**CreateImage组件代码说明文档**

**一、整体概述**

该 Vue 组件实现了一个超级 AI 画布页面，允许用户输入心情文本，通过与后端 API 交互生成相应图片，并展示生成的图片。同时，支持对生成图片进行编辑操作，以及显示图片生成过程中的加载状态和提示信息。

**二、模板部分**

1. **页面结构**：
   * 整个组件的根元素是 <body>，并应用了 a 类样式。
   * <div class="container"> 作为主要容器，用于包裹页面内的主要内容，设置了最大宽度、自动居中、内边距以及背景颜色等样式，同时有阴影和圆角效果。
   * <h1> 标签用于显示页面标题 “超级 AI 画布”，并设置了居中对齐。
2. **用户输入部分**：
   * <input> 元素用于用户输入心情文本，通过 v-model 指令与 emo 响应式数据绑定，实现数据的双向同步。placeholder 属性设置了输入框的提示文本 “输入你的心情”，并应用了 input-field 类样式。
3. **生成图片按钮**：
   * <button> 元素用于触发图片生成操作，点击按钮时调用 generateImage 方法。通过 :disabled="isGenerating" 动态绑定按钮的禁用状态，当 isGenerating 为 true 时，按钮处于禁用状态，防止用户重复点击。按钮应用了 generatebutton 类样式。
4. **加载提示**：
   * <p> 元素用于显示图片生成过程中的加载提示信息 “正在生成图片，请稍候...”。通过 v-if="isGenerating" 指令，只有当 isGenerating 为 true 时才会显示该元素，应用了 loadingtext 类样式。
5. **生成图片展示**：
   * <img> 元素用于展示生成的图片。通过 v-if ="imageUrl" 指令，只有当 imageUrl 有值时才会显示该图片。:src="imageUrl" 动态绑定图片的源地址，alt 属性设置了图片的替代文本 “Generated Image”。同时，为图片绑定了 @mouseenter、@mouseleave 和 @click 事件，分别对应鼠标进入、离开和点击操作，调用 handleMouseEnter、handleMouseLeave 和 editImage 方法。图片应用了 generatedimage 类样式，并设置了宽度和高度为 70%。
6. **提示框**：
   * <div class="tooltip" v-if="isHovered">点击进入创作</div> 用于显示提示框，当鼠标悬停在图片上时，通过 v-if="isHovered" 指令显示该提示框，内容为 “点击进入创作”。
7. **历史图片组件**：
   * <HistoryImages /> 用于引入 HistoryImages 组件，该组件可能用于展示历史生成的图片列表等相关功能。

**三、脚本部分**

1. **导入部分**：
   * import { ref } from 'vue'; 导入 Vue 的 ref 函数，用于创建响应式数据。
   * import { useRouter } from 'vue-router'; 导入 Vue Router 的 useRouter 函数，用于在组件中使用路由功能。
   * import HistoryImages from './HistoryImages.vue'; 导入自定义的 HistoryImages 组件。
2. **响应式数据定义**：
   * const emo = ref('我很高兴'); 创建一个名为 emo 的响应式数据，初始值为 “我很高兴”，用于存储用户输入的心情文本。
   * const imageUrl = ref(null); 创建一个名为 imageUrl 的响应式数据，初始值为 null，用于存储生成图片的 URL 地址。
   * const isGenerating = ref(false); 创建一个名为 isGenerating 的响应式数据，初始值为 false，用于表示图片是否正在生成中。
   * const isHovered = ref(false); 创建一个名为 isHovered 的响应式数据，初始值为 false，用于表示鼠标是否悬停在图片上。
3. **生成图片方法**：
   * generateImage 方法用于生成图片。首先将 isGenerating 设置为 true，表示图片开始生成。
   * 使用 fetch 方法向后端 http://127.0.0.1:5000/main 发送 POST 请求，请求头设置为 Content-Type: application/json，请求体为包含用户输入心情文本 emo.value 的 JSON 字符串。
   * 如果请求成功（response.ok 为 true），将响应数据转换为 blob 格式，并通过 URL.createObjectURL(blob) 创建图片的 URL 地址，赋值给 imageUrl，同时弹出提示框 “图片生成成功！”。
   * 如果请求失败，在控制台打印错误信息 “Failed to generate image”，并弹出提示框 “图片生成失败，请稍后重试。”。
   * 如果在请求过程中发生异常，在控制台打印错误信息 “Error: [具体错误信息]”，并弹出提示框 “发生错误，请稍后重试。”。
   * 无论请求成功与否，最后都将 isGenerating 设置为 false，表示图片生成结束。
4. **编辑图片方法**：
   * editImage 方法用于处理图片编辑操作。首先检查 imageUrl.value 是否有值，如果有值，则使用 router.push 方法跳转到名为 ImageEdit 的路由页面，并将 imageUrl.value 作为查询参数传递过去。
5. **鼠标事件方法**：
   * handleMouseEnter 方法用于处理鼠标进入图片区域的事件，将 isHovered 设置为 true，以显示提示框。
   * handleMouseLeave 方法用于处理鼠标离开图片区域的事件，将 isHovered 设置为 false，以隐藏提示框。

**历史图片组件文档**

**一、整体概述**

该 Vue 组件用于展示历史图片相关功能。用户可以查看历史图片的名称列表，点击图片名称查看具体图片，对图片进行编辑操作，还能删除不需要的图片。组件通过与后端 API 交互，获取历史图片信息并进行相应的增删查操作。

**二、模板部分**

**1. 整体结构**

* 最外层使用 <div> 包裹，设置了相对定位样式，作为整个组件的容器。

**2. 历史图片导航**

* <h1> 标签显示 “历史图片”，作为导航标题。点击该标题会触发 toggleImageNames 方法，用于切换二级导航（图片名称列表）的显示状态。

**3. 二级导航（图片名称列表）**

* <span> 元素包含了图片名称的列表。
* <ul> 列表使用 v-if 指令，根据 showImageNames 的值决定是否显示。
* <li> 元素通过 v-for 指令遍历 historyImages 数组，显示每张图片的名称。每个列表项都有一个唯一的 key（图片的 id），点击列表项会触发 renderImage 方法，用于显示选中的图片。同时，每个列表项都有一个 “删除” 按钮，点击按钮会触发 deleteImage 方法，用于删除对应的图片。

**4. 显示选中的图片**

* <img> 标签使用 v-if 指令，根据 showImage 的值决定是否显示选中的图片。图片的 src 属性绑定到 selectedImageUrl，显示选中图片的具体内容。图片绑定了鼠标进入、离开和点击事件，分别触发 handleMouseEnter、handleMouseLeave 和 editImage 方法。
* <div> 元素作为提示框，根据 isHovered 的值决定是否显示，提示用户点击图片可进入创作。

**三、脚本部分**

**1. 导入部分**

* 从 vue 中导入 ref 和 onMounted。ref 用于创建响应式数据，onMounted 是一个生命周期钩子，用于在组件挂载后执行代码。
* 从 vue-router 中导入 useRouter，用于进行路由跳转。

**2. 响应式数据定义**

* isHovered：用于控制提示框的显示状态，初始值为 false。
* historyImages：存储从后端获取的历史图片信息，初始值为一个空数组。
* showImageNames：控制二级导航（图片名称列表）的显示状态，初始值为 false。
* selectedImageUrl：存储选中图片的 URL，初始值为空字符串。
* selectedImageName：存储选中图片的名称，初始值为空字符串。
* showImage：控制选中图片的显示状态，初始值为 false。

**3. 方法定义**

**fetchHistoryImages**

* 这是一个异步方法，用于向后端 API http://127.0.0.1:5000/history-images 发送 GET 请求，获取历史图片信息。
* 如果请求成功（response.ok 为 true），将响应数据解析为 JSON 格式，并赋值给 historyImages。
* 如果请求失败，捕获错误并在控制台打印错误信息。

**toggleImageNames**

* 用于切换 showImageNames 的值，实现二级导航的显示和隐藏。
* 当 showImageNames 变为 false 时，同时将 showImage 设置为 false，隐藏选中的图片。

**renderImage**

* 接收图片的 id 和 name 作为参数。
* 根据图片的 id 构建图片的 URL，赋值给 selectedImageUrl。
* 将图片的 name 赋值给 selectedImageName。
* 将 showImage 设置为 true，显示选中的图片。

**onMounted**

* 在组件挂载后，调用 fetchHistoryImages 方法，获取历史图片信息。

**editImage**

* 检查 selectedImageUrl 是否有值，如果有值，则使用 router.push 方法跳转到名为 ImageEdit 的路由页面，并将 selectedImageUrl 作为查询参数传递过去。

**handleMouseEnter**

* 当鼠标进入图片区域时，将 isHovered 设置为 true，显示提示框。

**handleMouseLeave**

* 当鼠标离开图片区域时，将 isHovered 设置为 false，隐藏提示框。

**deleteImage**

* 这是一个异步方法，接收图片的 id 作为参数。
* 向后端 API http://127.0.0.1:5000/delete-image/${imageId} 发送 DELETE 请求，删除指定的图片。
* 如果请求成功（response.ok 为 true），调用 fetchHistoryImages 方法更新历史图片列表。
* 如果请求失败，捕获错误并在控制台打印错误信息。

**图片编辑组件文档**

**一、整体概述**

该 Vue 组件实现了一个图片编辑功能页面，用户可以在已有图片的基础上进行绘图、擦除等操作，并能够保存编辑后的图片。组件通过与后端 API 交互实现图片的保存功能，同时提供了取消编辑返回上一页的操作。

**二、模板部分**

**（一）标题部分**

<h1 style="text-align: center;">发挥想象力，创作属于自己的画布</h1>：展示页面标题，提示用户该页面的功能是进行图片创作。

**（二）工具图标容器**

1. <div class="tool-icons">：包含绘图工具图标的容器。
2. <h3>绘图工具</h3>：工具区域的标题。
3. 三个 <el-icon> 元素：
   * 每个 <el-icon> 代表一种绘图工具（画笔、橡皮擦、彩色画笔）。
   * @click="toggleTool('brush')" 等点击事件，当用户点击图标时，调用 toggleTool 方法并传入对应的工具名称，用于切换当前使用的工具。
   * :class="{ 'active-icon': currentTool === 'brush' }" 根据当前激活的工具，动态添加 active-icon 类，以改变图标的样式，突出显示当前使用的工具。

**（三）画布容器**

<div class="canvas-container">：包含两个画布元素，用于实现图片编辑功能。

1. <canvas ref="backgroundCanvas"></canvas>：作为背景画布，用于显示从路由参数中获取的原始图片。
2. <canvas ref="drawingCanvas"></canvas>：用于用户进行绘图、擦除等操作，用户在该画布上的操作不会影响背景画布的原始图片。

**（四）按钮组**

<div class="button-group">：包含保存和取消编辑按钮的容器。

1. <button @click="showSaveModal" class="save-button">保存</button>：点击该按钮调用 showSaveModal 方法，用于显示保存图片的模态框。
2. <button @click="cancelEdit" class="cancel-button">不保存</button>：点击该按钮调用 cancelEdit 方法，用于取消当前编辑并返回上一页。

**（五）自定义模态框**

1. <div v-if="isSaveModalVisible" class="modal">：当 isSaveModalVisible 为 true 时显示该模态框，用于用户输入保存图片的名称并执行保存操作。
2. <div class="modal-content">：模态框的内容区域。
   * <span class="close" @click="hideSaveModal">&times;</span>：关闭模态框的按钮，点击调用 hideSaveModal 方法。
   * <h2>保存图片</h2>：模态框的标题。
   * <input v-model="imageName" placeholder="请输入图片名称" />：用户输入图片名称的输入框，通过 v-model 与 imageName 响应式数据绑定。
   * <div class="modal-buttons">：包含取消和确认保存按钮的容器。
     + <button @click="hideSaveModal" class="modal-cancel-button">取消</button>：点击取消保存并关闭模态框。
     + <button @click="saveImageAndClose" class="modal-confirm-button">确认</button>：点击调用 saveImageAndClose 方法，执行保存图片的操作。

**三、脚本部分**

**（一）导入部分**

1. import { ref, onMounted } from 'vue';：导入 Vue 的 ref 函数用于创建响应式数据，onMounted 生命周期钩子用于在组件挂载后执行代码。
2. import { useRouter, useRoute } from 'vue-router';：导入 Vue Router 的 useRouter 和 useRoute 函数，分别用于进行路由跳转和获取路由参数。
3. import { Edit, Close, Brush } from '@element-plus/icons-vue';：从 ElementPlus 图标库中导入所需的图标组件，用于在模板中显示绘图工具图标。

**（二）响应式数据定义**

1. const router = useRouter(); 和 const route = useRoute();：获取 Vue Router 的路由实例和当前路由对象。
2. const backgroundCanvas = ref(null); 和 const drawingCanvas = ref(null);：分别用于引用背景画布和绘图画布的 DOM 元素。
3. const backgroundCtx = ref(null); 和 const drawingCtx = ref(null);：分别用于获取背景画布和绘图画布的 2D 绘图上下文。
4. const isDrawing = ref(false);：用于标记用户是否正在进行绘图操作，初始值为 false。
5. const lastX = ref(0); 和 const lastY = ref(0);：用于记录用户绘图时上一个点的坐标，初始值为 0。
6. const imageName = ref('');：用于存储用户输入的保存图片的名称，初始值为空字符串。
7. const isSaveModalVisible = ref(false);：用于控制保存图片模态框的显示状态，初始值为 false。
8. const currentTool = ref('brush');：用于记录当前使用的绘图工具，初始值为 brush（画笔）。

**（三）方法定义**

1. **绘图相关方法**：
   * startDrawing(e)：当用户在绘图画布上按下鼠标时触发。如果绘图上下文不存在则返回，否则设置 isDrawing 为 true，并根据鼠标在画布上的位置计算出对应的坐标，记录在 lastX 和 lastY 中。
   * draw(e)：当用户在绘图画布上移动鼠标时触发。如果绘图上下文不存在或用户没有在绘图状态则返回，否则根据鼠标当前位置计算坐标，开始绘制路径，并根据当前使用的工具设置绘图样式（画笔颜色、线条宽度、橡皮擦模式等），绘制完成后恢复默认绘制模式，并更新 lastX 和 lastY。
   * stopDrawing()：当用户松开鼠标或鼠标移出绘图画布时触发，设置 isDrawing 为 false，结束绘图操作。
2. **模态框相关方法**：
   * showSaveModal()：点击保存按钮时调用，设置 isSaveModalVisible 为 true，显示保存图片的模态框。
   * saveImageAndClose()：点击模态框中的确认保存按钮时调用。首先检查 imageName 是否为空，若为空则提示用户输入图片名称。然后创建一个临时画布，将背景画布和绘图画布的内容合并到临时画布上，获取临时画布的 Base64 编码数据，并向后端 http://127.0.0.1:5000/save 发送 POST 请求保存图片。如果保存成功，提示用户并关闭模态框，跳转到首页；如果失败，提示用户保存失败。
   * hideSaveModal()：点击模态框中的取消按钮或关闭按钮时调用，设置 isSaveModalVisible 为 false，并清空 imageName。
3. **其他方法**：
   * cancelEdit()：点击取消编辑按钮时调用，使用 router.push('/') 跳转到首页，取消当前编辑操作。
   * toggleTool(tool)：当用户点击绘图工具图标时调用，将 currentTool 设置为用户点击的工具名称，实现工具切换。
4. **生命周期钩子**：
   * onMounted(() => {... })：在组件挂载后执行。创建背景画布和绘图画布的 DOM 元素，并获取它们的 2D 绘图上下文。从路由参数 route.query.imageUrl 中获取图片 URL，创建一个 Image 对象并设置其 src 为该 URL。当图片加载完成后，设置背景画布和绘图画布的实际像素大小和 CSS 样式大小，并将图片绘制到背景画布上。同时为绘图画布添加鼠标按下、移动、松开和移出事件监听器，分别对应 startDrawing、draw、stopDrawing 方法。