Vue3 組件

在 Vue 3中，組件是構建應用的基本單元，它們允許你將界面拆分成獨立可復用的部分，以便更好地管理和擴展你的應用。下面，我們將詳細介紹如何創建和使用組件、組件間的屬性和通信，以及組件生命周期鉤子的概念。

創建和使用組件

Vue 3支持使用單文件組件（Single File Components，SFCs），這些組件將模板、JavaScript 和 CSS 封裝在單個 `.vue` 文件中。首先，我們來創建一個簡單的組件：

```
<!-- MyComponent.vue -->
<template>
  <div>Hello, Vue 3!</div>
</template>

<script>
export default {
  name: 'MyComponent'
}
</script>

<style scoped>
div {
  color: blue;
}
</style>
```

要在另一個組件或應用中使用這個組件，你需要在父組件中導入並註冊它：

```
<template>
  <div>
    <MyComponent/>
  </div>
</template>

<script>
import MyComponent from './MyComponent.vue';

export default {
  components: {
    MyComponent
  }
}
</script>
```

屬性和組件通信

組件間的通信通常通過屬性（props）和事件來實現。父組件可以通過屬性向子組件傳遞數據，而子組件則可以通過發射事件來與父組件通信。

Props

```
<!-- ChildComponent.vue -->
<template>
  <div>{{ message }}</div>
</template>

<script>
export default {
  props: ['message']
}
</script>
```

在父組件中，你可以這樣使用它：

```
<ChildComponent message="Hello from parent!" />
```

Emit Events

子組件可以使用`$emit`方法發射事件：

```
<button @click="$emit('custom-event', 'some data')">Click Me</button>
```

父組件可以通過監聽這個事件來響應：

```
<ChildComponent @custom-event="handleEvent" />

<script>
export default {
  methods: {
    handleEvent(data) {
      console.log(data); // 'some data'
    }
  }
}
</script>
```

組件生命周期鉤子

Vue 3組件有多個生命周期鉤子，允許你在組件的不同階段執行代碼：

- `beforeCreate`：在組件實例創建之前執行。
- `created`：在組件實例創建完成後執行。
- `beforeMount`：在組件掛載到DOM之前執行。
- `mounted`：在組件掛載完成後執行。
- `beforeUpdate`：在數據更新之前執行。
- `updated`：在數據更新後執行。
- `beforeUnmount`：在組件卸載之前執行。
- `unmounted`：在組件卸載後執行。

以下是一個生命周期鉤子的示例：

```
<script>
export default {
  mounted() {
    console.log('Component is mounted!');
  }
}
</script>
```

這些生命周期鉤子為你提供了在組件的不同階段介入並執行邏輯的能力，有助於管理組件的資源和行為。

通過結合使用組件、屬性、事件以及生命周期鉤子，你可以構建出功能豐富且高度可復用的Vue 3應用。

事件宣告：Vue 自定義事件警告

這個錯誤提示表明你在 Vue.js 組件中使用了一個自定義事件 `customEvent`，但是沒有在組件中透過 `emits` 選項來聲明它。當你在父組件中使用子組件並嘗試綁定一個自定義事件到子組件上時，如果子組件沒有透過 `emits` 來明確聲明這個事件是一個自定義事件，Vue 將會顯示這個警告。

解決這個警告的方法是在子組件中使用 `emits` 選項來聲明你的自定義事件。下面是一個簡單的例子，展示了如何修正這個問題：

子組件 (MyComponent.vue)

```
<template>
  <!-- 子組件的模板內容 -->
</template>

<script>
export default {
  props: {
    // 宣告 props，比如這裡的 message
    message: String,
  },
  emits: [
    'customEvent' // 在這裡聲明自定義事件
  ],
  methods: {
    someMethod() {
      // 假設在某些條件下，你想要觸發這個自定義事件
      this.$emit('customEvent', { /* 傳遞給父組件的數據 */ });
    }
  }
}
</script>
```

父組件 (App.vue)

在父組件中，你可以這樣使用 `MyComponent` 並監聽 `customEvent`：

```
<template>
  <MyComponent message="Justin" @customEvent="handleCustomEvent" />
</template>

<script>
import MyComponent from './MyComponent.vue';

export default {
  components: {
    MyComponent
  },
  methods: {
    handleCustomEvent(eventData) {
      // 處理自定義事件
    }
  }
}
</script>
```

通過這樣的修改，你就可以消除 Vue 的警告，並正確地使用自定義事件。這種方式確保了組件之間的通訊清晰且易於維護。

Vue.js 生產環境配置

特性標誌（Feature Flags）和樹搖（Tree-shaking）。特性標誌允許 Vue 在編譯時進行更精細的優化，從而在生產環境中得到更小的打包體積和更好的性能。當你看到這樣的警告時，這意味著你需要在打包工具的配置中顯式地設置這些特性標誌。

假設你正在使用的是 Webpack 作為打包工具，下面是一種方法來解決這個問題：

1. 在你的 `webpack.config.js` 文件中，需要使用 `DefinePlugin` 插件來定義這些特性標誌。`DefinePlugin` 允許你創建一個在編譯時可以配置的全局常量。
2. 對於 `__VUE_PROD_HYDRATION_MISMATCH_DETAILS__` 這個特定的特性標誌，你可以根據你的需要將其設置為 `true` 或 `false`。如果你希望在生產模式下獲取更多關於水合不匹配的詳細錯誤信息，可以將其設置為 `true`。通常情況下，為了減小打包體積，你應該將其設置為 `false`。

以下是 `webpack.config.js` 中的一個範例配置片段：

```
const { DefinePlugin } = require('webpack');

module.exports = {
  // 其他配置...
  plugins: [
    // 其他插件...
    new DefinePlugin({
      // 在這裡定義你的特性標誌
      __VUE_OPTIONS_API__: JSON.stringify(true),
      __VUE_PROD_DEVTOOLS__: JSON.stringify(false),
      // 將 __VUE_PROD_HYDRATION_MISMATCH_DETAILS__ 設置為 false 來禁用生產環境的水合不匹配詳情
      __VUE_PROD_HYDRATION_MISMATCH_DETAILS__: JSON.stringify(false),
    }),
  ],
};
```

記住，在修改了 Webpack 配置之後，需要重啟你的打包過程以使更改生效。

如果你使用的是其他打包工具（如 Rollup、Vite 等），則需要查找相應的配置方法來定義這些特性標誌。大多數現代打包工具都有類似於 Webpack 的 `DefinePlugin` 的機制來實現這一點。

專案使用 Vue CLI 打包

參考官方文件https://github.com/vuejs/core/tree/main/packages/vue#bundler-build-feature-flags (https://github.com/vuejs/core/tree/main/packages/vue#bundler-build-feature-flags)

在使用 Vue CLI 的 Vue 3 專案中設定 `__VUE_PROD_HYDRATION_MISMATCH_DETAILS__` 為 `true`，需要在專案的根目錄下創建或修改 `vue.config.js` 文件，並在其中使用 webpack 的 `DefinePlugin` 來定義該變量。這是因為 Vue CLI 內部使用 webpack 作為其打包工具，而 `DefinePlugin` 允許你創建全局常量。

以下是如何進行設定的步驟：

1. 在專案的根目錄下找到或創建一個 `vue.config.js` 文件。
2. 修改 `vue.config.js` 文件，加入 `DefinePlugin` 的配置，如下：

```
const { defineConfig } = require('@vue/cli-service');
const webpack = require('webpack');

module.exports = defineConfig({
  configureWebpack: {
    plugins: [
      new webpack.DefinePlugin({
        __VUE_PROD_HYDRATION_MISMATCH_DETAILS__: true
      })
    ]
  }
});
```

3. 保存文件並重新啟動你的 Vue CLI 服務（例如，重新運行 `npm run serve`）。

這樣設定後，`__VUE_PROD_HYDRATION_MISMATCH_DETAILS__` 將在你的 Vue 3 應用中全局可用，並且被設置為 `true`。這意味著在生產環境中，如果出現水合錯誤，Vue 將提供更詳細的錯誤資訊，幫助你進行調試。

請注意，由於這會增加生產環境中的錯誤詳情，可能會對錯誤報告的清晰度有所幫助，但也可能會略微增加打包後的應用大小。因此，你應該基於專案的實際需要來決定是否啟用此設定。