ch3_Vue_Component

註冊組件

1. vue component註冊記得用方法三
2. 組件名稱要寫在html dom裡面，要馬全部小寫要馬kebab case。
3. 每個組件都有自己的setup():setup() 之前寫的methods computed data watch lifestylecycle 都放在setup裡面。
4. components跟watch不用return 其他的都要。
5. 改成composition的原因之一是為了防止this勾錯人。

寫法一 local component (Options API)

const App = Vue.createApp({
data(){},
methods:{},
computed:{},
components:{
'my-component': {
template:
 Hello ,
},
},
})
App.mount('#app')

寫法二　global component　(Options API)
const App = Vue.createApp({ data(){}, methods:{}, computed:{}, }) App.component('counter',{　// counter是組件名稱

template: `
<h1>野原新之助</h1>
<img src =”./XXX/Shinnosuke10.png”>
`,
})
App.mount(‘#app’)

寫法三　(Composition API)

const 組件名稱 = defineComponent({
setup(){},
template:``,
Vue.createApp(組件名稱).mount(
`#app`)

基本範例：組件定義

```javascript
// 定義一個簡單的 Vue 組件
import { defineComponent, ref } from 'vue';

export default defineComponent({
  name: 'MyComponent',
  setup() {
    const count = ref(0);
    const increment = () => count.value++;

    return {
      count,
      increment
    };
  }
});
```

組件的結構：

1. **name**：定義組件的名稱。在 Vue 中，組件應該有唯一的名稱。
2. **setup**：這是 Vue 3 中的新方法，用於組件的邏輯，並且是基於 Composition API 的。所有的狀態、方法和計算都在這裡定義，並返回要暴露給模板的屬性和方法。
3. **ref**：用來創建響應式變數，ref 用來包裝基礎數據類型或引用類型。

動態組件

<component :is="組件名稱"/>
<keep-alive>
<component :is="組件名稱 /變數"　　//變數=組件名時，顯示該組件。
</keep-alive>

1. @click="content='name'" 事件綁定：
• 每個 <button> 標籤都包含一個 @click 事件綁定，當用戶點擊按鈕時，會修改 Vue 實例中的 content 屬性的值。
• 例如，當用戶點擊 "Name" 按鈕時，content 會被設置為 'name'，這樣 Vue 會根據 content 的值來加載對應的組件。

2. <keep-alive> 標籤：
• keep-alive 是 Vue 提供的一個特殊標籤，用來包裹動態組件（例如 <component :is="content"></component>）。
• 它的作用是 保持已加載組件的狀態，當組件從 DOM 中移除後，保持其狀態不被銷毀。這樣可以提高性能，特別是在組件的切換過程中，避免每次切換時都重新渲染組件。

3. <component :is="content"></component>：
• component 是 Vue 中的動態組件的語法。
• :is="content" 表示根據 content 變量的值來決定要加載哪個組件。
• 如果 content 為 'name'，那麼 Vue 就會加載並渲染一個名為 name 的組件。
• 如果 content 為 'date'，則會加載 date 組件，依此類推。

這段代碼展示了 Vue.js 的 動態組件 和 組件緩存 技術，讓你能夠根據用戶的操作動態顯示不同的內容，同時保持已顯示組件的狀態，提升應用性能。

組件的狀態指的是組件內部的 **數據** 和 **狀態變化**，以及它如何影響組件的行為和渲染。具體來說，組件的狀態包括了所有影響組件渲染和行為的資訊，這些資訊通常儲存在組件的 **data** 中，並且可以隨著用戶交互、時間推移或者外部事件的發生而改變。

父子組件

```
const App = {
    components: {
        MyComponent,
        YourComponent,
},
```

```
}

Vue.createApp(App).mount('#app')
```

資料傳遞

- props 和 emit 是 Vue 元件之間的通訊方式，主要用於父子元件之間的資料傳遞。
- Vue.component的命名
  - props:接收資料可以用kebab-case 和camelCase
  - templates:使用資料只能用camelCase
- 傳遞方向
  - props:是從上面傳下來的
  - emit:下層傳到上層的自訂事件
  - props不寫的話要加一個底線當作placeholder。沒有用的話靜態檢查工具會報錯。底線是一個慣例，不影響到後面操作的功能。
- Parent 上層
  - 負責命名屬性
  - 使用該事件
  - 呼叫函數接收資料
- Child　下層
  - 用props接收
  - 負責命名事件
  - this.$emit('事件',值1,...)

有關屬性

寫在props後面的大括號裡面，分別依序放入

1. attrs:attrs 是一個對象，包含了所有傳遞給組件的非 props 屬性。這些屬性會被自動繫結到組件的根元素上，並且可以在組件中使用。

2. slots:slots 是 Vue 組件的插槽功能，可以讓父組件將其內容插入到子組件的指定位置，從而達到更靈活的組件組合。

3. emit:emit 是 Vue 組件用來向父組件傳遞事件的方式。它允許子組件發送自定義事件，父組件可以在其上綁定監聽器來處理這些事件。

功能表格

功能	`attrs`	`slots`	`emit`
定義	包含所有傳遞給組件的非 props 屬性	允許父組件在子組件的預設位置插入內容	允許子組件向父組件發送自定義事件
用途	用於接收並傳遞屬性（如 `class`、`style`）到根元素	用來傳遞父組件的內容到子組件，實現組件內容的插入	子組件向父組件發送事件，以便父組件做相應處理
示例	`<div v-bind="attrs"></div>`	`<slot></slot>`	`this.$emit('event-name', payload)`
可以傳遞的內容	所有傳遞給組件但未作為 `props` 處理的屬性	任意的 HTML 或 Vue 組件	任意的事件名稱和附帶的數據（如數字、對象等）
是否可改變	`attrs` 只是傳遞屬性，不能修改屬性值	插槽內容通常由父組件提供，因此由父組件控制插入的內容	由子組件發送，父組件可以選擇性地處理
例子	`<MyComponent class="example" />`	`<MyComponent><div>Slot Content</div></MyComponent>`	`this.$emit('customEvent', someData)`

總結

- **attrs** 用來傳遞屬性到子組件的根元素，通常用於處理外部傳入的屬性，如 class、style 等。
- **slots** 用來允許父組件動態插入內容到子組件的預設位置，這是一種靈活的組件內容傳遞方式。
- **emit** 用來在子組件中發送自定義事件，父組件可以監聽這些事件並做出相應處理。

這三個功能通常結合使用，讓組件之間的交互和內容傳遞變得更加靈活且強大。

資料傳遞 props&emit V.S. import&export

1. 使用 props 和 emit:props 和 emit 是 Vue 组件之间的通信方式，主要用于父子组件之间的数据传递。它们是 Vue 的核心概念，适用于组件内和组件间的互动。

這兩者的優點是它們是 Vue 本身的機制，非常適合在元件之間進行資料交換，尤其是父子關係元件。

2. 使用 import 和 export:import 和 export 是 JavaScript 的 ES6 模組機制，用於在不同的檔案之間共用程式碼。之間的數據傳遞。

使用場景：

import和export用於在不同的JavaScript模組之間導入和匯出功能，比如函數、類別、物件等。

適用於將業務邏輯、工具函數、常數或配置傳遞到不同的文件中。

何時使用 props 和 emit：

- 組件之間的通信：當需要在父子組件之間傳遞數據時，使用 props 和 emit。
- 事件驅動的通信：當子組件需要向父組件傳遞事件或通知時，使用 emit。 何

何時使用 import 和 export：

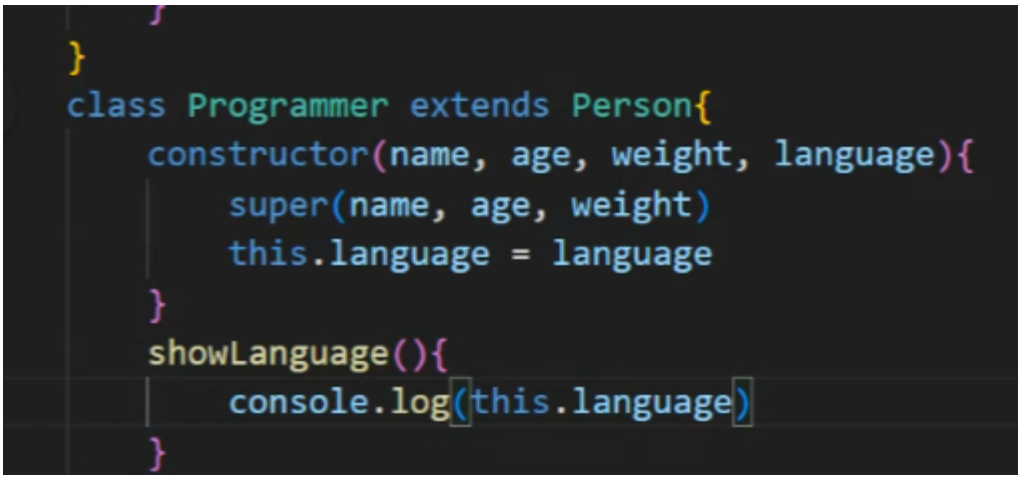
- 跨文件/模塊共享功能：當你需要在多個文件之間共享函數、常量或業務邏輯時，使用 import 和 export。
- 避免重複代碼：通過模塊化代碼來提升重用性和可維護性。

对比与选择：

特性	`props` / `emit`	`import` / `export`
适用场景	组件之间的数据传递，父子组件通信	不同文件或模块之间共享功能、逻辑、工具等
数据传递方式	`props` 用于父组件传递给子组件，`emit` 用于子组件通知父组件	用于模块化代码，共享变量、函数、对象等
通信范围	仅限于 Vue 组件间的通信（父子关系）	适用于跨组件、跨文件共享功能
复杂度	Vue 特有的机制，适合组件间通信，简单易用	JavaScript 原生功能，适合模块化管理
是否响应式	`props` 和 `emit` 是 Vue 响应式的	`import/export` 不涉及响应式

- 如果處理 Vue 元件間的通信，尤其是父子元件之間，使用 **props** 和 **emmit**。
- 如果您在處理跨文件或跨模組的程式碼共享，請使用**import**和**export**來引入和匯出功能。

JS的class



1. js裡面的class 不用寫靜態屬性 就可以用this.屬性

相似之處：

1. 面向物件編程：兩者都支持面向物件編程（OOP），並且都可以透過 class 定義物件的模板，封裝屬性和方法。
2. 建構函數：在 JavaScript 和 Java 中，class 通常都有一個建構函數（constructor），用來初始化物件實例。
3. 繼承：兩者都支持繼承機制，Java 的類別可以繼承父類，JavaScript 也支持透過 extends 關鍵字來繼承父類。

不同之處：

1.語法差異：

- **JavaScript**：class 是 ES6 引入的語法糖，基於原型繼承（prototype inheritance）。JavaScript 的類別本質上是對基於原型的物件建構的封裝。
- **Java**：Java 中的 class 是面向物件編程的核心，類別是經過編譯成字節碼後執行的，並且 Java 嚴格要求類別與繼承結構。Java 類別是完全的面向物件設計，每個物件都必須是某個類別的實例。

2.建構函數與方法的差異：

- **JavaScript**：建構函數使用 constructor，方法定義時沒有顯式的訪問修飾符，所有方法預設是公共的。
- **Java**：方法和建構函數可以有顯式的訪問修飾符（public, private, protected），它支持封裝，方法和屬性的訪問權限可以進行精確控制。

3.類型檢查與靜態類型：

- **JavaScript**：是弱類型語言，變數與物件的類型是動態決定的。
- **Java**：是強類型語言，類別的成員變數和方法的參數類型必須在編譯時確定，並且具有類型安全。

4.實例與物件建立：

- **JavaScript**：類別的實例是動態建立的，透過 new 關鍵字來建立物件。
- **Java**：類別的實例也

5.總結：

- **JavaScript** 的 class 是基於原型的語法糖，它的類別並不是真正的「類」，而是對基於原型的物件繼承機制的封裝。這意味著 JavaScript 的物件是動態的，並且在執行時確定結構。
- **Java** 的 class 是面向物件編程的核心，類別在編譯階段就被確定，並且 Java 是強類型語言，每個類別都有嚴格的結構和型別檢查。

雖然在語法上看起來相似，但它們在內部運作機制和語言特性上有很大的不同。

單一檔案組件

- 定義:將HTML，CSS，JS放在同一個檔案(.Vue檔)，包含三個區塊。
- <template>HTML DOM template</template>
- <script>JS code</script>
- <style>CSS code style</style>
- template 或script一定要有一個，style有沒有沒關係，沒有先後順序。
    - Vue2 的 style 一定要放在最後，Vue3 則沒有此要求。(若在網路上找資料，要小心Vue2的觀點)

Else

1. component命名優先順序:kebabcase >> camelCase >>PascalCase
2. 組件的data一定要是function。因為這樣才可以重複使用。

3. 如果counter組件有好幾個，他們應該獨立各自運作。