**一键换肤功能的实现总结**

**一、思路：**

1.用户通过点击主题色卡更改layout页面背景色样式

2.把用户选择的主题色卡对应的样式通过创建link标签动态的引到layout页面

3.一键换肤功能实现

**二、功能实现过程中的知识点：**

1.通过localStorage(本地存储)保存用户选择的主题样式，方便用户在下一次进入页面时，仍然是上一次的选择

 localStorage.setItem('color', 'default')

2.通过watchEffect监听color值得变化，创建link标签动态引入用户选择的主题对应的css文件，并将其添加到head中

watchEffect(async () => {

    color.value = themeColors.color

    let link = document.createElement('link');

    link.type = 'text/css';

    link.id = "theme";

    link.rel = 'stylesheet';

    link.href = url;

    document.getElementsByTagName("head")[0].appendChild(link);

})

1. 通过pinia来获取localStorage中存储的color值，因为pinia有数据双向绑定的功能，我们需要在用户选择主题色时，layout页面能同时拿到选择的颜色，这样页面不需要刷新就可以变色

import { defineStore } from "pinia";

import { reactive, computed } from 'vue';

import type { Colors } from "@/types/ThemeColors"

export const useThemeStore = defineStore('themeStore', () => {

  const themeColors = reactive<Colors>({

    id: 1,

    appearanceUrl: '',

    title: '',

    color: localStorage.getItem('color') as string

  })

  let color = computed(() => themeColors.color)

  const setThemeColor = (payLoad: Colors) => {

    Object.assign(themeColors, payLoad)

    localStorage.setItem('color', themeColors.color)

    localStorage.setItem('id', themeColors.id + '')

  }

  const getThemeColor = () => {

    return localStorage.getItem('color') as string

  }

  const getThemeId = () => {

    return localStorage.getItem('id') as string

  }

  return { themeColors, color, setThemeColor, getThemeColor, getThemeId }

})

4.判断当前生产或开发环境，来改变css文件路径

const url = import.meta.env.MODE == 'production' ? `dist/theme/theme-${color.value}.css` : `../src/theme/theme-${color.value}.css`

5.在用户第一次进入页面时，需要在localStorage里存一次默认样式，否则页面没有任何样式会变成白色，判断localStorage中是否存在值，如果没有，自动添加默认样式

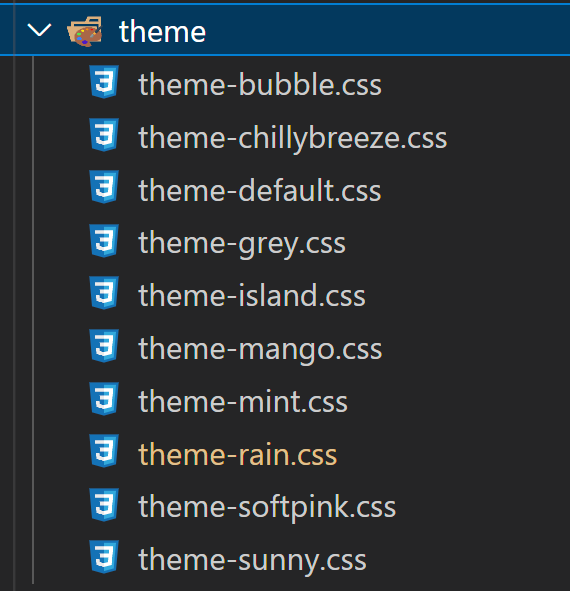
if (localStorage.getItem("color") == null && localStorage.getItem("id") == null) {

    localStorage.setItem('color', 'default')

    localStorage.setItem('id', '1')

}

1. 我们需要单独的为每个主题色创建一个css文件，他们之间互不影响



1. 在style中我们需要使用变量来改变颜色

 background: var(--gradient-bg-color);

1. **到此为止，一键换肤功能就实现了，来总结一下开发过程中踩的坑**

1.@import和require 在引入文件后，除非页面被关闭，否则不可被删除，所以采用link标签

2.更换element-plus组件中按钮颜色是，需要更改组件自带的颜色变量的值

3.跨页面传值时，一个页面需要在另一个页面的值发生改变时同时改变需要用到pinia来实现数据双向绑定