**1.Pinia 简介**

       Pinia是一个轻量级的状态管理库

       Pinia官网

<https://pinia.vuejs.org/zh>

       状态管理库是用于管理应用程序全局状态的工具

       以登录为例:

               使用Pinia创建一个userStore来集中管理用户的登录状态和过期时间

               当用户登录成功时:

                       设置userStore中用户的登录状态为已登录,并设置过期时间

               当用户退出登录时:

                       修改userStore中用户的登录状态为未登录,并删除过期时间

       Pinia 和 组件通信 的区别

       虽然Vue提供的父传子、子传父以及跨组件通信也可以用于状态共享,

       但在大型项目中,随着组件数量的增加,会导致以下问题:

       1.组件之间传递大量的props,会使项目变得非常繁琐和难以维护

       2.非父子组件间过度依赖provide/inject,使状态散落在各个组件之间

       Pinia 可以解决以下问题:

       1.全局状态管理

               所有组件都可以访问和修改状态,而不用在每个组件内部进行状态管理

       2.简化组件之间的通信

               避免在组件之间传递大量的props

       3.状态持久化

               可以将应用程序的状态保存到本地存储中,

               在应用程序重启后会保留状态,对于登录等场景非常有用

       总的来说,Pinia可以处理大型项目中复杂的状态管理需求,

       而父传子、子传父以及跨组件通信,可以解决一些简单的状态传递问题,

       更适合小型项目

       Pinia 和 localStorage 的区别

       localStorage

               LocalStorage只能存储字符串类型的数据

               LocalStorage有大小限制,通常为5MB左右

       Pinia

               Pinia可以存储任何类型的数据,包括对象、数组等

               Pinia没有大小限制,可以存储大量的数据

       总的来说,对于复杂的状态管理需求,使用Pinia是更好的选择,

       而对于简单的状态管理需求,使用localStorage是更简单的解决方案

**9.生命周期函数**

生命周期函数

       是组件实例从创建到销毁过程中不同时间点自动调用的函数

挂载阶段

       onBeforeMount

               在组件实例即将被挂载到DOM树之前调用

               此时模板还未编译或渲染到DOM,通常用于执行初始化操作,

               如:获取异步数据、设置初始属性值等

       onMounted

               在组件成功挂载到DOM并完成首次渲染后调用

               此时可以访问和操作DOM元素,

               并执行与页面交互相关的逻辑

更新阶段

       onBeforeUpdate (由于响应式数据变化)

               在组件更新之前即将重新渲染时调用

               可以根据新的参数判断是否需要进行特殊处理,

               甚至可以选择阻止此次更新过程

       onUpdated

               在组件完成更新并重新渲染后调用

               可以基于新的渲染结果处理更新后的数据

卸载阶段

       onBeforeUnmount

               在组件从DOM中销毁之前调用

               用于释放资源,如:清理计时器、解绑事件监听器等

       onUnmounted

               在组件已经从DOM中移除并销毁后调用

               确保组件所占用的所有资源都被正确释放

错误处理

       onErrorCaptured

               在捕获到组件中的错误时调用

               用于处理错误,如:记录错误日志等

注

组件挂载的过程

模板编译

       将组件的模板转换为JS代码

渲染

       在模板编译后生成的JS代码渲染到页面上,

       生成虚拟DOM

挂载

       在渲染完成后将虚拟DOM挂载到真实的DOM树上,

       使其在页面上显示出来

<script setup>

import { onMounted, onUpdated, ref } from 'vue'

//在组件成功挂载到DOM并完成首次渲染后调用

onMounted(() => {

console.log("onMounted")

})

//在组件更新之后调用

onUpdated(() => {

console.log("onUpdated:",user.value)

})

const user = ref(0)

console.log("user:",user.value)

</script>

<template>

<h3>App.vue</h3>

{{ user }}

<button @click="user++">添加用户</button>

</template>

<style scoped>

</style>