

Vue 組件結構：生命週期



1. Vue 組件生命週期：組件在網頁上運件時的重要時機點，對應三個建的 Hooks函式：



2. 組件生命週期，觸發時機說明：

這裏有三個組件，今天焦點放在 App.vue 及 Main.vue。我們在上層組件 App.vue 載入 Main.vue 這個組件(import Main from “./Main.vue” 把 Main 這個組件畫在畫面上(<Main v-if=“visible”></Main>，當 Main 這個組件第一次畫到畫面上時就會觸發掛載的時機，稱之為=》

onMounted，

再看 Main.vue 有一個響應式狀態 text (let text=ref(“這是一段文字內容”)), text 就顯示在內文中 (<main @click=“updateText”>{{ text}})，當我們點擊內文，就會去改變響應式狀態的內容

(let updateText=function(){ text.value=“更新過的文字內容”})，當響應式狀態內容有改變，就會導致 main 組件被重新繪

製，這就是所謂組件更新的時機點，稱之為 onUpdated，

最後回頭看 App.vue，Main 組件透過 visible的狀態(

```
let visible=ref(true);
```

```
let hide=function(){
```

```
  visible.value=false; } )
```

來管理這個組件是否要顯示 <Main v-if=“visible”></Main>，如果 visible = true 就把 main 顯示出來，default 設定為 true，所以一開始會掛載，然後設計一個按鈕叫「隱藏」，當按下時呼叫 hide 函式，把 visible 改成 false，main 組件就會從畫面中消失，從畫面中被移除，這個時候會觸發 onUnmounted，稱為組件卸載。



3. 生命週期語法：從 vue 載入 onMounted , onUpdated , onUnmounted

使用生命週期 Hooks

載入 Hooks · 傳入要執行的函式

- 載入 Hooks
`import {onMounted, onUpdated, onUnmounted} from "vue";`
- 傳入對應的生命週期觸發事件
`onMounted`(組件掛載時要執行的函式)
`onUpdated`(組件更新時要執行的函式)
`onUnmounted`(組件卸載時要執行的函式)

5:56 / 34:34

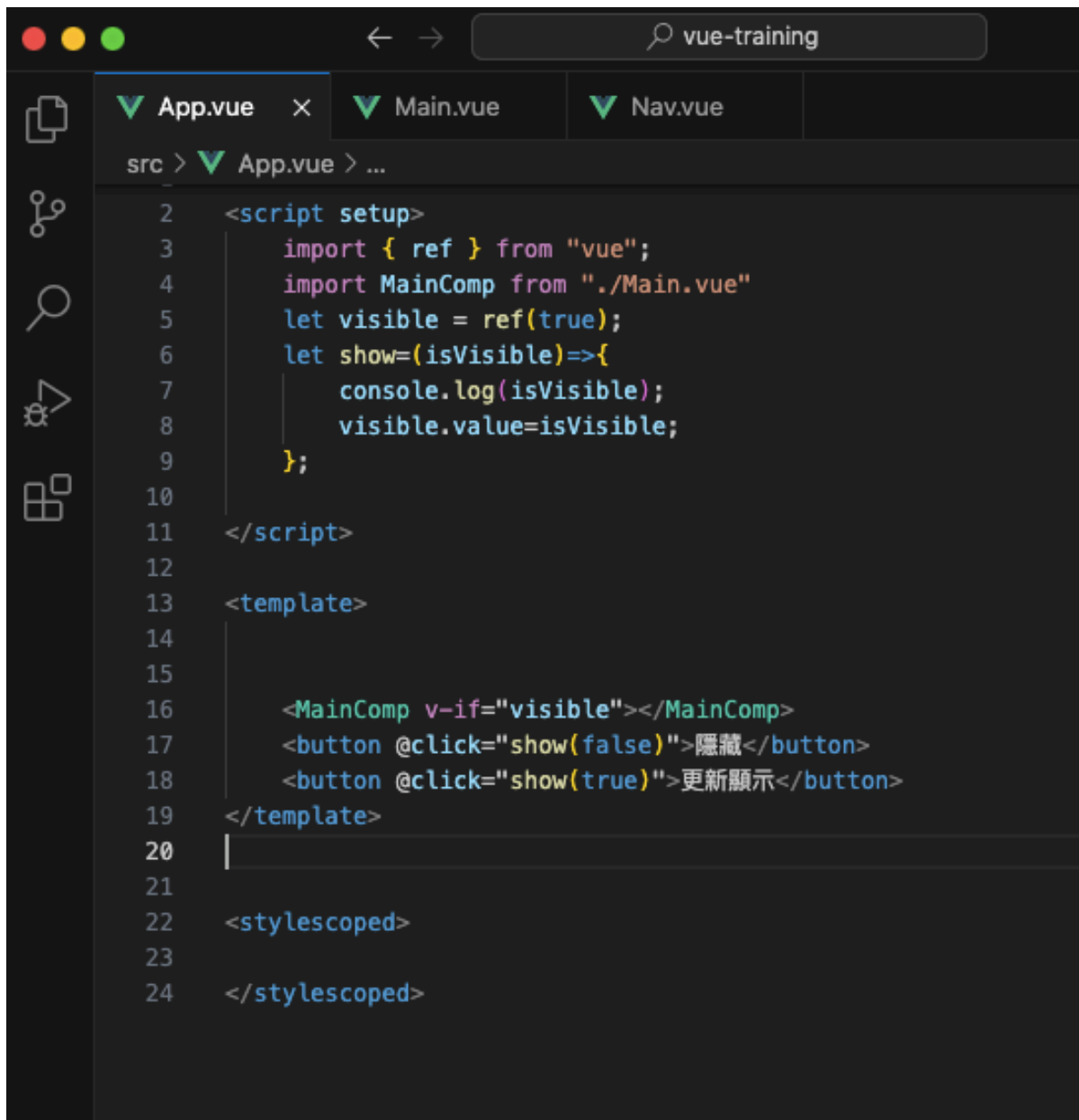
4. 基本使用案例:

有一個響應式狀態 叫 text (let text=ref("這是一段文字內容")) ， 這個 text 會顯示在畫面上 (main @click="updateText"> {{ text}}) ， 當點擊文字時， 會執行 updateText 函式， 更新響應式狀態， 把 text 的文字做一個更新， 如果我們希望文字有更新時， 可以做一些額外的動作， 就可以利用 onUpdated 這個 Hook, 在 script 裏這樣寫

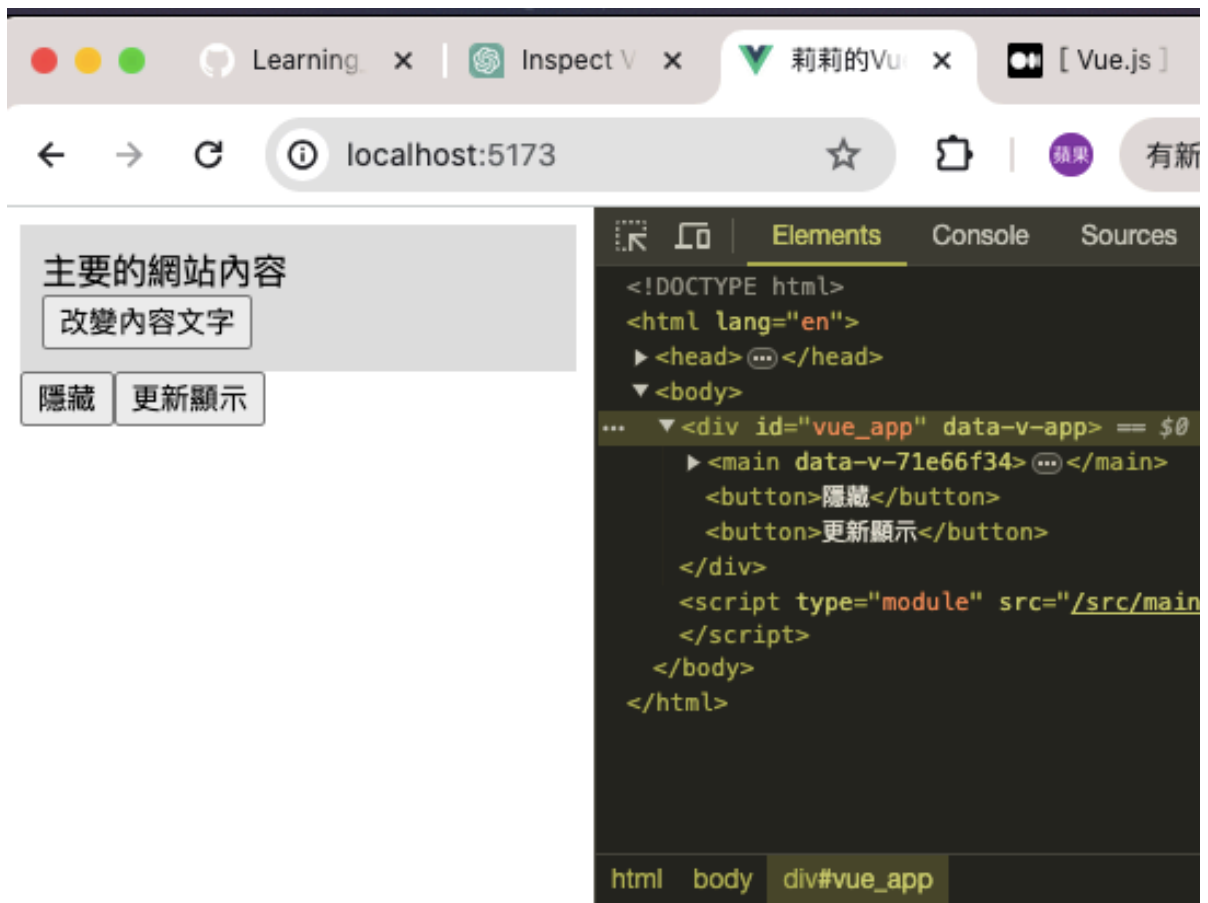
```
import ( onUpdated , ref ) from "vue"; // 二個函式一起載入
onUpdated function(){
  console.log("組件文字已更新");
};
```



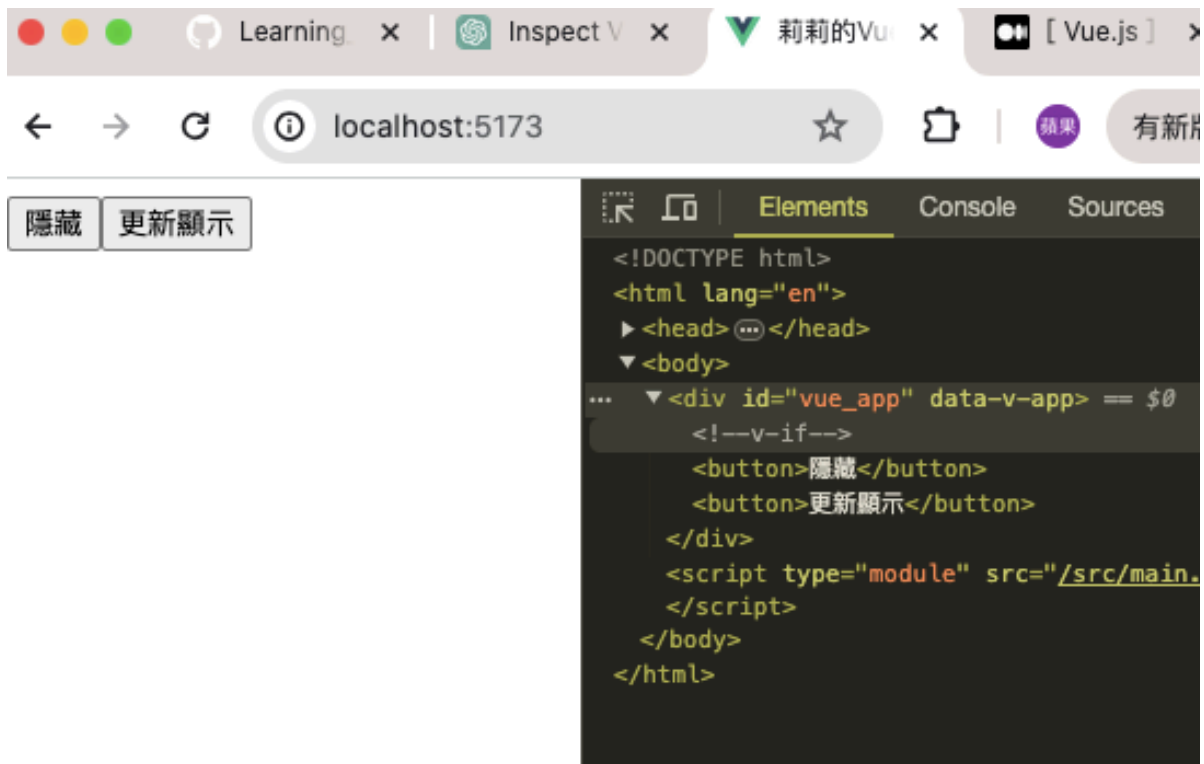
5. 先測試 生命週期 如何運作，先準備一個情境，可以隱藏及顯示 main component：
在 App.vue 載入響應式狀態函式，設定一個 visible 變數，初始值為 true，
然後做二個 button，一個為顯示，一個為隱藏，然後看 Elements 變化：



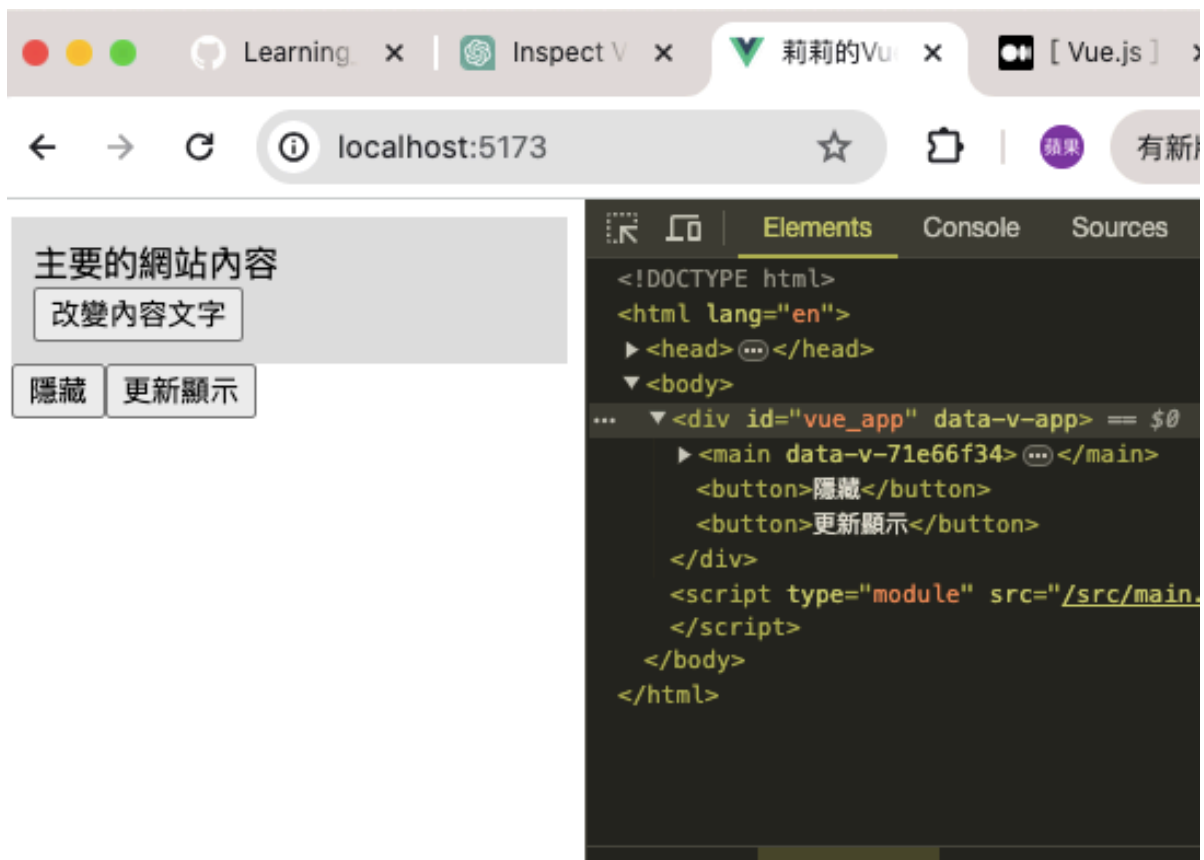
```
2 <script setup>
3   import { ref } from "vue";
4   import MainComp from "./Main.vue"
5   let visible = ref(true);
6   let show=(isVisible)=>{
7     console.log(isVisible);
8     visible.value=isVisible;
9   };
10
11 </script>
12
13 <template>
14
15
16   <MainComp v-if="visible"></MainComp>
17   <button @click="show(false)">隱藏</button>
18   <button @click="show(true)">更新顯示</button>
19 </template>
20
21
22 <style scoped>
23
24 </style>
```



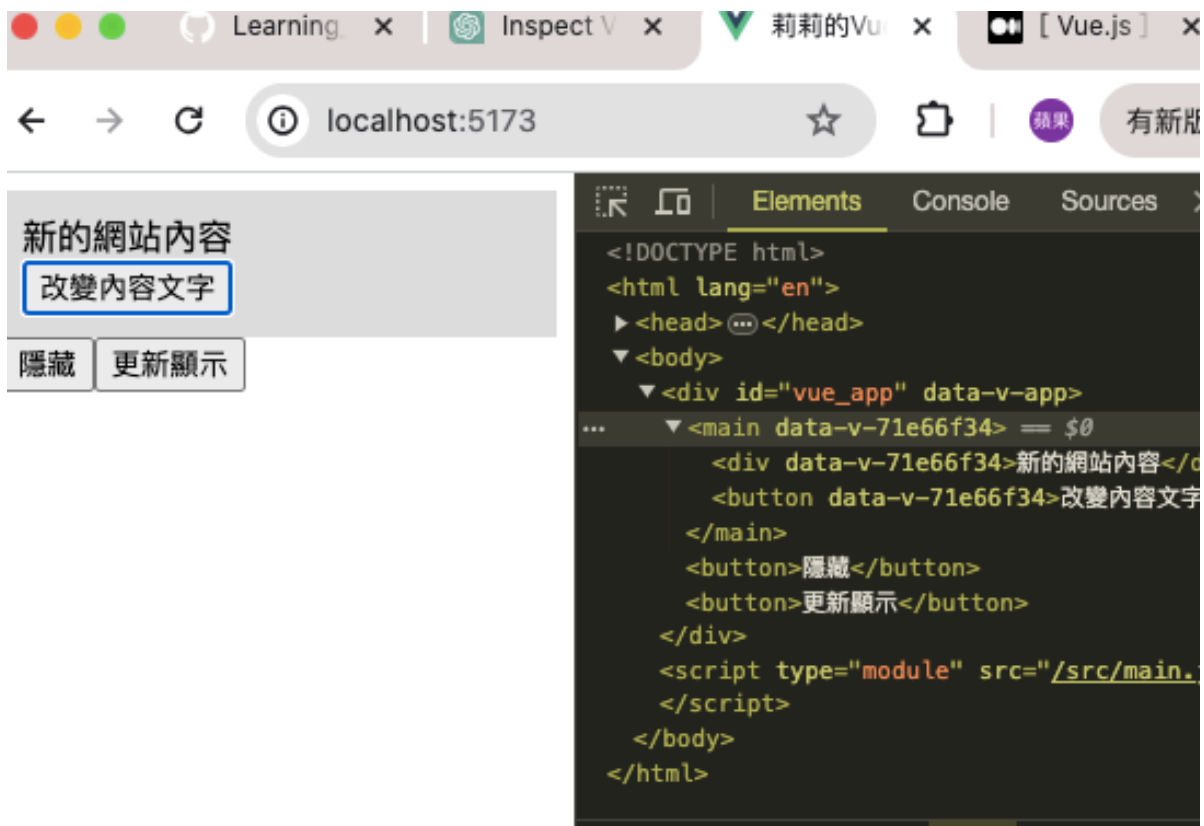
當按下 隱藏 按鈕時，MainComp 被隱藏了，就是卸載概念 onUnmounted：



當 按下 更新顯示 ， 又出現了，掛載概念，onMount：

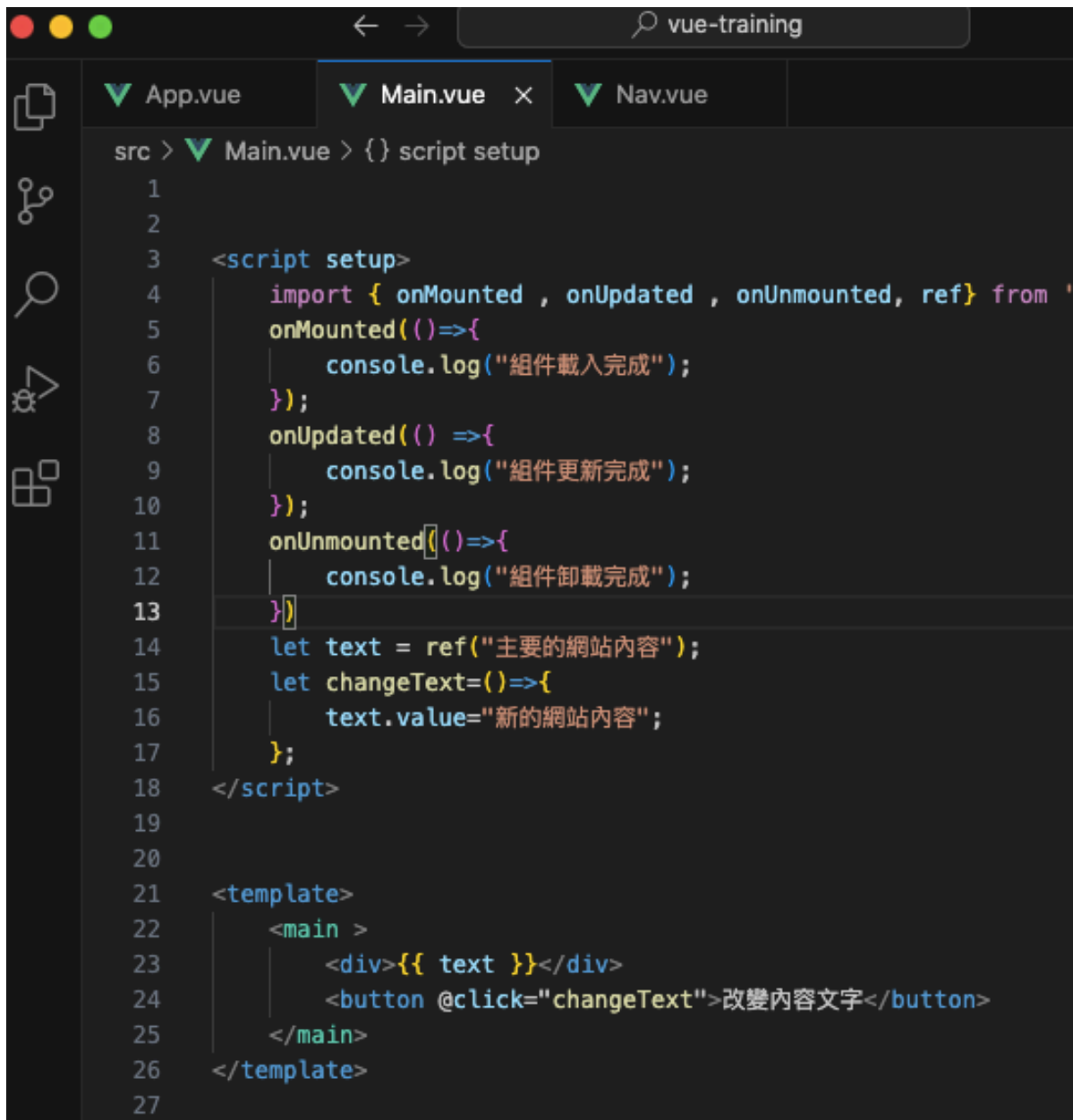


當按下「改變內容文字」則顯示 新的網站內容，就是更新的概念 onUpdated：



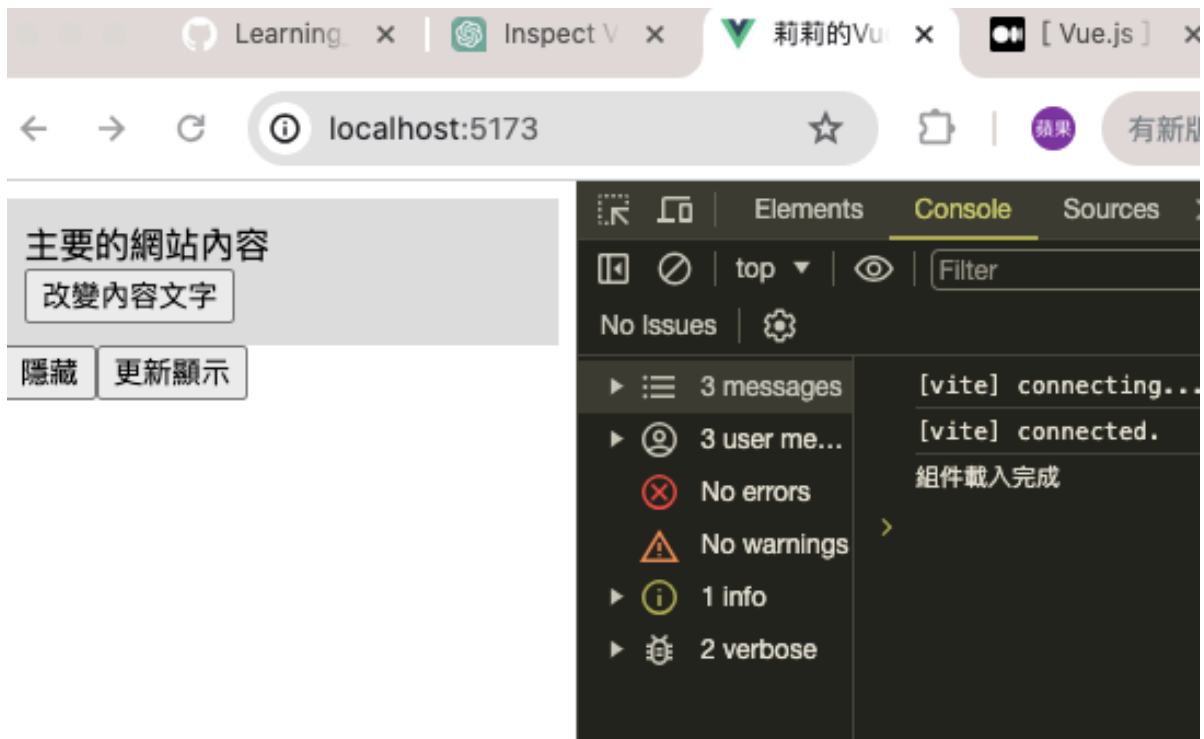
6. 所以我們可以 在 Main.vue 載入 Hooks 這三個函式 onMounted，onUpdated，onUnmounted，來觀查

這三個函式作用的時機點：

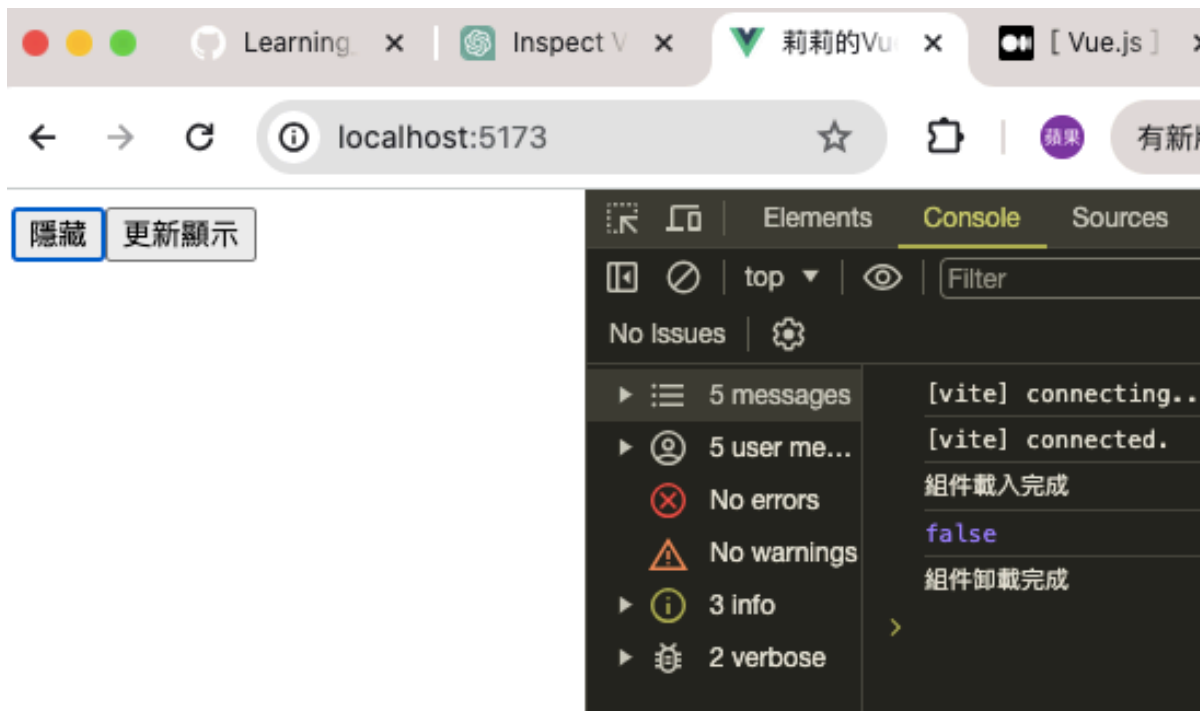


```
src > ▼ Main.vue > {} script setup
1
2
3   <script setup>
4     import { onMounted , onUpdated , onUnmounted, ref } from '
5     onMounted(()=>{
6       console.log("組件載入完成");
7     });
8     onUpdated(() =>{
9       console.log("組件更新完成");
10    });
11    onUnmounted(()=>{
12      console.log("組件卸載完成");
13    });
14    let text = ref("主要的網站內容");
15    let changeText=()=>{
16      text.value="新的網站內容";
17    };
18  </script>
19
20
21  <template>
22    <main >
23      <div>{{ text }}</div>
24      <button @click="changeText">改變內容文字</button>
25    </main>
26  </template>
27
```

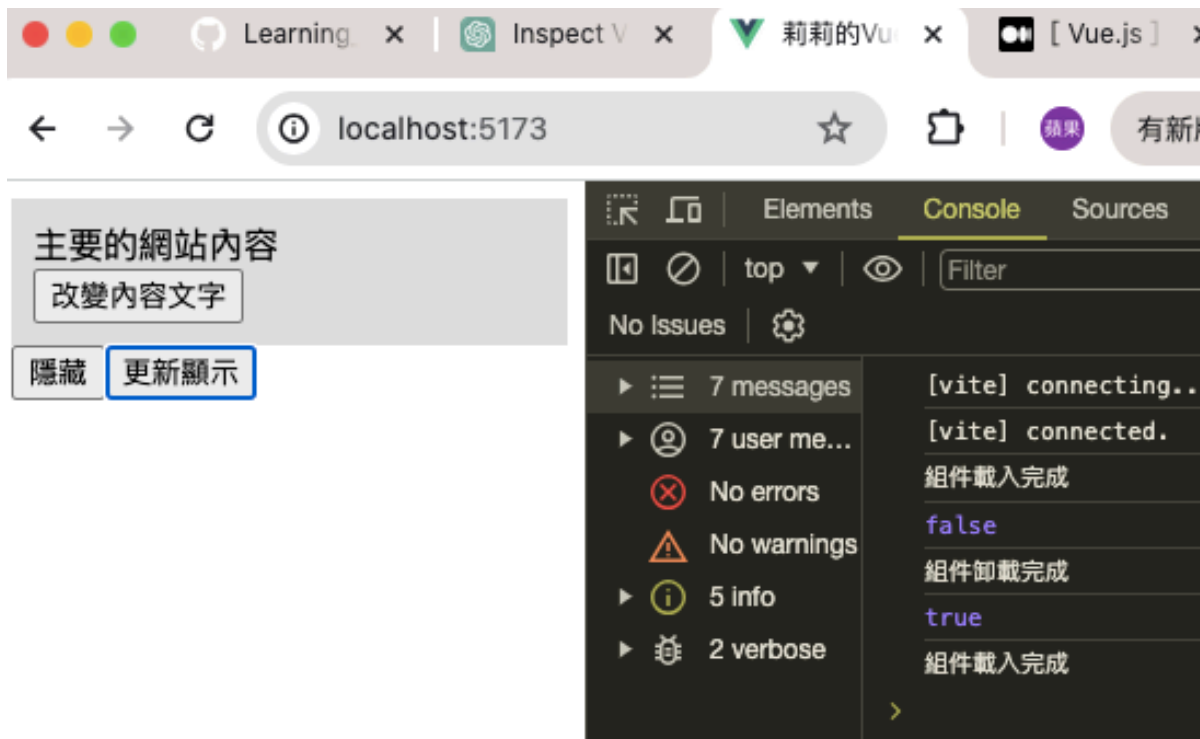
當網頁首次 load 進來時：log 顯示 組件載入完成：



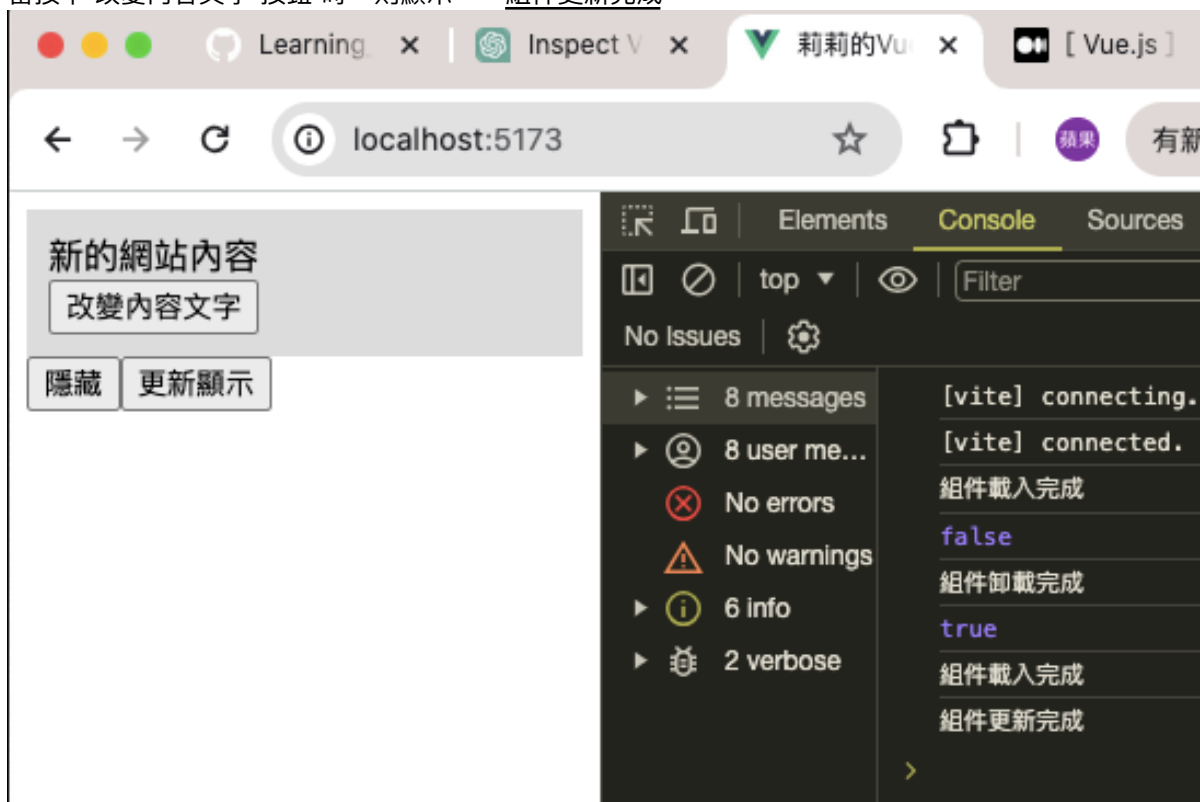
當按下隱藏 按鈕 時，顯示 組件卸載完成：



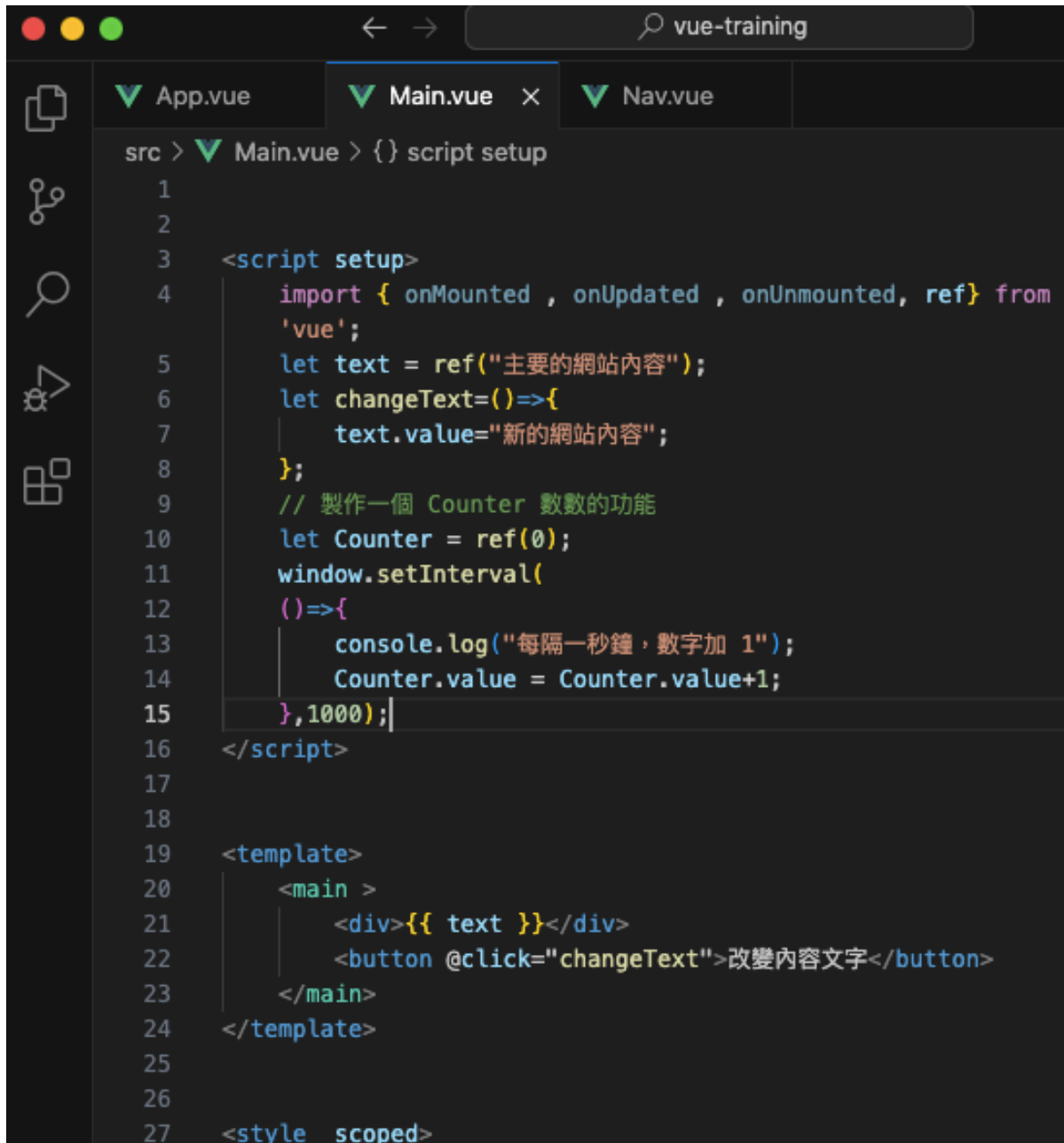
當按下 更新顯示 按鈕 時，顯示 組件載入完成：



當按下 改變內容文字 按鈕 時，則顯示：組件更新完成：

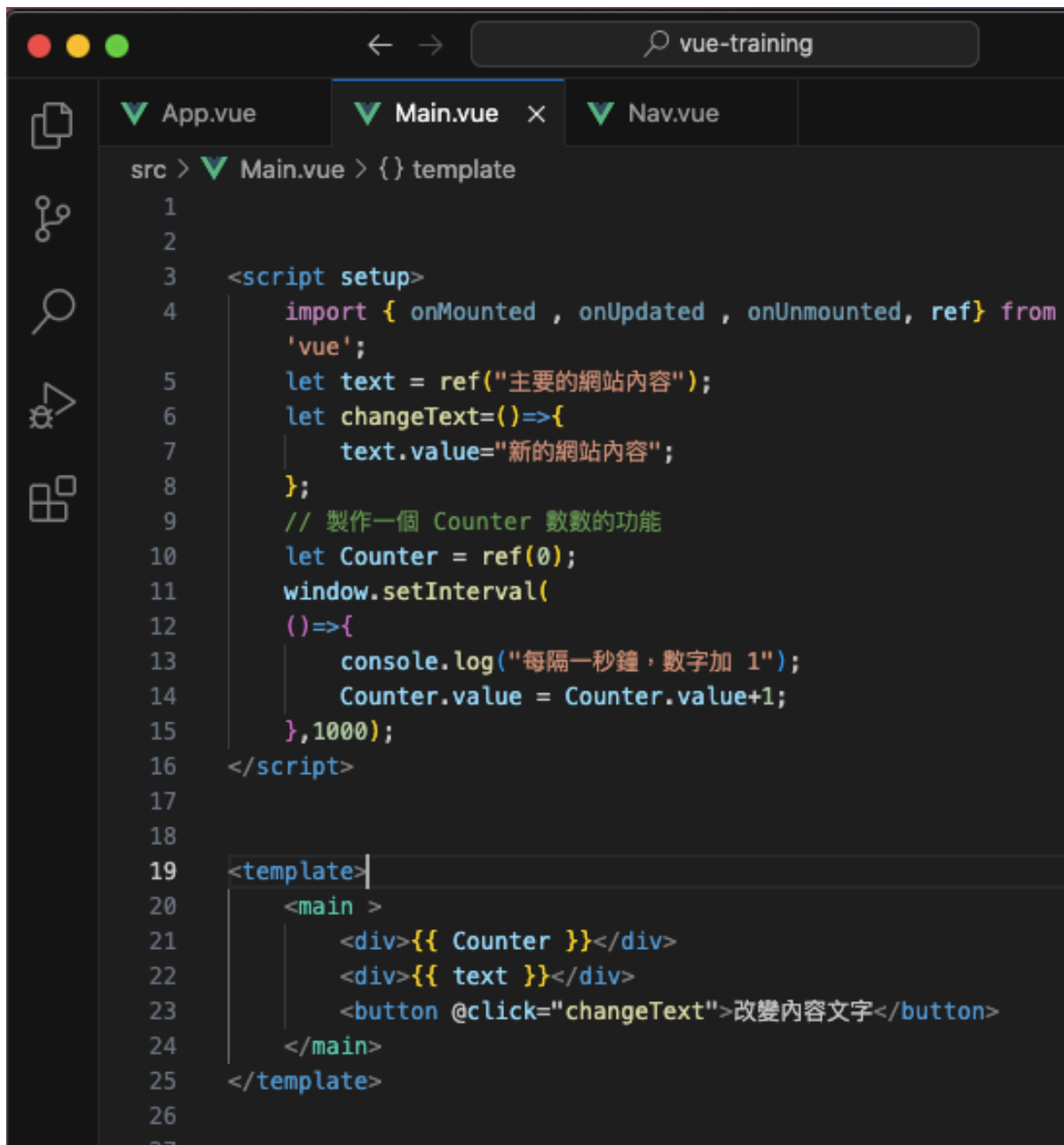


7. 談一個真實情境：如果沒有好好處理 這個組件會有狀況，在 main 做一個計時器，就時數數 1, 2 3, 4. 5 的功能 Counter，建立一個響應式狀態 叫 Counter:，利用 window 的內建函式 setInterval (數字 1000 代表 一秒鐘):



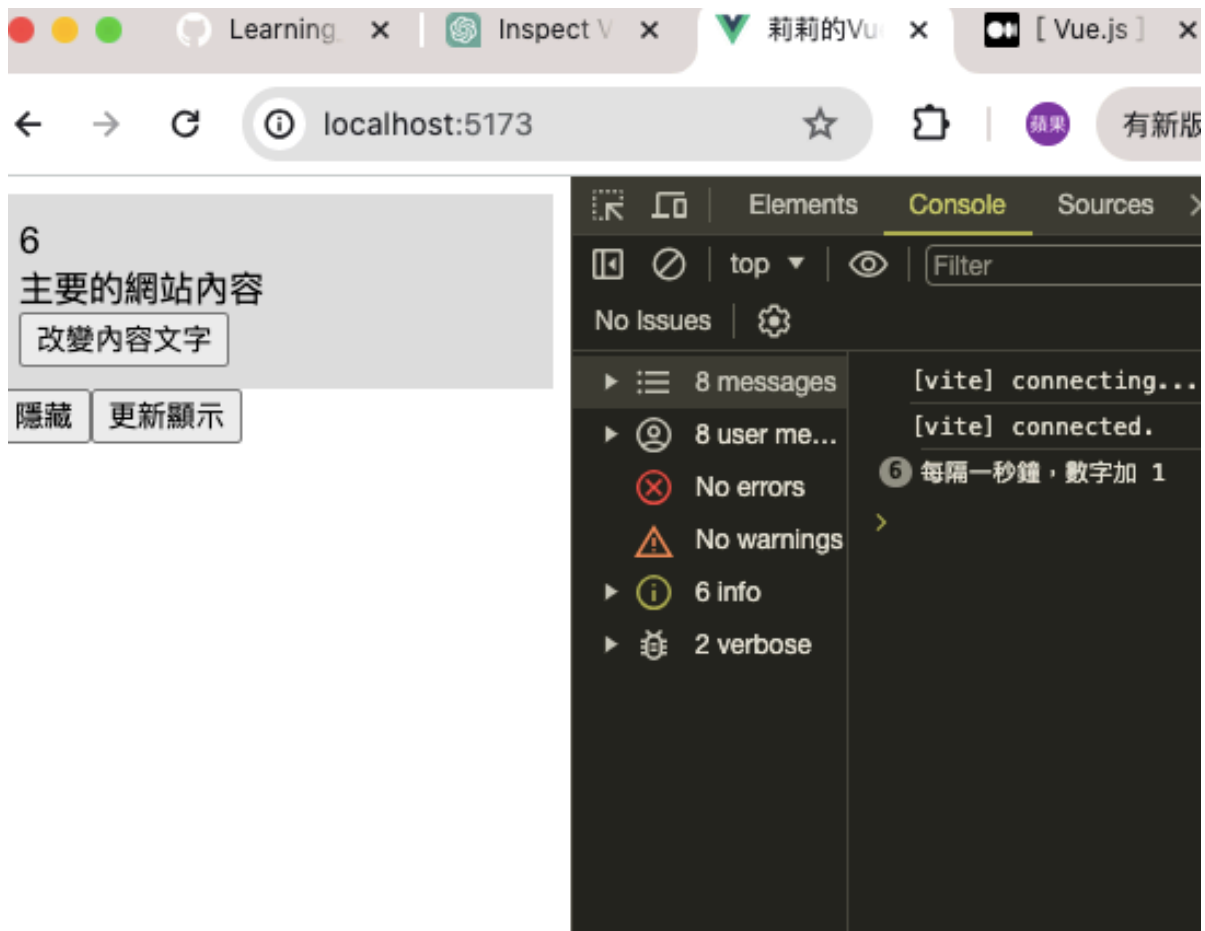
The image shows a web browser window at the top with the address bar displaying "vue-training". Below the browser is a code editor with three tabs: "App.vue", "Main.vue" (which is active), and "Nav.vue". The code editor shows the following code for Main.vue:

```
src > ▼ Main.vue > {} script setup
1
2
3 <script setup>
4   import { onMounted , onUpdated , onUnmounted, ref } from
   'vue';
5   let text = ref("主要的網站內容");
6   let changeText=()=>{
7     text.value="新的網站內容";
8   };
9   // 製作一個 Counter 數數的功能
10  let Counter = ref(0);
11  window.setInterval(
12    ()=>{
13      console.log("每隔一秒鐘，數字加 1");
14      Counter.value = Counter.value+1;
15    },1000);|
16 </script>
17
18
19 <template>
20   <main >
21     <div>{{ text }}</div>
22     <button @click="changeText">改變內容文字</button>
23   </main>
24 </template>
25
26
27 <style scoped>
```



The image shows a code editor window with a dark theme. At the top, there's a search bar containing 'vue-training'. Below it, a tab bar shows three files: 'App.vue', 'Main.vue' (which is active), and 'Nav.vue'. The main editor area displays the content of 'Main.vue'. The file path 'src > Main.vue > {} template' is shown at the top of the editor. The code is written in Vue 3 syntax, featuring a <script setup> block and a <template> block. The script block imports Vue lifecycle hooks and the ref function, initializes a text ref, defines a changeText function, and sets up a counter that increments every 1000ms. The template block contains a main container with a div for the counter, a div for the text, and a button to trigger the text change.

```
src > Main.vue > {} template
1
2
3 <script setup>
4   import { onMounted , onUpdated , onUnmounted, ref } from
   'vue';
5   let text = ref("主要的網站內容");
6   let changeText=()=>{
7     text.value="新的網站內容";
8   };
9   // 製作一個 Counter 數數的功能
10  let Counter = ref(0);
11  window.setInterval(
12    ()=>{
13      console.log("每隔一秒鐘，數字加 1");
14      Counter.value = Counter.value+1;
15    },1000);
16 </script>
17
18
19 <template>
20   <main >
21     <div>{{ Counter }}</div>
22     <div>{{ text }}</div>
23     <button @click="changeText">改變內容文字</button>
24   </main>
25 </template>
26
27
```



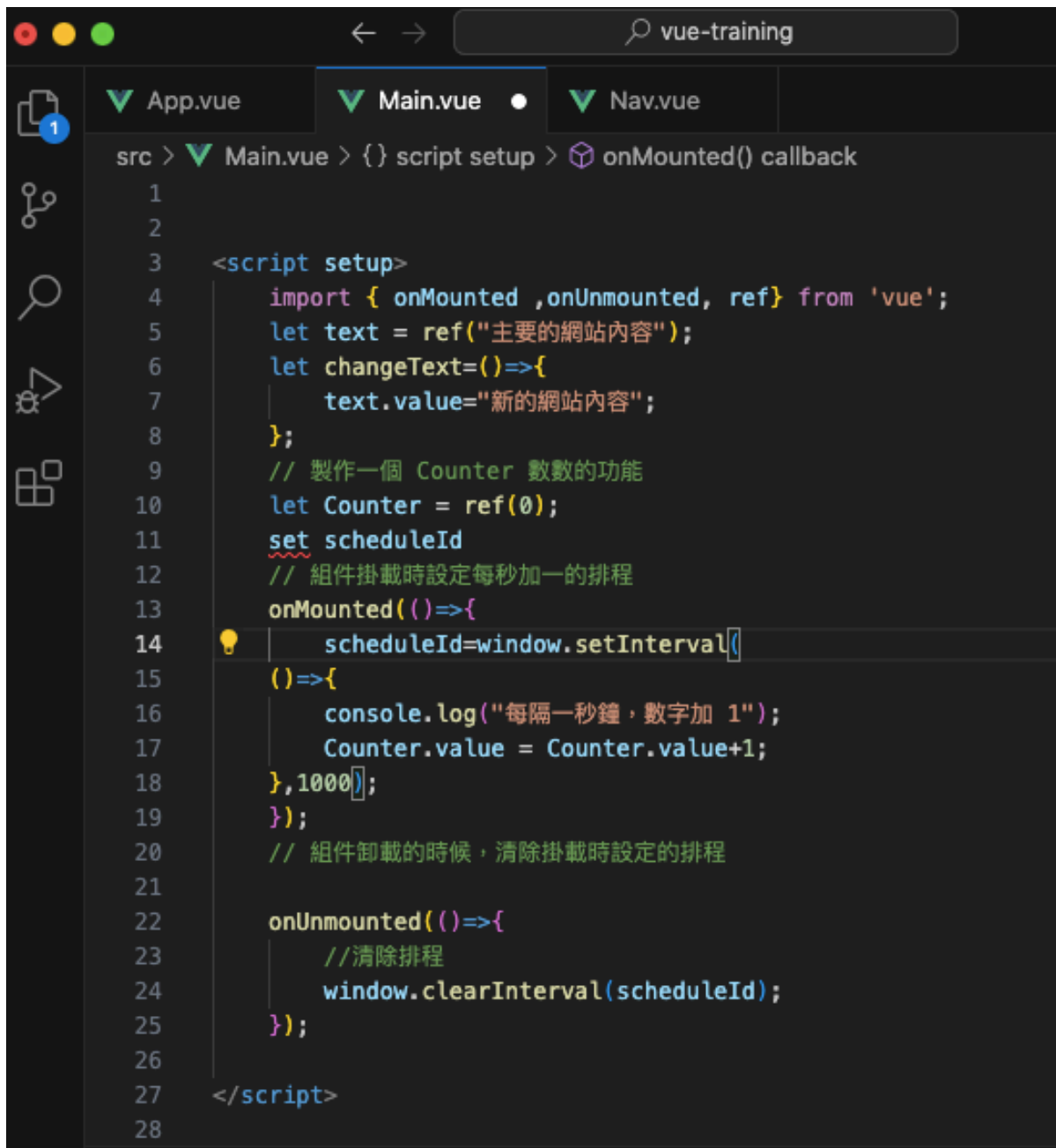
如果按下隱藏，背後的數數排程還跑，再按回重新顯示，後面的排程會以雙倍速度跑，顯然是個隱患：所以我們需要使用生命週期，例用卸載的時機點，把設定的排程清空，組件掛載開始數數，

設定一個排程編號 (scheduleId)，在建立排程時給定編號 `scheduleId=window.setInterval(`

```
()=>{  
  console.log("每隔一秒鐘，數字加 1");  
  Counter.value = Counter.value+1;  
},1000);
```

然後在組件載卸載時清除排程，利用 javascript 內建函式 `window.clearInterval`，給定剛剛的排程編號，來清除排程

`window.clearInterval(scheduleId);`，可以利用生命週期把程式寫的乾淨屬落



The image shows a code editor window with a dark theme. At the top, there's a search bar with the text "vue-training". Below it, a file explorer shows three files: "App.vue", "Main.vue" (selected), and "Nav.vue". The main editor area displays the code for "Main.vue". The breadcrumb navigation at the top of the editor reads "src > Main.vue > {} script setup > onMounted() callback". The code is as follows:

```
1
2
3 <script setup>
4   import { onMounted, onUnmounted, ref } from 'vue';
5   let text = ref("主要的網站內容");
6   let changeText=()=>{
7     text.value="新的網站內容";
8   };
9   // 製作一個 Counter 數數的功能
10  let Counter = ref(0);
11  set scheduleId
12  // 組件掛載時設定每秒加一的排程
13  onMounted(()=>{
14    scheduleId=window.setInterval(
15    ()=>{
16      console.log("每隔一秒鐘，數字加 1");
17      Counter.value = Counter.value+1;
18    },1000);
19  });
20  // 組件卸載的時候，清除掛載時設定的排程
21
22  onUnmounted(()=>{
23    //清除排程
24    window.clearInterval(scheduleId);
25  });
26
27 </script>
28
```