Vue3 和表單

在 Vue3 中處理表單、驗證和錯誤處理是開發中常見的需求，我們將通過下面的內容和範例來進行詳細解說。

表單處理基礎

在 Vue3 中，表單處理通常涉及到模板綁定（Template Binding）和響應式數據（Reactive Data）。Vue3 提供了 `v-model` 指令來實現雙向數據綁定，這對於表單處理特別有用。

範例：基本文本輸入

```
<template>
  <form @submit.prevent="submitForm">
    <input v-model="formData.name" type="text" placeholder="請輸入您的名字" />
    <button type="submit">提交</button>
  </form>
</template>

<script setup>
import { ref } from 'vue';

const formData = ref({
  name: ''
});

const submitForm = () => {
  alert(`提交的名字是：${formData.value.name}`);
};
</script>
```

在這個範例中，我們使用了 `v-model` 來綁定一個文本輸入框的值到 `formData.name`。表單提交時，將阻止其預設行為（刷新頁面），並通過 `submitForm` 方法彈出一個對話框顯示輸入的名字。

表單驗證和錯誤處理

驗證表單輸入是確保數據質量和用戶體驗的重要一環。Vue3 本身不包含內置的表單驗證功能，但我們可以通過簡單的邏輯和一些第三方庫來實現。

範例：簡單的表單驗證

假設我們需要確保用戶名不為空，並且年齡是一個有效的數字：

```
<template>
  <form @submit.prevent="submitForm">
    <div>
      <input v-model="formData.name" type="text" placeholder="姓名" />
      <p v-if="errors.name">{{ errors.name }}</p>
    </div>
    <div>
      <input v-model.number="formData.age" type="number" placeholder="年齡" />
      <p v-if="errors.age">{{ errors.age }}</p>
    </div>
    <button type="submit">提交</button>
  </form>
</template>

<script setup>
import { ref, reactive, watchEffect } from 'vue';

const formData = reactive({
  name: '',
  age: null
});

const errors = reactive({
  name: '',
  age: ''
});

watchEffect(() => {
  if (!formData.name) {
    errors.name = '姓名為必填項目';
  } else {
    errors.name = '';
  }

  if (!formData.age || isNaN(formData.age) || formData.age < 0) {
    errors.age = '請輸入有效的年齡';
  } else {
    errors.age = '';
  }
});

const submitForm = () => {
  if (!errors.name && !errors.age) {
    alert(`提交成功：${formData.name}, ${formData.age}`);
  } else {
    alert('表單有誤，請修正後再次提交');
  }
};
</script>
```

在這個範例中，我們使用了 `reactive` 函數來創建一個響應式的表單數據對象和錯誤信息對象。利用 `watchEffect` 監聽 `formData` 的變化，並進行實時驗證，更新 `errors` 對象。表單提交時，檢查是否有錯誤信息，如果沒有則彈出成功提交的提示。

script setup

在Vue.js或類似的現代前端框架中，`<script setup>`和普通的`<script>`標籤都被用來撰寫JavaScript或TypeScript代碼，但它們在使用和語法上有顯著的不同：

`<script setup>`

- **Composition API 精簡語法**：`<script setup>`是一種針對Vue 3引入的精簡語法，旨在使使用Composition API更加簡潔和易於閱讀。它允許你直接定義組件的setup函數內容，而無需明確定義一個`setup`函數。
- **自動導入**：在`<script setup>`中聲明的變量和方法會自動被視為組件的一部分，可以直接在模板中使用，無需額外的return語句。
- **更好的類型推斷**：使用`<script setup>`可以獲得更好的TypeScript類型推斷支持。

- **簡化的API使用**：某些Vue特性，如`defineProps`和`defineEmits`，可以直接在`<script setup>`中使用，無需導入。

`<script>`

- **傳統選項API**：使用普通的`<script>`標籤通常意味著你在使用Vue的選項API。這是Vue較早版本中使用的方法，通過定義一個包含`data`、`methods`、`props`等選項的對象來創建組件。
- **顯式定義**：在選項API中，所有的數據和方法都需要被顯式地包裝在對象選項中，並且需要使用`this`關鍵字來訪問組件的上下文。
- **更適合簡單組件**：對於一些簡單的組件，使用選項API和普通的`<script>`標籤可能會更簡單直觀。

結論

選擇`<script setup>`還是普通的`<script>`取決於你的具體需求和偏好。如果你喜歡更現代、更簡潔的語法，並且正在使用Vue 3的Composition API，那麼`<script setup>`可能是更好的選擇。如果你正在處理一個簡單的組件，或者更習慣於Vue的傳統選項API，那麼使用普通的`<script>`可能會更適合你。