如何理解 Composition API 和 Options API

Vue.js 3 引入了 Composition API，这是一个与 Options API 并行的 API，允许用户以更灵活和可维护的方式组织和重用代码。

它们只是同一个底层系统所提供的两套不同的接口。实际上，选项式 API 是在组合式 API 的基础上实现的！

1. **Options API**

这是 Vue.js 2 中使用的传统 API。在 Options API 中，我们将代码划分为不同的选项（如 `data`、`methods`、`computed`、`watch` 等）。这种方式的优点是易于理解和上手，但在处理大型和复杂的组件时，可能会变得难以管理和维护。代码的重用性也相对较低，因为不同的选项之间不能直接访问彼此的数据和方法。

2. **Composition API**

这是 Vue.js 3 中引入的新 API。在 Composition API 中，我们使用函数（如 `ref`、`reactive`、`computed`、`watchEffect` 等）来组织和重用代码。这种方式的优点是更加灵活和可维护。我们可以将相关的逻辑放在同一个函数中，使代码更容易理解和重用。此外，由于函数可以返回值，我们可以轻松地在组件之间共享逻辑。

总的来说，Composition API 是对 Options API 的补充，而不是替代。对于小型项目和初学者来说，Options API 更容易理解和上手。而对于大型项目和有一定经验的开发者来说，Composition API 更灵活和可维护。

1. 为什么要用 Composition API

> - 更好的逻辑复用

组合式 API 最基本的优势是它使我们能够通过组合函数来实现更加简洁高效的逻辑复用。在选项式 API 中我们主要的逻辑复用机制是 mixins，而组合式 API 解决了 mixins 的所有缺陷。

组合式 API 提供的逻辑复用能力孵化了一些非常棒的社区项目，比如 VueUse，一个不断成长的工具型组合式函数集合。组合式 API 还为其他第三方状态管理库与 Vue 的响应式系统之间的集成提供了一套简洁清晰的机制，例如不可变数据、状态机与 RxJS。

> - 更灵活的代码组织

许多用户喜欢选项式 API 的原因是它在默认情况下就能够让人写出有组织的代码：大部分代码都自然地被放进了对应的选项里。然而，选项式 API 在单个组件的逻辑复杂到一定程度时，会面临一些无法忽视的限制。我们现在可以很轻松地将这一组代码移动到一个外部文件中，不再需要为了抽象而重新组织代码，大大降低了重构成本，这在长期维护的大型项目中非常关键。

> - 更好的类型推导

用组合式 API 重写的代码可以享受到完整的类型推导，不需要书写太多类型标注。大多数时候，用 TypeScript 书写的组合式 API 代码和用 JavaScript 写都差不太多！

> - 更小的生产包体积

搭配 `<script setup>` 使用组合式 API 比等价情况下的选项式 API 更高效，对代码压缩也更友好。这是由于 `<script setup>` 形式书写的组件模板被编译为了一个内联函数，和 `<script setup>` 中的代码位于同一作用域。不像选项式 API 需要依赖 this 上下文对象访问属性，被编译的模板可以直接访问 `<script setup>` 中定义

的变量，无需从实例中代理。这对代码压缩更友好，因为本地变量的名字可以被压缩，但对象的属性名则不能。