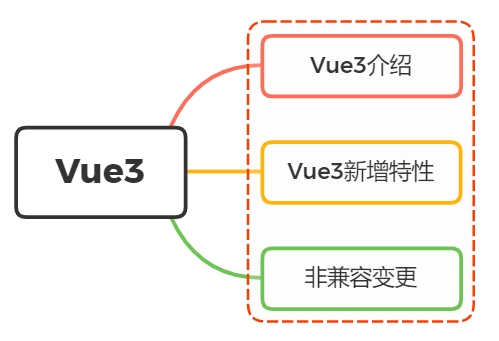
# 面试官：vue3有了解过吗？能说说跟vue2的区别吗？



## 一、Vue3介绍

关于vue3的重构背景，尤大是这样说的：

「Vue 新版本的理念成型于 2018 年末，当时 Vue 2 的代码库已经有两岁半了。比起通用软件的生命周期来这好像也没那么久，但在这段时期，前端世界已经今昔非比了

在我们更新（和重写）Vue 的主要版本时，主要考虑两点因素：首先是新的 JavaScript 语言特性在主流浏览器中的受支持水平；其次是当前代码库中随时间推移而逐渐暴露出来的一些设计和架构问题」

简要就是： - 利用新的语言特性(es6) - 解决架构问题

## 哪些变化



从上图中，我们可以概览Vue3的新特性，如下：

* 速度更快
* 体积减少
* 更易维护
* 更接近原生
* 更易使用

### 速度更快

vue3相比vue2

* 重写了虚拟Dom实现
* 编译模板的优化
* 更高效的组件初始化
* undate性能提高1.3~2倍
* SSR速度提高了2~3倍



### 体积更小

通过webpack的tree-shaking功能，可以将无用模块“剪辑”，仅打包需要的

能够tree-shaking，有两大好处：

* 对开发人员，能够对vue实现更多其他的功能，而不必担忧整体体积过大
* 对使用者，打包出来的包体积变小了

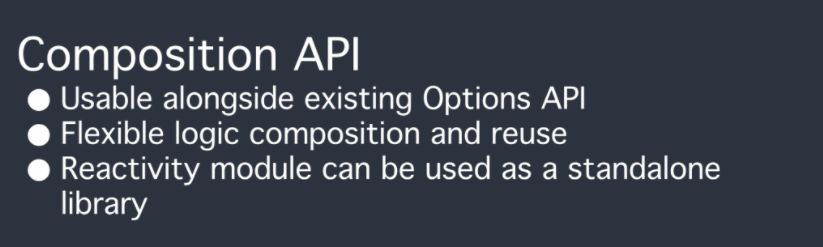
vue可以开发出更多其他的功能，而不必担忧vue打包出来的整体体积过多



### 更易维护

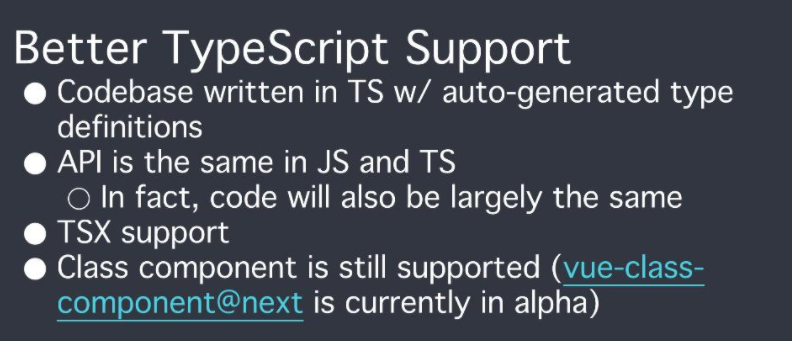
#### compositon Api

* 可与现有的Options API一起使用
* 灵活的逻辑组合与复用
* Vue3模块可以和其他框架搭配使用



#### 更好的Typescript支持

VUE3是基于typescipt编写的，可以享受到自动的类型定义提示

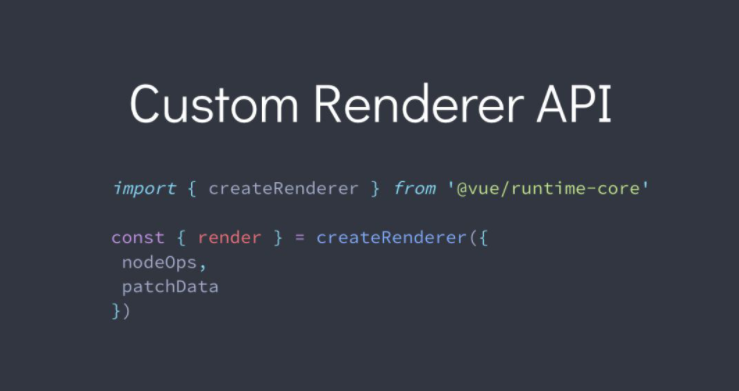


#### 编译器重写



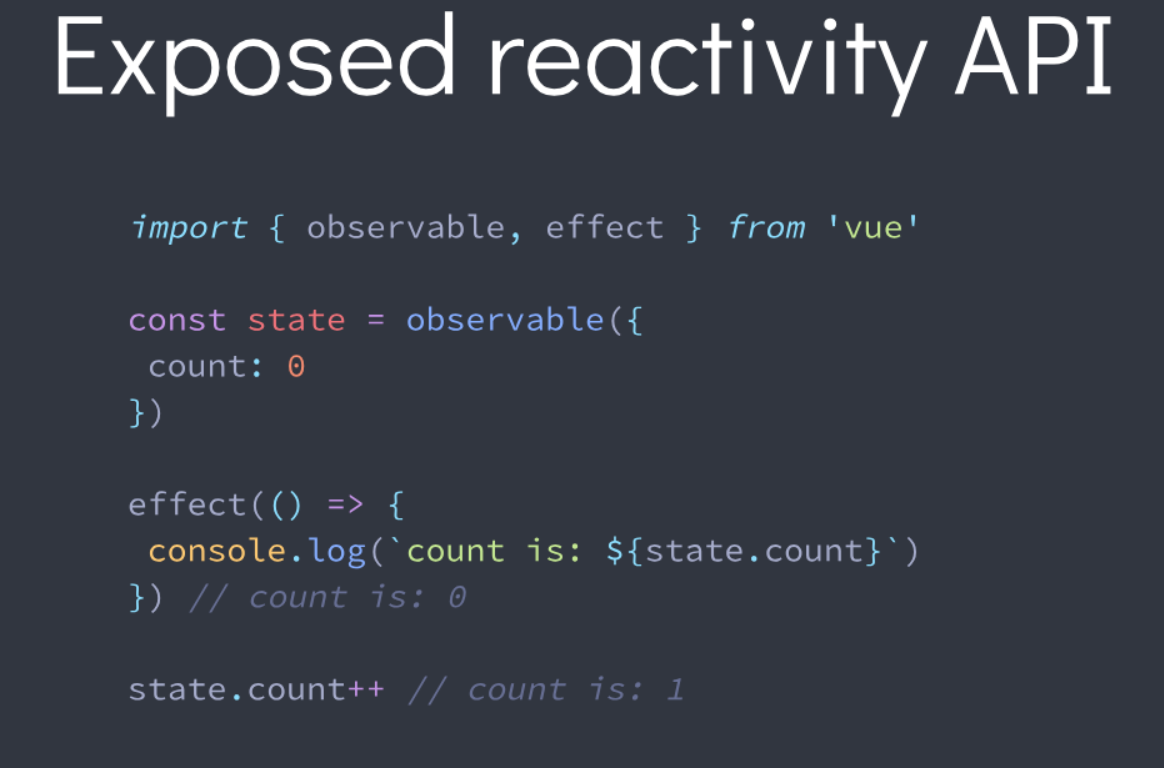
### 更接近原生

可以自定义渲染 API

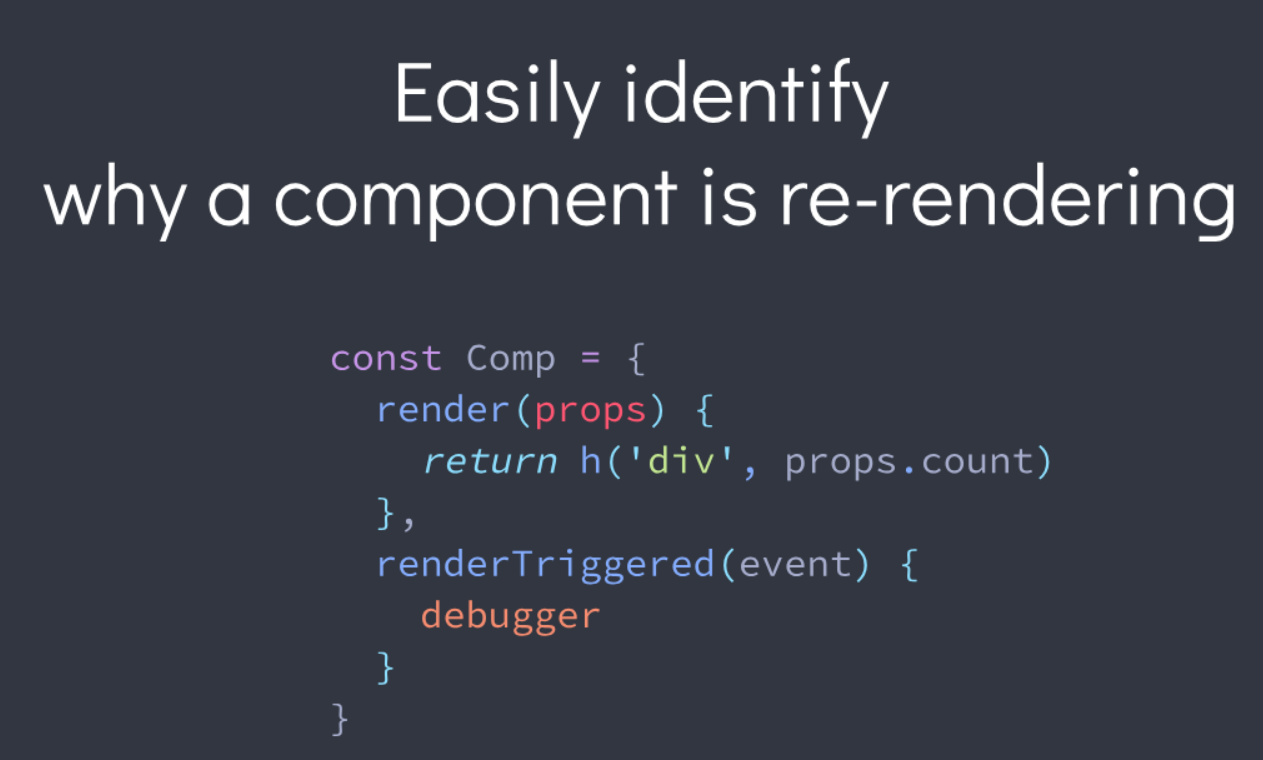


### 更易使用

响应式 Api 暴露出来



轻松识别组件重新渲染原因



## 二、Vue3新增特性

Vue 3 中需要关注的一些新功能包括：

* framents
* Teleport
* composition Api
* createRenderer

### framents

在 Vue3.x 中，组件现在支持有多个根节点

<!-- Layout.vue -->  
<template>  
 <header>...</header>  
 <main v-bind="$attrs">...</main>  
 <footer>...</footer>  
</template>

### Teleport

Teleport 是一种能够将我们的模板移动到 DOM 中 Vue app 之外的其他位置的技术，就有点像哆啦A梦的“任意门”

在vue2中，像 modals,toast 等这样的元素，如果我们嵌套在 Vue 的某个组件内部，那么处理嵌套组件的定位、z-index 和样式就会变得很困难

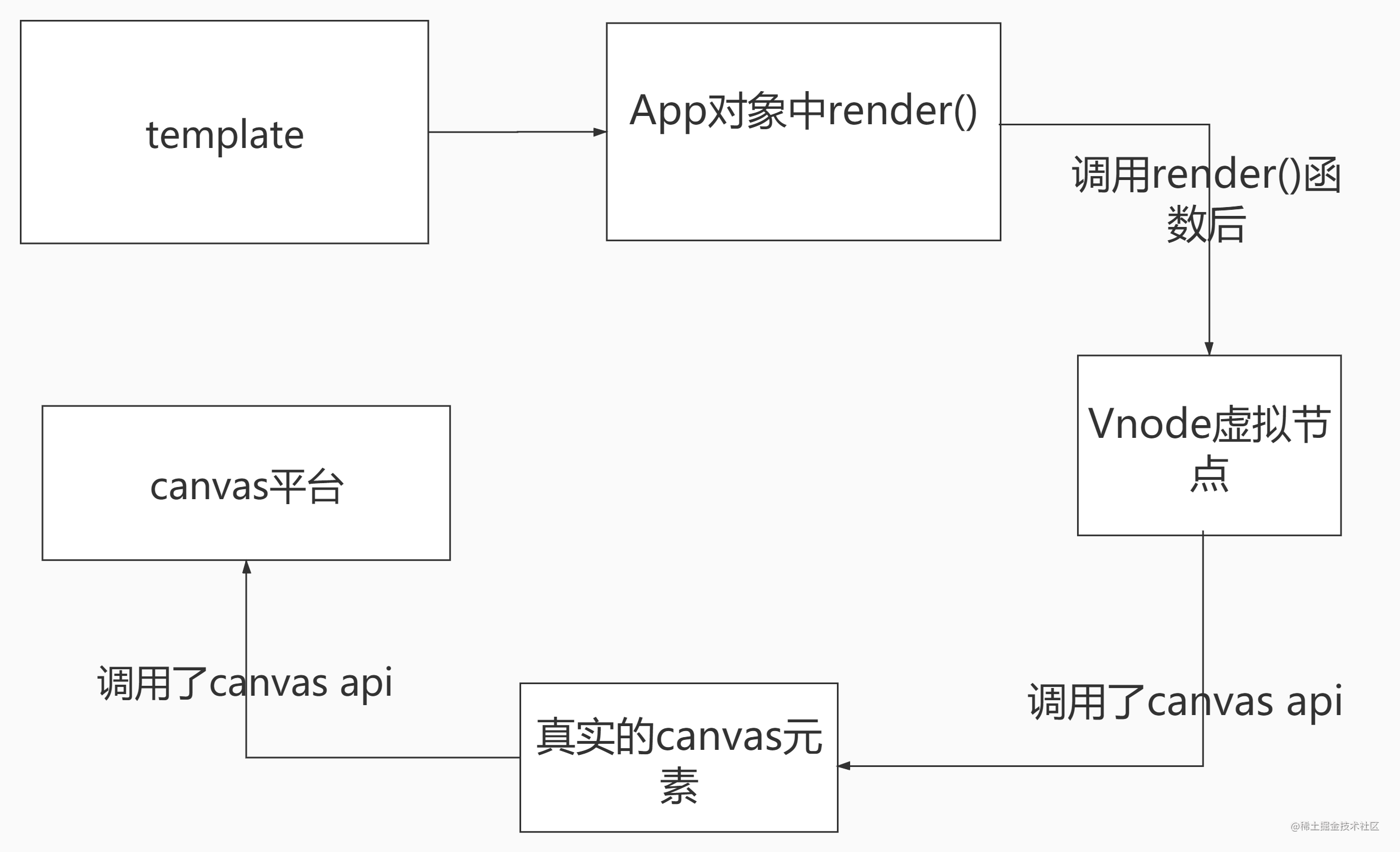
通过Teleport，我们可以在组件的逻辑位置写模板代码，然后在 Vue 应用范围之外渲染它

<button @click="showToast" class="btn">打开 toast</button>  
<!-- to 属性就是目标位置 -->  
<teleport to="#teleport-target">  
 <div v-if="visible" class="toast-wrap">  
 <div class="toast-msg">我是一个 Toast 文案</div>  
 </div>  
</teleport>

### createRenderer

通过createRenderer，我们能够构建自定义渲染器，我们能够将 vue 的开发模型扩展到其他平台

我们可以将其生成在canvas画布上



关于createRenderer，我们了解下基本使用，就不展开讲述了

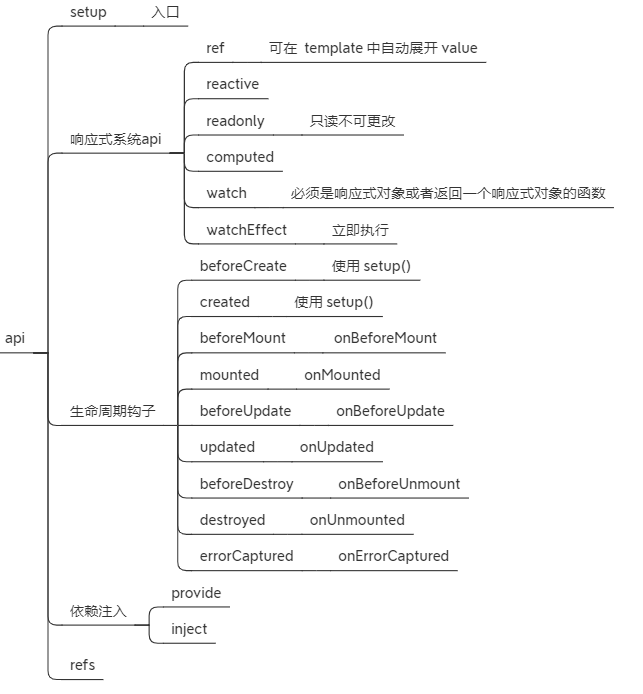
import { createRenderer } from '@vue/runtime-core'  
  
const { render, createApp } = createRenderer({  
 patchProp,  
 insert,  
 remove,  
 createElement,  
 // ...  
})  
  
export { render, createApp }  
  
export \* from '@vue/runtime-core'

### composition Api

composition Api，也就是组合式api，通过这种形式，我们能够更加容易维护我们的代码，将相同功能的变量进行一个集中式的管理



关于compositon api的使用，这里以下图展开



简单使用:

export default {  
 setup() {  
 const count = ref(0)  
 const double = computed(() => count.value \* 2)  
 function increment() {  
 count.value++  
 }  
 onMounted(() => console.log('component mounted!'))  
 return {  
 count,  
 double,  
 increment  
 }  
 }  
}

### 三、非兼容变更

### Global API

* 全局 Vue API 已更改为使用应用程序实例
* 全局和内部 API 已经被重构为可 tree-shakable

### 模板指令

* 组件上 v-model 用法已更改
* <template v-for>和 非 v-for节点上key用法已更改
* 在同一元素上使用的 v-if 和 v-for 优先级已更改
* v-bind="object" 现在排序敏感
* v-for 中的 ref 不再注册 ref 数组

### 组件

* 只能使用普通函数创建功能组件
* functional 属性在单文件组件 (SFC)
* 异步组件现在需要 defineAsyncComponent 方法来创建

### 渲染函数

* 渲染函数API改变
* $scopedSlots property 已删除，所有插槽都通过 $slots 作为函数暴露
* 自定义指令 API 已更改为与组件生命周期一致
* 一些转换 class 被重命名了：
* v-enter -> v-enter-from
* v-leave -> v-leave-from
* 组件 watch 选项和实例方法 $watch不再支持点分隔字符串路径，请改用计算函数作为参数
* 在 Vue 2.x 中，应用根容器的 outerHTML 将替换为根组件模板 (如果根组件没有模板/渲染选项，则最终编译为模板)。VUE3.x 现在使用应用程序容器的 innerHTML。

### 其他小改变

* destroyed 生命周期选项被重命名为 unmounted
* beforeDestroy 生命周期选项被重命名为 beforeUnmount
* [prop default工厂函数不再有权访问 this 是上下文
* 自定义指令 API 已更改为与组件生命周期一致
* data 应始终声明为函数
* 来自 mixin 的 data 选项现在可简单地合并
* attribute 强制策略已更改
* 一些过渡 class 被重命名
* 组建 watch 选项和实例方法 $watch不再支持以点分隔的字符串路径。请改用计算属性函数作为参数。
* <template> 没有特殊指令的标记 (v-if/else-if/else、v-for 或 v-slot) 现在被视为普通元素，并将生成原生的 <template> 元素，而不是渲染其内部内容。
* 在Vue 2.x 中，应用根容器的 outerHTML 将替换为根组件模板 (如果根组件没有模板/渲染选项，则最终编译为模板)。Vue 3.x 现在使用应用容器的 innerHTML，这意味着容器本身不再被视为模板的一部分。

### 移除 API

* keyCode 支持作为 v-on 的修饰符
* $on，$off和$once 实例方法
* 过滤filter
* 内联模板 attribute
* $destroy 实例方法。用户不应再手动管理单个Vue 组件的生命周期。

## 参考文献

* https://vue3js.cn/docs/zh/guide/migration/introduction.html#%E6%A8%A1%E6%9D%BF%E6%8C%87%E4%BB%A4
* https://composition-api.vuejs.org/zh/#api-%E4%BB%8B%E7%BB%8D