

3-22 套用版型建立代辦事項

4-32 v-for 與其使用細節 `Vue.set`

4-33 Computed 與 Watch `Time stamp`

4-34 表單細節 `select`

4-35 v-on 事件修飾符 冒泡捕捉 按鍵修飾符 `keycode` 、 別名修飾 、 包含相應按鍵 (ex:alt+ctrl)
滑鼠修飾符 (滑鼠 左鍵、右鍵、中間鍵)

6-41 x-template component 元件 建立

6-43 props 概念

6-44 props 使用注意事項 `<keep-alive>`

電商網站前置作業 `cli` 前置作業

11 - 85 啟用一個 Vue Cli 並且 引用帶入專屬 API `import axios` `dev.env.js` 環境變數

11 - 86 引用 Bootstrap 套件，並客製化樣式 `node-sass sass-loader` 控管 css

11 - 87 製作登入介面 `router` 初步設定

11 - 89 登入 API 補充說明 (跨域) 解決 CROS cookie

11 - 90 驗證登入及 Vue Router 的配置 導航守衛 - 驗證頁面(登入狀態才可到某頁面)

個人 單元小整理 (85-90)

11 - 91 套用 Bootstrap Dashboard 版型 拆解 dashboard 形成細小元件

11 - 92 製作產品列表 開始製作 page - 產品清單 `Product.vue`

11 - 93 Vue 中運用 Bootstrap 及 jQuery `bootstrap` 外掛 - `Modal` 彈出視窗

11 - 94 產品的新增修改 `input checkbox` 透過 `:true-value` 和 `false-value` 控制
新建產品、編輯產品套用不同 ajax http method

11 - 95 串接上傳檔案 API `webAPI` - `new FormData` 傳送 、 `file upload` 重點

11 - 96 增加使用者體驗 讀取中 `(loading...)` 效果製作 Vue 套件 - `vue-loading-overlay` `fontawesome`

11 - 98 增加使用者體驗 錯誤的訊息回饋 `Event Bus` 外層使用 `$on` 註冊好，然後 內層則使用 `$emit` 觸發它

3-22 套用版型建立代辦事項

底下的 `data` 之中如果有資料是 布林值 並綁在 `input` 的 `checkbox` 之上的話 透過點擊

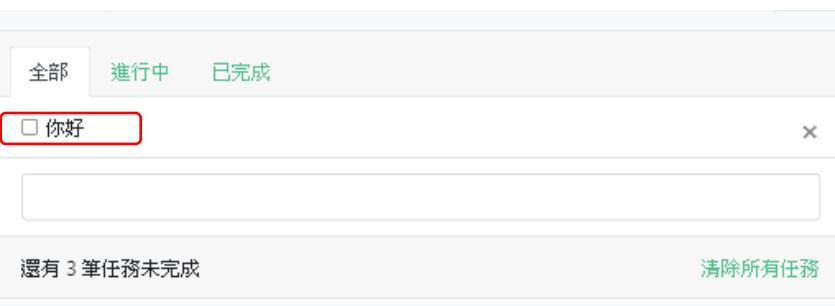
將可即時切換 true & false

```
<div id="app">
  <div class="input-group mb-3">
    <div class="input-group-prepend">
      <span class="input-group-text" id="basic-addon1">待辦事項</span>
    </div>
    <input type="text" class="form-control" placeholder="準備要做的任務"
      v-model="newTodo" @click="addTodo">
    <div class="input-group-append">
      <button class="btn btn-primary" type="button">新增</button>
    </div>
  </div>

  <div class="card text-center">
    <div class="card-header">
      <ul class="nav nav-tabs card-header-tabs">
        <li class="nav-item">
          <a class="nav-link active" href="#">全部</a>
        </li>
```

```
19
20 var app = new Vue({
21   el: '#app',
22   data: {
23     newTodo: '',
24     todos:[// 陳列代辦事項所有內容
25       {
26         id : '2',           // 代辦事項的
27         title : '你好',    // 代辦內容
28         completed : false // 是否完成
29       },
30     ],
31   },
32   methods :{
33     addTodo: function(){
34       }
35     }
36   }
37 });


```



```
    ▼ data
      newTodo: ""
    ▼ todos: Array[1]
      ▼ 0: Object
        completed: false
        id: "2"
        title: "你好"
```

範例：基本 todolist 新增項目進單之中（透過 enter、click 加入）

其中 `` 因綁了 `v-for="item in todos"` 所以可使內容可重複渲染

```
<div id="app">
  <div class="input-group mb-3">
    <div class="input-group-prepend">
      <span class="input-group-text" id="basic-addon1">待辦事項</span>
    </div>
    <input type="text" class="form-control" placeholder="準備要做的任務"
      v-model="newTodo" @keyup.enter="addTodo">
    <div class="input-group-append">
      <button class="btn btn-primary" type="button" @click="addTodo">新增</button>
    </div>
  </div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    newTodo: '', // input 輸入框 (欲鍵入的代辦內容)
    todos:[ ] // 陳列代辦事項 內容的地方
    {
      id : '', // 代辦事項的 title與checkbox 如果要互相綁定要給 id
      title : '', // 代辦內容
      completed : false // 是否完成
    },
  ],
  methods : { // 將input 輸入的 資料加到 代辦清單列表
    /* 取出 輸入的代辦內容，再製作一個 id (使用當下時間)
     | 並透過 Math.floor 將取出的時間轉化為數字*/
    addTodo: function(){
      let value = this.newTodo;
      let timeStamp = Math.floor(Date.now());
      console.log(value,timeStamp);
      this.todos.push({ //將以上兩筆資料、預設completed => todos
        id:timeStamp,
        title:value,
        completed:false
      });
      this.newTodo = ""; // 推完資料後清空
    }
  }
});
```

4-32 v-for 與其使用細節

如果要操作陣列裡的資料內容，不可直接透過索引方式操作（這樣即便 console 看的內容是有被更動，但畫面是不會跟著更動的）。

當有一筆原本不再 data 裡面的資料 我們想要寫進去 則可以使用 Vue.set() 將資料寫進 Vue 的 data 裡面
用法 :Vue.set(要加入的目標陣列，要加進的索引位置，要加入的值)

範例：有一陣列 有自己原始資料，我想透過按下按鈕 來改變陣列 第 0 筆資料，結果 console 看資料有被更改但畫面卻沒有連動更改：

The code editor shows a script block with Vue.js code. The browser screenshot shows a list of three items with an input field next to each item's age. A button labeled '無法運行' (Cannot run) is present. The developer tools' Vue.js tab shows the array state.

```
<h4>不能運作的狀況</h4>
<p>講師說明</p>
<button class="btn btn-outline-primary" @click="cantWork">無法運行</button>
<ul>
  <li v-for="(item, key) in arrayData" :key="item.age">
    {{ key }} - {{ item.name }} {{ item.age }} 歲 <input type="text">
  </li>
</ul>
```

```
16  el: '#app',
17  data: {
18    arrayData: [
19      {name: '小明', age: 16},
20      {name: '漂亮阿姨', age: 24},
21      {name: '杰倫', age: 20},
22    ],
23    methods: {
24      cantWork: function () {
25        this.arrayData[0] = {
26          name: '阿強',
27          age: 99
28        }
29        console.log(this.arrayData)
30      }
31    }
32  });

```

講師說明

無法運行

- 0 - 小明 16 歲
- 1 - 漂亮阿姨 24 歲
- 2 - 杰倫 20 歲

The developer tools' Vue.js tab shows the array state as an observer object with three items. The first item's age has been updated to 99.

```
REVIEW_UNKOWN_SCHEME
▼ (3) [ {...}, {...}, {...}, __ob__: Observer ] ⓘ
  ► 0: {name: "阿強", age: 99}
  ► 1: {__ob__: Observer}
  ► 2: {__ob__: Observer}
  length: 3
  ► __ob__: Observer {value: Array(3), dep: Dep, vmCount: 0}
  ► __proto__: Array
```

改成使用 Vue.set

```
cantWork: function () {
  Vue.set(this.arrayData, 0, {
    name: '阿強',
    age: 99
  })
  console.log(this.arrayData)
}
```

不能運作的狀況

講師說明

無法運行

- 0 - 阿強 99 歲
- 1 - 漂亮阿姨 24 歲
- 2 - 杰倫 20 歲

純數字的迴圈

The developer tools' Vue.js tab shows the array state as an observer object with three items. The first item's age has been updated to 99, and the array's length is now 3.

```
▼ (3) [ {...}, {...}, {...}, __ob__: Observer ] ⓘ
  ► 0:
    age: 99
    name: "阿強"
  ► __ob__: Observer {value: {...}, dep: Dep, vmCount: 0}
  ► get age: f reactiveGetter()
  ► set age: f reactiveSetter(newVal)
  ► get name: f reactiveGetter()
  ► set name: f reactiveSetter(newVal)
  ► __proto__: Object
  ► 1: {__ob__: Observer}
  ► 2: {__ob__: Observer}
  length: 3
  ► __ob__: Observer {value: Array(3), dep: Dep, vmCount: 0}
  ► __proto__: Array
```

4-32 v-for 與其使用細節

v-for 搭配 filter 技巧：

篩選符合陣列內容的資料 (針對 name) :

Filter

請製作過濾資料

```
<h4>Filter</h4>
<p>請製作過濾資料</p>
<input type="text" class="form-control"
    v-model="filterText"
    @keyup.enter="filterData"
>
<ul>
    <li v-for="(item, key) in filterArray" :key="item.age">
        {{ key }} - {{ item.name }} {{ item.age }} 歲
    </li>
</ul>
```

Filter

請製作過濾資料

杰倫

- 0 - 杰倫 20 歲

```
12  var app = new Vue({
13      el: '#app',
14      data: {
15          arrayData: [
16              {name: '小明', age: 16},
17              {name: '漂亮阿姨', age: 24},
18              {name: '杰倫', age: 20}
19          ],
20          filterArray: [],
21          filterText: ''
22      },
23      methods: {
24          filterData: function () {
25              let vm = this;
26              vm.filterArray = vm.arrayData.filter(item=>{
27                  return item.name.match(vm.filterText);
28              });
29          }
30      }
31  })
```

將 this 賦予到 vm 再做使用得原因是 當江振烈使用 filter 跑迴圈時 this 會指向 window 而不是欲想的 new Vue。 Console.log 以下示例

```
27  filterData: function () {
28      let vm = this;
29      console.log(this)
30      vm.filterArray = vm.arrayData.filter(item=>{
31          console.log(this)
32          console.log(vm.filterText, item.name, item.name.match(vm.filterText))
33          return item.name.match(vm.filterText);
34      });
35  },
```

```
▶ Vue {_uid: 0, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: Vue, ...} for.js:29
▶ Vue {_uid: 0, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: Vue, ...} for.js:31
杰倫 小明 null for.js:32
▶ Vue {_uid: 0, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: Vue, ...} for.js:31
杰倫 漂亮阿姨 null for.js:32
▶ Vue {_uid: 0, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: Vue, ...} for.js:31
杰倫 杰倫 ▶ ["杰倫", index: 0, input: "杰倫", groups: undefined] for.js:32
```

透過以上 filter 技巧再延伸一個範例比較：

此範例是一開始資料都在，當鍵入 要搜尋的選項就會直接觸發 Filter 了

Filter

Computed

- 0 - 小明 16 歲
- 1 - 漂亮阿姨 24 歲
- 2 - 杰倫 20 歲

Filter

Methods

- 0 - 小明 16 歲
- 1 - 漂亮阿姨 24 歲
- 2 - 杰倫 20 歲

Filter

Computed

杰倫

- 0 - 杰倫 20 歲

Filter

Methods

杰倫

- 0 - 杰倫 20 歲

Computed 與 Methods 的差異 (用法大致相同，但效能有差)：

```
<h4>Filter</h4>
<p>Computed</p>
<input type="text"
      class="form-control"
      v-model="filterText">
<ul>
  <li v-for="(item, key) in filterByComputed()" :key="item.age">
    {{ key }} - {{ item.name }} {{ item.age }} 歲
  </li>
</ul>

<h4>Filter</h4>
<p>Methods</p>
<input type="text"
      class="form-control"
      v-model="filterText">
<ul>
  <li v-for="(item, key) in filterByMethods()" :key="item.age">
    {{ key }} - {{ item.name }} {{ item.age }} 歲
  </li>
</ul>
```

```
11  var app = new Vue({
12    el: '#app',
13    data: {
14      arrayData: [
15        { name: '小明', age: 16 },
16        { name: '漂亮阿姨', age: 24 },
17        { name: '杰倫', age: 20 }
18      ],
19      filterText: ''
20    },
21    computed: {
22      filterByComputed: function () {
23        let vm = this;
24        return vm.arrayData.filter(item => {
25          return item.name.match(vm.filterText);
26        })
27      }
28    },
29    methods: {
30      filterByMethods: function () {
31        let vm = this;
32        return vm.arrayData.filter(item => {
33          return item.name.match(vm.filterText);
34        });
35      }
36    }
37  },
```

4-33 Computed 與 Watch

Computed:

一般接到後端傳回的資料常會接到一串數字，通常都是針對 Date.now() 處理過的數字。

Ex :

```
> Math.floor(Date.now() / 1000)
< 1594008390
```

 也常被拿來當作 id 做使用

那如果前接到類似情況 要把這段數字變成 日期顯示，在 Vue 應用如下：

```
<p>使用 Computed 來呈現時間格式。</p>
<p> {{formatTime}} </p>
```

使用 Computed 來呈現時間格式。

2020/7/6 12:10:41

```
var app = new Vue({
  el: '#app',
  data: {
    newDate: 0
  },
  mounted : function () {
    this.newDate = Math.floor(Date.now() / 1000);
  },
  computed: {
    formatTime: function () {
      let dates = new Date(this.newDate * 1000); // 把以上除1000 調回
      let year = dates.getFullYear();
      let month = dates.getMonth() + 1;
      let date = dates.getDate();
      let hours = dates.getHours();
      let minutes = dates.getMinutes();
      let seconds = dates.getSeconds();
      return `${year}/${month}/${date} ${hours}:${minutes}:${seconds}`
    }
  }
});
```

Watch :

一般 watch 常用於監聽的概念，當特程式產生特定變化 會執行特定 function

範例：

我要 watch box，當他被執行某 動作時 ，我也會執行某 設定在 watch 裡的 function
使用 trigger 來觸發旋轉 box，並在三秒後改變回來：

Watch

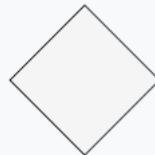
使用 trigger 來觸發旋轉 box、並在三秒後改變回來



Counter

Watch

使用 trigger 來觸發旋轉 box、並在三秒後改變回來



Counter

```
<style>
  .box {
    transition: all .5s;
  }
  .box.rotate {
    transform: rotate(45deg);
  }
</style>

<h4>Watch</h4>
<p>使用 trigger 來觸發旋轉 box、並在三秒後改變回來</p>
<div class="box"
      :class="{'rotate': trigger}">
</div>
<hr>
<button class="btn btn-outline-primary"
        @click="trigger = true">Counter</button>
```

```
19
20 var app = new Vue({
21   el: '#app',
22   data: {
23     trigger: false,
24   },
25   watch: {
26     trigger: function(){
27       let vm = this;
28       setTimeout(function(){
29         vm.trigger = false;
30       },2000)
31     }
32   },
33 });
34 });
35
36
37
```

6-41 x-template - component 元件 建立

以下建立一個全域的 Vue component，他將可使所有 vue app 都可以使用

```
<div id="app">
  <!-- 元件掛載版本 -->
  <table class="table">
    <thead>
    </thead>
    <tbody>
      <!-- <row-component>
          v-for="(item, key) in data"
          :person="item"
          :key="key">
        </row-component>
      <tr is="row-component"
          v-for="(item, key) in data"
          :person="item"
          :key="key">
        </tr>
      </tbody>
    </table>
  <!-- 原始掛載版本 -->
  <table class="table">
    <thead>
    </thead>
    <tbody>
      <tr v-for="(item, key) in data" :item="item" :key="key">
        <td>{{ item.name }}</td>
        <td>{{ item.cash }}</td>
        <td>{{ item.icash }}</td>
      </tr>
    </tbody>
  </table>
</div>
```

因應 tbody 裡面只能直接放 tr 所以以上 component 寫法會出錯，因此官方表示 元件 要使用一個特殊的 is attribute來實現 -->

```
4
5
6
7
8
9
10
11
12  Vue.component('row-component',{
13    data: function (){
14      },
15    props: ['person'],
16    template : ` 
17      <tr>
18        <td>{{ person.name }}</td>
19        <td>{{ person.cash }}</td>
20        <td>{{ person.icash }}</td>
21      </tr>
22    `
23  })
24 }
25
26 var app = new Vue({
27   el: '#app',
28   data: {
29     data: [
30       {name: '小明',cash: 100,icash: 500,},
31       {name: '杰倫',cash: 10000,icash: 5000,},
32       {name: '漂亮阿姨',cash: 500,icash: 500,},
33       {name: '老媽',cash: 10000,icash: 100,},
34     ]
35   }
36 });
37 }
```

局部 元件：讓我這個 row-component 元件只可以在 該 app 內做重複使用：

只要把 props & template 提出來令賦予到一個變數上，並在 app 裡 增添 components 屬性並綁上即可。

```
Vue.component('row-component',{
  data: function (){
    },
    props: ['person'],
    template : ` 
      <tr>
        <td>{{ person.name }}</td>
        <td>{{ person.cash }}</td>
        <td>{{ person.icash }}</td>
      </tr>
    `
  })
var app = new Vue({
  el: '#app',
  data: {
    data: [
      {name: '小明',cash: 100,icash: 500,},
      {name: '杰倫',cash: 10000,icash: 5000,},
      {name: '漂亮阿姨',cash: 500,icash: 500,},
      {name: '老媽',cash: 10000,icash: 100,},
    ]
  }
});
```

>>>

```
let childComponent = {
  props: ['person'],
  template : ` 
    <tr>
      <td>{{ person.name }}</td>
      <td>{{ person.cash }}</td>
      <td>{{ person.icash }}</td>
    </tr>
  `
}

var app = new Vue({
  el: '#app',
  data: {
    data: [
      {name: '小明',cash: 100,icash: 500,},
      {name: '杰倫',cash: 10000,icash: 5000,},
      {name: '漂亮阿姨',cash: 500,icash: 500,},
      {name: '老媽',cash: 10000,icash: 100,},
    ],
    components : {
      'row-component' : childComponent
    }
  });
});
```

6-42 元件 data 注意事項 & 多元件優勢比較

<H2>重複元件的 data 是獨立的 不互相影響</H2>

```
<div id="app1">
  <btntest></btntest>
  <btntest></btntest>
  <btntest></btntest>
</div>
```

```
<hr>
```

<H2>自行重複的 DIV data不是獨立的 會互相影響</H2>

```
<div id="app2">
  <div class="count">
    <div>你已經點了 {{count}} 下</div>
    <button @click="count += 1">click me</button>
  </div>
  <div class="count">
    <div>你已經點了 {{count}} 下</div>
    <button @click="count += 1">click me</button>
  </div>
  <div class="count">
    <div>你已經點了 {{count}} 下</div>
    <button @click="count += 1">click me</button>
  </div>
</div>
```

重複元件的 data 是獨立的 不互相影響

```
你已經點了 4 下 click me
你已經點了 1 下 click me
你已經點了 3 下 click me
```

自行重複的 DIV data不是獨立的 會互相影響

```
你已經點了2下 click me
你已經點了2下 click me
你已經點了2下 click me
```

```
5   Vue.component('btntest', {
6     data() {
7       return { count: 0 }
8     },
9     template: `
10    <div class="count">
11      <div>你已經點了 {{count}} 下</div>
12      <button @click="count += 1">click me</button>
13    </div>
14  })
15
16
17  let vm1 = new Vue({
18    el: '#app1',
19  })
20
21
22  let vm2 = new Vue({
23    el: '#app2',
24    data: {
25      count: 0,
26    }
27  })
28
29
```

6-43 props 概念

正確的 props 運用 是以下 第二段 從外部傳入 (動態傳入)

Props 定義的用意 主要在 元件使用時 跟元件 告知：我要傳進一個 url 喔 那是有外部傳進去的 要記得對應

```
<!-- 當你使用DOM 中的模板時，camelCase (駝峰命名法) 的  
| | prop 名需要使用其等價的kebab-case (短橫線分隔命名) 命名 -->  
<div id="app">  
<h2>從外部傳入(透過value(字串)) - 靜態傳遞</h2>  
<photo img-url="https://images.unsplash.com/photo-152220453834..."  
<!-- 類似 v-bind 概念 在attribute 前面加 " : " --><br><br><br>  
<h2>從外部傳入(透過key)：資料可能隨意變動-動態傳遞</h2>  
<photo :img-url="url"></photo>  
</div>
```

```
5  Vue.component('photo', {  
6    // 同學請依據課程內容，自行填寫 Props 的寫法  
7    props: ['imgUrl'],  
8    template:  
9      <div>  
10          
11        <p>風景照</p>  
12      </div>  
13    </template>  
14  })  
15  
16  var app = new Vue({  
17    el: '#app',  
18    data: {  
19      url: 'https://images.unsplash.com/photo-152220453834...'  
20    }  
21  });
```

從外部傳入(透過value(字串)) - 靜態傳遞



風景照

從外部傳入(透過key)：資料可能隨意變動-動態傳遞



風景照

電商網站前置作業

cli 2

npm install -g @vue/cli 下載 cli

vue list 查看你要使用何種專案

vue init <template-name> <project-name> 初始化專案 ex : vue init webpack vue-test

npm i --save axios vue-axios 安裝 axios

11 – 85 啟用一個 Vue Cli 並且 引用帶入專屬 API

測試撈取 申請好的 api 資料 -

Vue-axios 官方提供 :

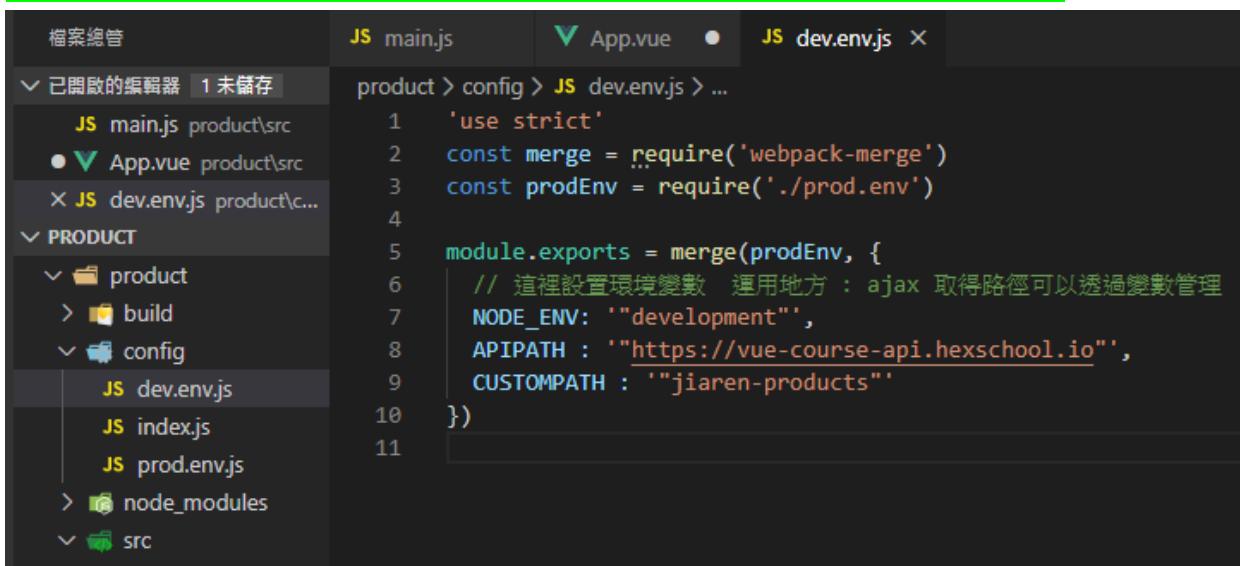
在進入點程式

```
import axios from 'axios';
import VueAxios from 'vue-axios';
Vue.use(VueAxios, axios);
```

抓取 api 測試 :

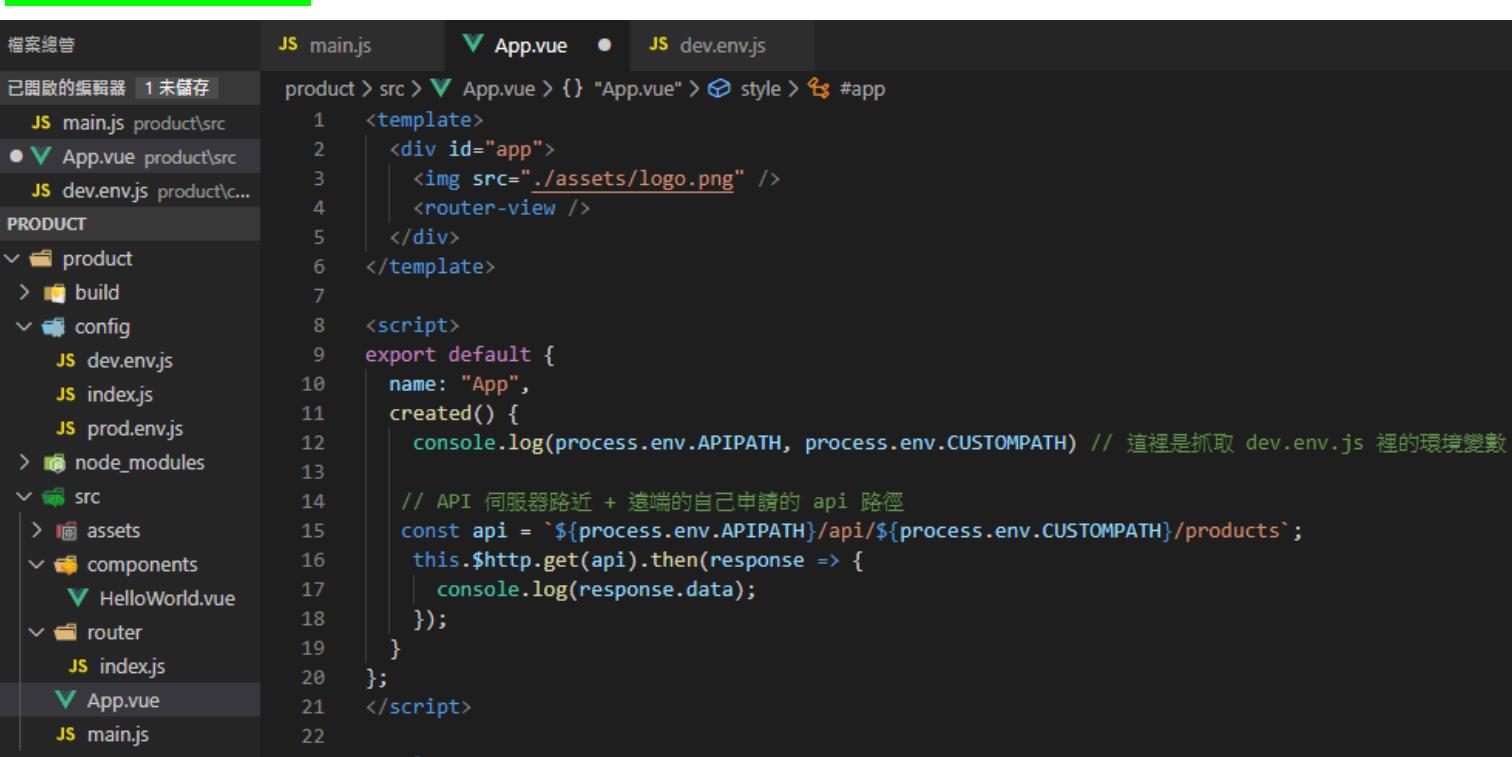
```
this.$http.get(api).then((response) => {
  console.log(response.data)
})
```

首先 在 config > dev.env.js 建立環境變數這裡把要撈取的 路徑用變數管理 :



```
product > config > JS dev.env.js > ...
1  'use strict'
2  const merge = require('webpack-merge')
3  const prodEnv = require('./prod.env')
4
5  module.exports = merge(prodEnv, {
6    // 這裡設置環境變數 運用地方 : ajax 取得路徑可以透過變數管理
7    NODE_ENV: '"development"',
8    APIPATH: '"https://vue-course-api.hexschool.io"',
9    CUSTOMPATH: '"jiaren-products"'
10 })
11
```

在 APP.vue 做 測試 :



```
product > src > App.vue > {} "App.vue" > style > #app
1  <template>
2    <div id="app">
3      
4      <router-view />
5    </div>
6  </template>
7
8  <script>
9    export default {
10      name: "App",
11      created() {
12        console.log(process.env.APIPATH, process.env.CUSTOMPATH) // 這裡是抓取 dev.env.js 裡的環境變數
13
14        // API 伺服器路徑 + 遠端的自己申請的 api 路徑
15        const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/products`;
16        this.$http.get(api).then(response => {
17          console.log(response.data);
18        });
19      }
20    };
21  </script>
22
23  <style>
```

Console.log 回傳的 ajax 結果：



Welcome to Your Vue.js App

Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#)
[Docs for This Template](#)

Ecosystem

[vue-router](#) [vuex](#) [vue-loader](#) [awesome-vue](#)

```
[HTTP] WAITING FOR UPDATE SIGNAL FROM WDS...
https://vue-course-api.hexschool.io jiaren-products
⚠ DevTools failed to load SourceMap: Could not load content for chrome-extension://gighmmpliobk1fepjocnamgkkbiglidom/include_postload.js.map: HTTP error: status code 404, net::ERR_UNKNOWN_URL_SCHEME
vue-devtools Detected Vue v2.6.11 backend.js:2237
App.vue?26cd:12
App.vue?26cd:15
▼ {success: true, products: Array(1), pagination: {...}, messages: Array(0)} ⓘ
  ► messages: []
  ► pagination: {total_pages: 1, current_page: 1, has_pre: false, has_next: false, category...}
  ▼ products: Array(1)
    ▼ 0:
      category: "Nikon D750"
      id: "-M8r6sxhBRTA3qpZs2sV"
      image: "https://images.unsplash.com/photo-1516550135131-fe3dcb0bedc7?ixlib=rb-0.3.5&ixid=MnwxMjA3fDB9MHxwaG90by1wYWdlfHx8fGVufDB8fHx8"
      num: 1
      origin_price: "35000"
      price: "33000"
      title: "測試的產品"
      unit: "台"
      ► __proto__: Object
      length: 1
      ► __proto__: Array(0)
      success: true
      ► __proto__: Object
```

11 - 86. 引用 Bootstrap 套件，並客製化樣式

Install :

```
npm i bootstrap --save
```

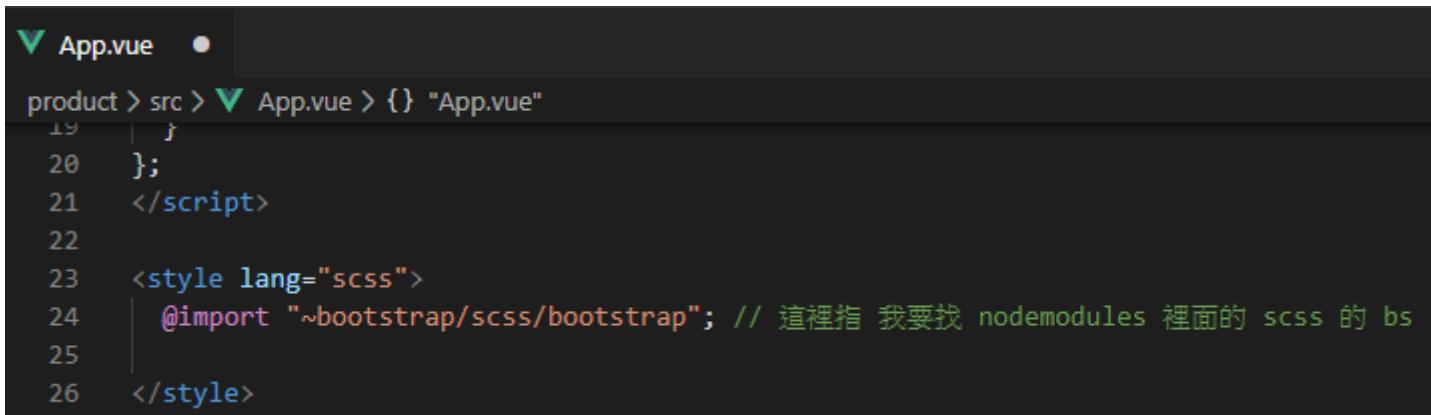
安裝後會跳出 你還缺少 jq 和 peer 等 bootstrap 的其他套件要安裝

```
PS D:\CodePractice\product\product> npm i bootstrap --save
npm WARN ajv-keywords@3.5.1 requires a peer of ajv@^6.9.1 but none is installed. You may need to install it with --save-dev.
npm WARN bootstrap@4.5.0 requires a peer of jquery@1.9.1 - 3 but none is installed. You may need to install it with --save-dev.
npm WARN bootstrap@4.5.0 requires a peer of popper.js@^1.16.0 but none is installed. You may need to install it with --save-dev.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.3 (node_modules\fsevents)
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted {"os": "darwin", "arch": "any"} (current: {"os": "win32", "arch": "x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\fsevents)
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os": "darwin", "arch": "any"} (current: {"os": "win32", "arch": "x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\fsevents)
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os": "darwin", "arch": "any"} (current: {"os": "win32", "arch": "x64"})
```

接著 我打算使用 scss 做撰寫 所以要在下載安裝

```
npm i node-sass sass-loader@7.1.0 --save      >>>練習期間的 sass loader 當下版本有問題
```

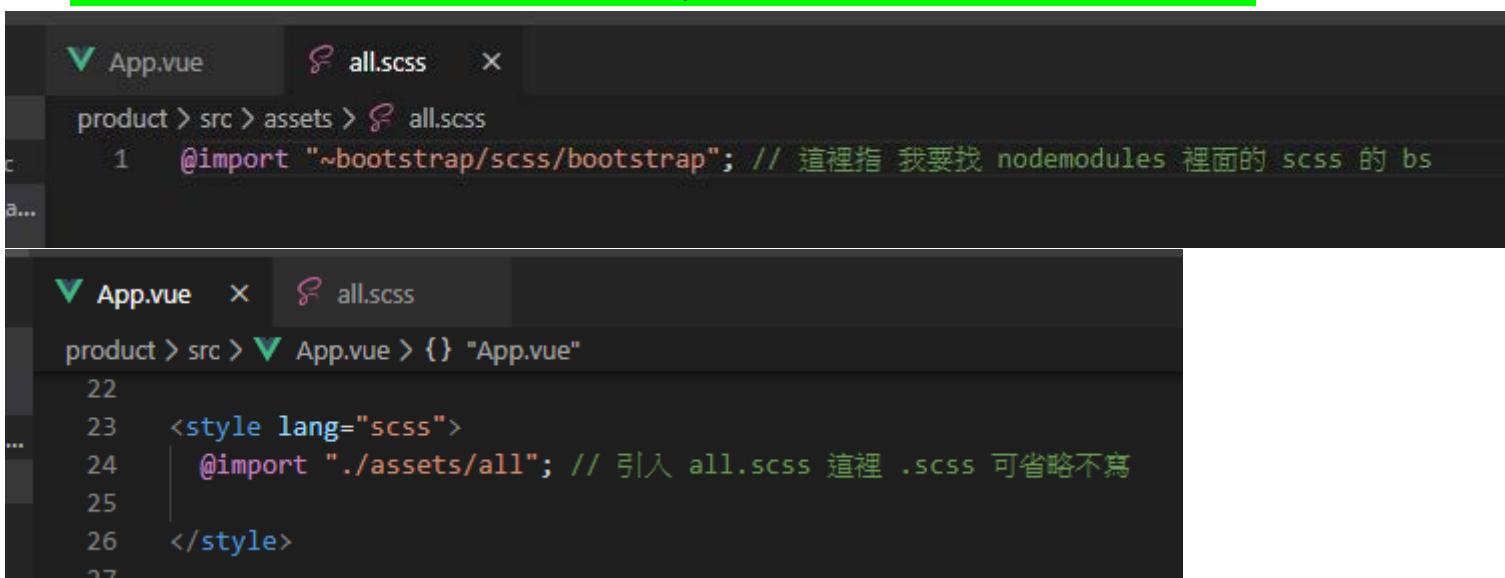
所以這邊針對板本下載。



```
App.vue
```

```
product > src > App.vue > {} "App.vue"
  19  ...
  20  };
  21  </script>
  22
  23  <style lang="scss">
  24    @import "~bootstrap/scss/bootstrap"; // 這裡指 我要找 nodemodules 裡面的 scss 的 bs
  25
  26  </style>
```

此時如果我要集中管理 scss 可以將以上 import 移動到 asset 裡面 在新建一個 scss



```
App.vue
```

```
product > src > assets > all.scss
  1  @import "~bootstrap/scss/bootstrap"; // 這裡指 我要找 nodemodules 裡面的 scss 的 bs
```

```
App.vue
```

```
product > src > App.vue > {} "App.vue"
  22
  23  <style lang="scss">
  24    @import "./assets/all"; // 引入 all.scss 這裡 .scss 可省略不寫
  25
  26  </style>
  27
```

接著找到 node module 裡面 bs > scss > _variables.scss

把她另存新檔 在 asset > helpers > _variables.scss

```
product > src > assets > helpers > _variables.scss > [?] $grays
1 // Variables
2 //
3 // Variables should follow the `$component-state-property-size` convention
4 // consistent naming. Ex: $nav-link-disabled-color and $modal-primary-color
5
6 // Color system
7
8 $white: #fff !default;
9 $gray-100: #f8f9fa !default;
10 $gray-200: #e9ecef !default;
11 $gray-300: #dee2e6 !default;
12 $gray-400: #ced4da !default;
13 $gray-500: #adb5bd !default;
14 $gray-600: #6c757d !default;
15 $gray-700: #495057 !default;
16 $gray-800: #343a40 !default;
17 $gray-900: #212529 !default;
18 $black: #000 !default;
19
20 $grays: () !default;
21 // stylelint-disable-next-line scss/dollar-variable-default
22 $grays: map-merge(
23 | [
```

然後設定 all.scss 添加新路徑

```
product > src > assets > all.scss
1 @import "~bootstrap/scss/bootstrap"; // 這裡指 我要找 nodemodules 裡面的 scss 的 bs
2 @import "~bootstrap/scss/functions"; // 這裡載入 bs 套用變數的方法 ，有這些方法 bs 變數才能正確使用
3 @import "./helpers/_variables"; // 這裡載入 bs 我們自定義的變數
4
```

接著就可以試著更改 bs 的預設樣式與顏色了：透過 _variables 做修改

11 - 87. 製作登入介面

首先 在專案裡增添 components > pages > login 頁面

內容先 增添 bootstrap 官方 的 login 範例

view-source:<https://getbootstrap.com/docs/4.5/examples/sign-in/>

The screenshot shows a code editor with the following file structure on the left:

- 已開啟的編輯器
 - login.vue
 - App.vue
 - index.js
- PRODUCT
 - product
 - build
 - config
 - node_modules
 - src
 - assets
 - components
 - pages
 - login.vue
 - HelloWorld.vue
 - router
 - index.js
 - App.vue
 - main.js
 - static
- .babelrc
- .editorconfig
- .gitignore
- .postcssrc.js
- index.html
- package-lock.json
- package.json
- README.md

```
product > src > components > pages > login.vue > {} "login.vue"
1   <template>
2     <div>
3       <form class="form-signin">
4
5         <h1 class="h3 mb-3 font-weight-normal">Please sign in</h1>
6         <label for="inputEmail" class="sr-only">Email address</label>
7         <input
8           type="email"
9             id="inputEmail"
10            class="form-control"
11            placeholder="Email address"
12            required
13            autofocus
14          />
15         <label for="inputPassword" class="sr-only">Password</label>
16         <input
17           type="password"
18             id="inputPassword"
19            class="form-control"
20            placeholder="Password"
21            required
22          />
23         <div class="checkbox mb-3">
24           <label>
25             <input type="checkbox" value="remember-me" /> Remember me
26           </label>
27         </div>
28         <button class="btn btn-lg btn-primary btn-block" type="submit">Sign in</button>
29         <p class="mt-5 mb-3 text-muted">&copy; 2017-2020</p>
30       </form>
31     </div>
32   </template>
```

然後到 router 做 路由設定

The screenshot shows a code editor with the following file structure on the left:

- 已開啟的編輯器
 - login.vue
 - App.vue
 - index.js
- PRODUCT
 - product
 - build
 - config
 - node_modules
 - src
 - assets
 - components
 - pages
 - HelloWorld.vue
 - router
 - index.js
 - App.vue
 - main.js
 - static
- .babelrc

```
product > src > router > JS index.js > ...
1 import Vue from 'vue'
2 import Router from 'vue-router'
3 import HelloWorld from '@/components/HelloWorld'
4
5 Vue.use(Router)
6
7 export default new Router({
8   routes: [
9     {
10       path: '/',
11       name: 'HelloWorld',
12       component: HelloWorld
13     },
14     {
15       path: '/login', // 虛擬 path 一般建議打小寫
16       name: 'Login',
17       component : () => import('@/components/pages/login'), // 載入元件
18     }
19   ]
20 })
21 }
```

在來到 <https://getbootstrap.com/docs/4.5/examples/sign-in/signin.css>

把剛剛 bs login 範例的 css 紿複製起來加進 login style 裡面

The screenshot shows a code editor with three tabs: login.vue, App.vue, and index.js. The login.vue tab is active, displaying its source code. The code includes a CSS block with the 'scoped' attribute, which restricts the styles to this specific component. The styles define a full-height body with flexbox properties, centered alignment, and a light gray background. It also defines a form container with a maximum width of 330px, padding, and margin auto. The .checkbox class is defined with a font-weight of 400. The .form-control class is defined with a relative position, border-box box-sizing, and an auto height.

```
product > src > components > pages > login.vue > {} "login.vue" > ↗
43   </script>
44
45   <!-- Add "scoped" 讓此 css 只在此檔案下有作用 -->
46   <style scoped>
47     html,
48     body {
49       height: 100%;
50     }
51
52     body {
53       display: -ms-flexbox;
54       display: flex;
55       -ms-flex-align: center;
56       align-items: center;
57       padding-top: 40px;
58       padding-bottom: 40px;
59       background-color: #f5f5f5;
60     }
61
62     .form-signin {
63       width: 100%;
64       max-width: 330px;
65       padding: 15px;
66       margin: auto;
67     }
68     .form-signin .checkbox {
69       font-weight: 400;
70     }
71     .form-signin .form-control {
72       position: relative;
73       box-sizing: border-box;
74       height: auto;
75     }
```

此時 套上 bs 的 login 頁面就大概完成版型了 :

Please sign in

Remember me

Sign in

© 2017-2020

接著到 <https://github.com/hexschool/vue-course-api-wiki/wiki/%E7%99%BB%E5%85%A5%E5%8F%8A%E9%A9%97%E8%AD%89>

閱讀 API 規則 寫法

目前這個做法在 Chrome 更新後的 同源政策 有出現 cookie 的問題 所以可能無法順利登入
改善方法在下一個章節

登入

目前 /signin 暫不使用，課程「登入 API 補充說明 (跨域)」會有說明如何運用 /admin/signin 正確登入喔。

```
[API]: /signin
[方法]: post
[參數]:
{
  "username": "hexscholl@test.com",
  "password": "zzxxccvv"
}
[成功回應]:
{
  "success": true,
  "message": "登入成功",
  "uid": "XX4VbV871RRBXKhZKT7YX6zhsu02"
}
[失敗回應]:
{
  "success": false,
  "message": "登入失敗"
}
```

定義資料結構 並撰寫 登入事件，如果登入成功 就 轉址到首頁:

```
login.vue x App.vue JS index.js
product > src > components > pages > login.vue > {} "login.vue" > script > methods
1  <template>
2    <div>
3      <form class="form-signin" @submit.prevent="signin" > <!-- 做登入事件 -->
4        <h1 class="h3 mb-3 font-weight-normal">Please sign in</h1>
5        <label for="inputEmail" class="sr-only">Email address</label>
6        <input type="email" id="inputEmail" class="form-control" placeholder="Email address" required autofocus
7          v-model = "user.username" /> <!-- 繩上 username -->
8        <label for="inputPassword" class="sr-only">Password</label>
9        <input type="password" id="inputPassword" class="form-control" placeholder="Password" required
10          v-model="user.password" /> <!-- 繩上 password -->
11        <div class="checkbox mb-3">
12          <label>
13            <input type="checkbox" value="remember-me" /> Remember me
14          </label>
15        </div>
16        <button class="btn btn-lg btn-primary btn-block" type="submit">Sign in</button>
17        <p class="mt-5 mb-3 text-muted">&copy; 2017-2020</p>
18      </form>
19    </div>
20  </template>
21
22  <script>
23  export default {
24    name: "HelloWorld",
25    data() {
26      return {
27        user :{
28          username : '',
29          password : ''
30        }
31      };
32    },
33    methods : []
34    signin(){
35
36      // 依據 API 文件指出 以下只要傳入 /signin 即可 : 伺服器path +  signin
37      const api = `${process.env.APIPATH}/signin`;
38      const vm = this;
39      this.$http.post(api , vm.user).then(response => { // 然後 使用 post 傳入用戶資料  vm.user
40        console.log(response.data);
41        if (response.data.success){ // 這裡寫一個判別式 : 如果登入成功 就將路徑 轉到我們的首頁
42          vm.$router.push('/')
43        }
44      });
45    }
46  }
47  </script>
```

然後到臨時的首頁 HelloWorld 撰寫 登出 事件 (登出後又會轉址回到 登入頁面)

登出

```
[API]: /logout  
[方法]: post  
[成功回應]:  
{  
    "success": true,  
    "message": "已登出"  
}
```

login.vue

HelloWorld.vue

App.vue

index.js

product > src > components > HelloWorld.vue > {} "HelloWorld.vue" > style > h1

```
1   <template>  
2     <div class="hello">  
3       <a href="#" @click.prevent="sigout">登出</a>  
4     </div>  
5   </template>
```

```
7   <script>  
8     export default {  
9       name: 'HelloWorld',  
10      data () {  
11        return {  
12          msg: 'Welcome to Your Vue.js App'  
13        }  
14      },  
15      methods:{  
16        sigout(){  
17          // 依據 API 文件指出 此api 址需傳入 api 不須傳入其他資料 即可登出  
18          const api = `${process.env.APIPATH}/logout`;  
19          const vm = this;  
20          this.$http.post(api).then(response => { // 然後 使用 post 傳入用戶資料 vm.user  
21            console.log(response.data);  
22            if (response.data.success){ // 這裡寫一個判別式：如果登出成功 就將路徑 轉到我們登入頁面  
23              vm.$router.push('/login')  
24            }  
25          });  
26        }  
27      }  
28    }  
29  </script>
```

11-89 登入 API 補充說明 (跨域)

此章節 改善同源政策 Cookie 問題：

上一章節主要是缺少 將 cookie 正確的存起來 放置於 伺服器裡的 動作，以下做法便是達成此 動作 將 cookie 正確的存放在 Vue 的伺服器裡

<https://github.com/hexschool/vue-course-api-wiki/wiki/%E7%99%BB%E5%85%A5%E5%8F%8A%E9%A9%97%E8%AD%89>

增添一個 admin

登入 API 做法

前端 axios 請求附帶 Cookies 設定

前端要把 cookie 的開關打開

```
axios.defaults.withCredentials = true;
```

```
[API]: /admin/signin
[方法]: post
[參數]:
{
  "username": "hexscholl@test.com",
  "password": "zzxxccvv"
}
[成功回應]:
{
  "success": true,
  "message": "登入成功",
  "uid": "XX4VbV871RRBXKhZKT7YX6zhsu02"
}
[失敗回應]:
{
  "success": false,
  "message": "登入失敗"
```

The screenshot shows a code editor interface with the following details:

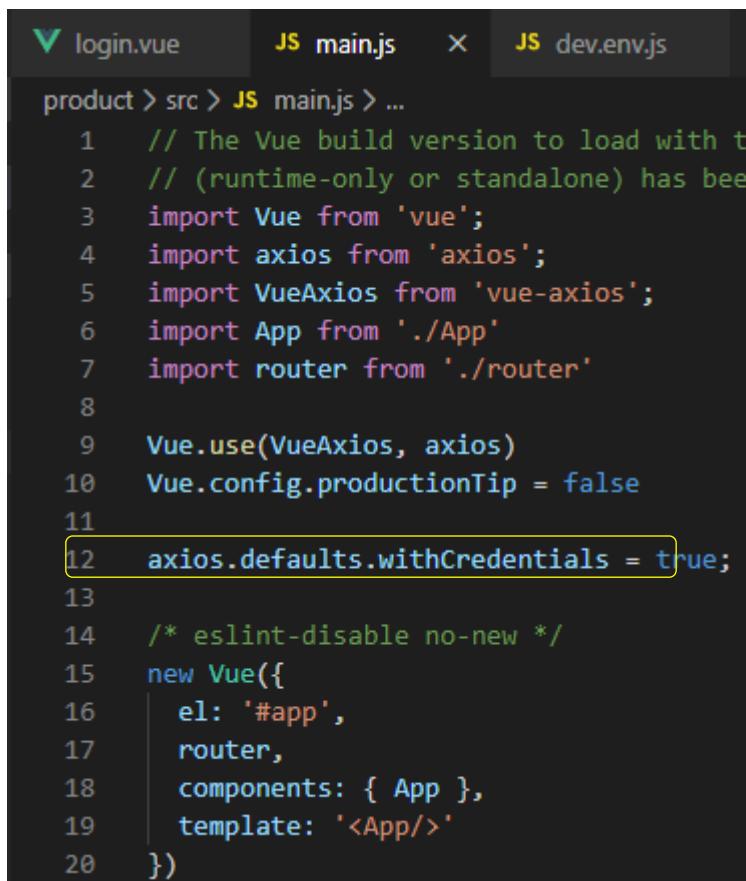
- Folder Structure:** The left sidebar shows a tree view of the project structure:
 - 已開啟的編輯器: login.vue
 - PRODUCT: product, config
 - src: assets, components, pages, router, static
 - node_modules
- File Content:** The main editor area shows the code for `login.vue`.

```
<script>
export default {
  name: "HelloWorld",
  data() {
    return {
      user :{
        username : '',
        password : ''
      }
    };
  },
  methods : {
    signin(){
      // 依據 API 文件指出 以下只要傳入 /signin 即可： 伺服器path + signin
      const api = `${process.env.APIPATH}/admin/signin`;
      const vm = this;
      this.$http.post(api , vm.user).then(response => { // 然後 使用 post 傳入用戶資料 vm.user
        console.log(response.data);
        if (response.data.success){ // 這裡寫一個判別式：如果登入成功 就將路徑 轉到我們的首頁
          vm.$router.push('/')
        }
      });
    }
  }
};</script>
```

在 axios 文件中 表示 axios CROS 跨域 達成 存 Cookie 動作需要 帶上 credentials 參數 將他寫入 Vue 應用程式裡 才會動作

<https://github.com/axios/axios>

```
// `timeout` specifies the number of milliseconds before the request times out.  
// If the request takes longer than `timeout`, the request will be aborted.  
timeout: 1000, // default is `0` (no timeout)  
  
// `withCredentials` indicates whether or not cross-site Access-Control requests  
// should be made using credentials  
withCredentials: false, // default  
  
// `adapter` allows custom handling of requests which makes testing easier.  
// Return a promise and supply a valid response (see lib/adapters/README.md).  
adapter: function (config) {  
  /* ... */
```



```
product > src > JS main.js > ...  
1  // The Vue build version to load with t  
2  // (runtime-only or standalone) has been  
3  import Vue from 'vue';  
4  import axios from 'axios';  
5  import VueAxios from 'vue-axios';  
6  import App from './App'  
7  import router from './router'  
8  
9  Vue.use(VueAxios, axios)  
10 Vue.config.productionTip = false  
11  
12  axios.defaults.withCredentials = true;  
13  
14  /* eslint-disable no-new */  
15  new Vue({  
16    el: '#app',  
17    router,  
18    components: { App },  
19    template: '<App/>'  
20  })
```

詳細後端運作原理 可以看

<https://www.udemy.com/course/vue-hexschool/learn/lecture/11560492#questions/11390656>

以上有個 bug 是目前 還未做 router 上的判別，所以使用者其實只要直接在 登入頁面或是其他頁面 輸入網址一樣可以導航到 登出 使用頁面去（指定頁面）。

下章節將改善此問題

11 – 90 驗證登入及 Vue Router 的配置

Router 目前還未判別我們在那些頁面是需要登入 那些頁面是不需要登入的

Vue Router 導航守衛 方法 ↓

<https://router.vuejs.org/zh/guide/advanced/navigation-guards.html>

使用 router.beforeEach :

他會在切換頁面時觸發，並帶有 三個參數 (即將到的頁面，從哪頁面來，到達下個頁面)

全局前置守衛

你可以使用 router.beforeEach 註冊一個全局前置守衛：

```
const router = new VueRouter({ ... })  
  
router.beforeEach((to, from, next) => {  
  // ...  
})
```

js

當一個導航觸發時，全局前置守衛按照創建順序調用。守衛是異步解析執行，此時導航在所有守衛 resolve 完之前一直處於等待中。

每個守衛方法接收三個參數：

- `to: Route` :即將要進入的目標路由對象

因此如果指定頁面是需要登入狀態時就可以使用 `to` 參數 當確認他沒有問題

或是 他不需要登入 時我們就可以放行他~

以下是 路由信息 (Vue 官網範例) meta requireAuth 表示此頁面是需要驗證的 (true or false 是個判斷基準)

所以路由就會通過撰寫的 method 來判別



ads via Carbon

路由元信息

定義路由的時候可以配置 `meta` 字段：

```
const router = new VueRouter({  
  routes: [  
    {  
      path: '/foo',  
      component: Foo,  
      children: [  
        {  
          path: 'bar',  
          component: Bar,  
          // a meta field  
          meta: { requiresAuth: true }  
        }  
      ]  
    }  
  ]  
})
```

js

安裝

介紹

基礎

起步

動態路由匹配

嵌套路由

編程式導航

命名路由

命名視圖

重定向和別名

路由組件傳參

HTML5 History 模式

進階

導航守衛

路由元信息

那麼如何訪問這個 `meta` 字段呢？

以下 是六角 API 可以檢查 用戶是否還在登入 狀態的 API

<https://github.com/hexschool/vue-course-api-wiki/wiki/%E7%99%BB%E5%85%A5%E5%8F%8A%E9%A9%97%E8%AD%89>

檢查用戶是否仍持續登入

```
[API]: /api/user/check  
[方法]: post  
[說明]: 檢查用戶是否仍持續登入  
[成功回應]:  
{  
    "success": true  
}  
[失敗回應]: 如用戶已登出狀況  
{  
    "success": false,  
    "message": "請重新登入"  
}
```

以上狀況下 就不需要再重複登入

接下來加入導航守衛：在 main.js 加入 beforeEach

The screenshot shows a code editor with the following structure:

- 檔案總管**: Shows the project structure with files like login.vue, main.js, dev.env.js, etc.
- login.vue**: A Vue component file.
- JS main.js**: The file where the navigation guard is added.
- JS dev.env.js**: Configuration file.
- PRODUCT**: A folder containing product, config, and src.
- product**: Contains build and config.
- config**: Contains dev.env.js, index.js, and prod.env.js.
- src**: Contains assets, components, pages, router, and static.
- pages**: Contains login.vue and HelloWorld.vue.
- router**: Contains index.js, App.vue, and main.js.

The **main.js** code is as follows:

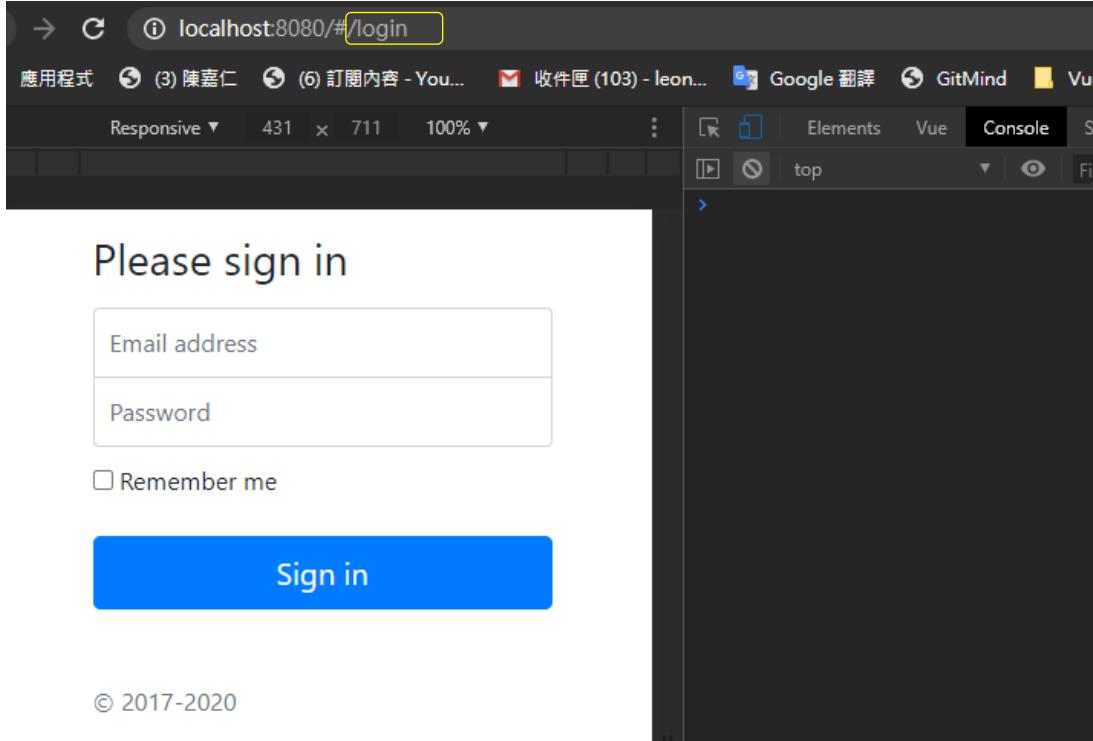
```
product > src > JS main.js > ...  
1 // The Vue build version to load with the `i  
2 // (runtime-only or standalone) has been set  
3 import Vue from 'vue';  
4 import axios from 'axios';  
5 import VueAxios from 'vue-axios';  
6 import App from './App'  
7 import router from './router'  
8  
9 Vue.use(VueAxios, axios)  
10 Vue.config.productionTip = false  
11  
12 axios.defaults.withCredentials = true;  
13  
14 /* eslint-disable no-new */  
15 new Vue({  
16     el: '#app',  
17     router,  
18     components: { App },  
19     template: '<App/>'  
20 })  
21  
22 router.beforeEach((to, from, next) => {  
23     next();  
24 })  
25  
26
```

其中 先給個 next() 方法主要是因為 加入 beforeEach 後 讓導航守衛 鎖住所有路由頁面，因此加入 next() 先給過 (將可隨意切換頁面)

此時 用 console 來看看 以上三種參數會打印出什麼：

```
21
22 router.beforeEach((to, from, next) => {
23   console.log('to', to)
24   console.log('from', from)
25   console.log('next', next)
26   next();
27 })
```

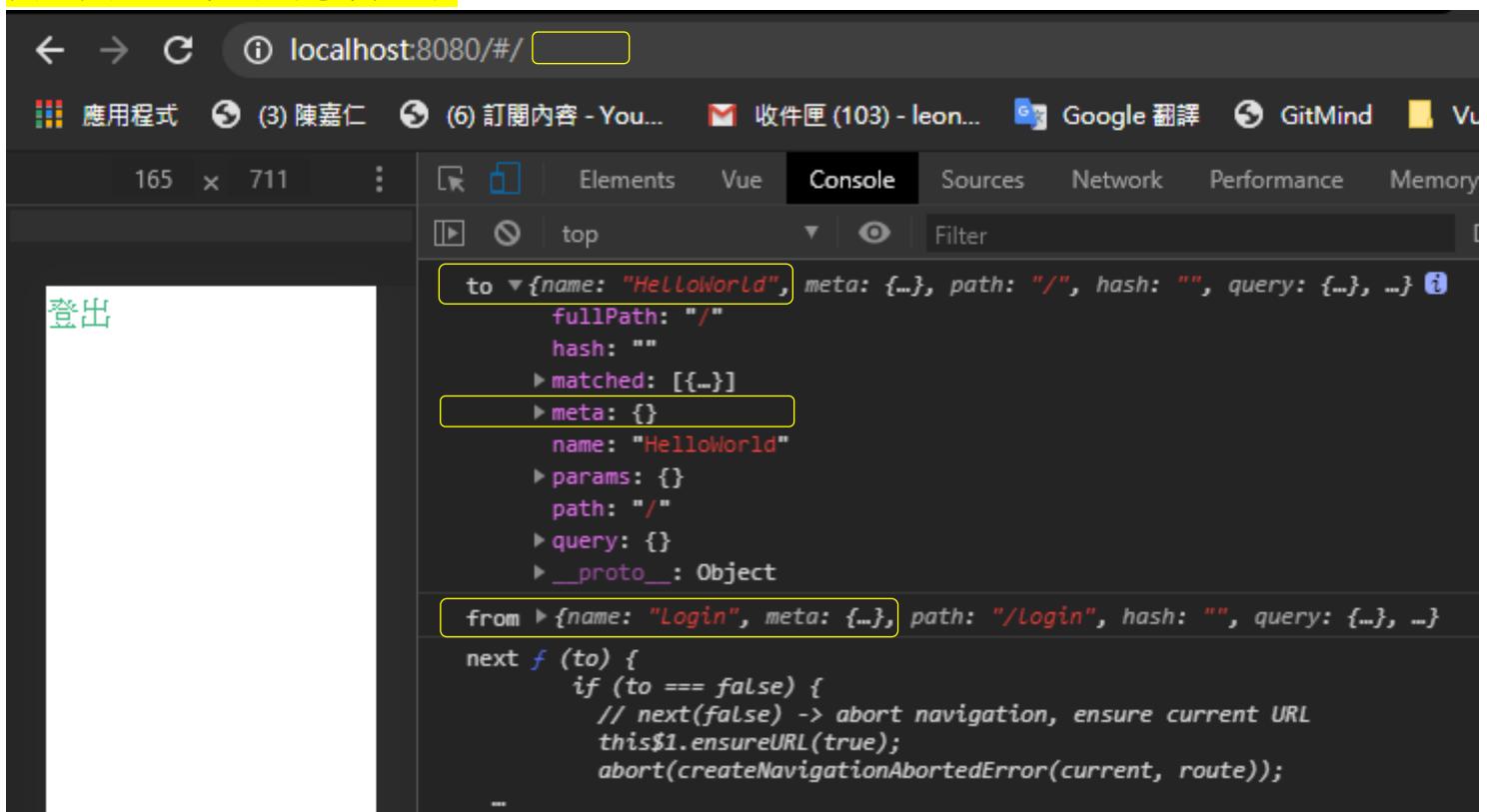
首先在 login 頁面



然後 變更手動 網址 切換到 首頁：

此時 console 的訊息顯示 to：到 HelloWorld / from：從 Login 頁面過來

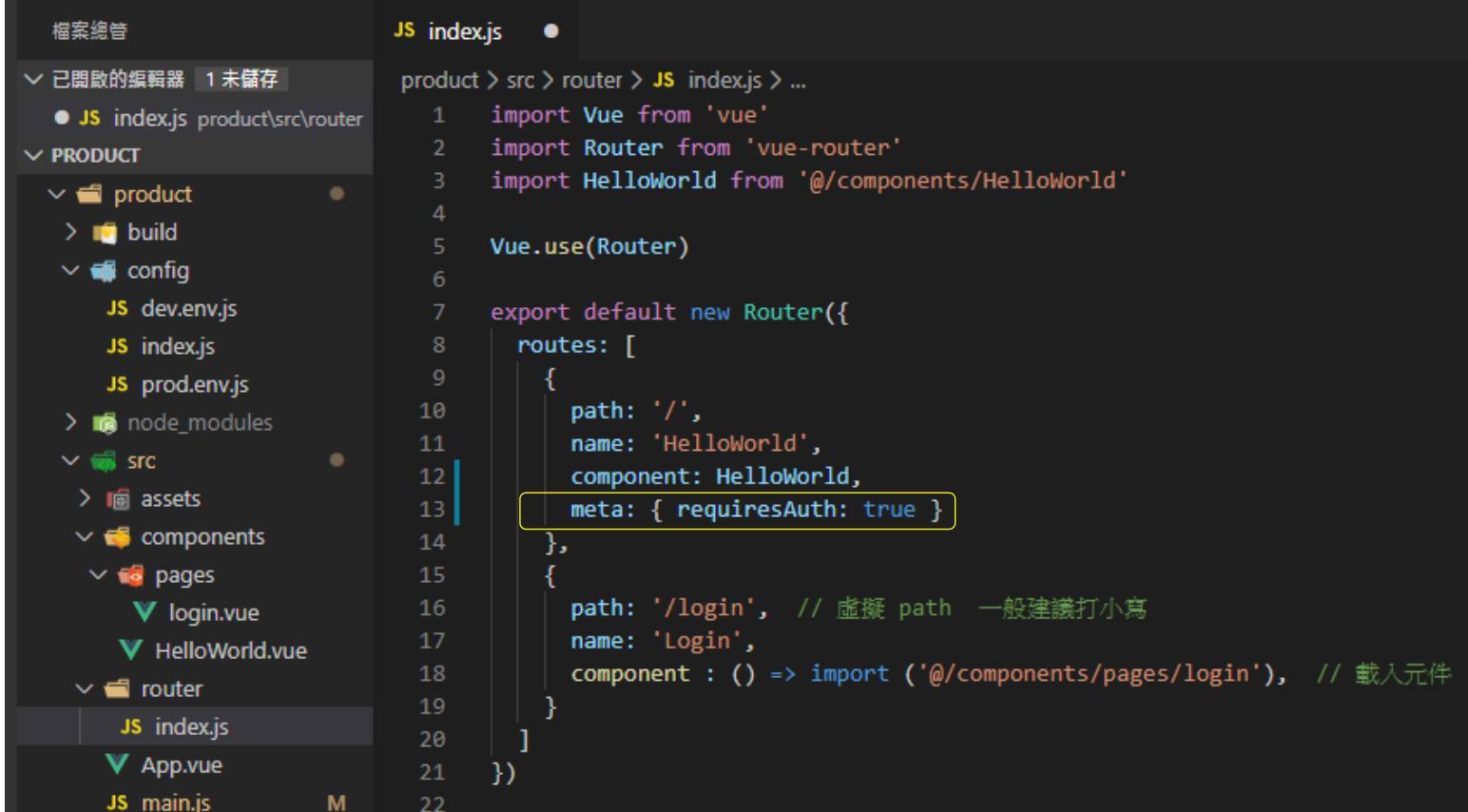
但此時的 meta (路由訊息) 是空值



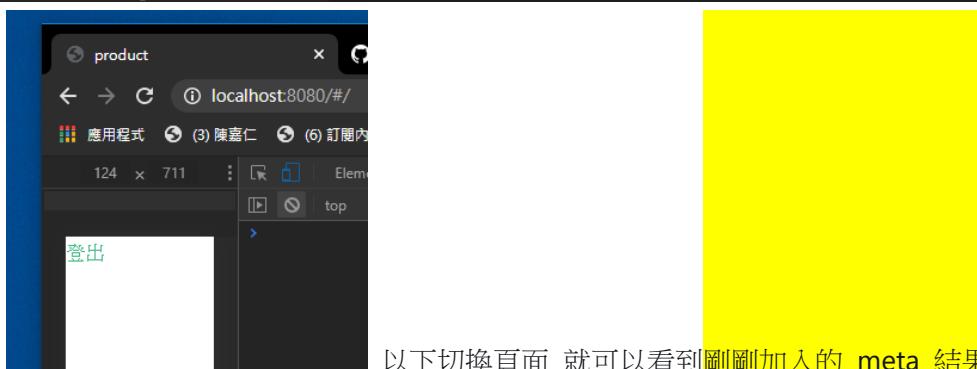
所以這時候可以把 meta：路由訊息 加在 Router 上面：

將剛剛 官方給的內容 增添進去

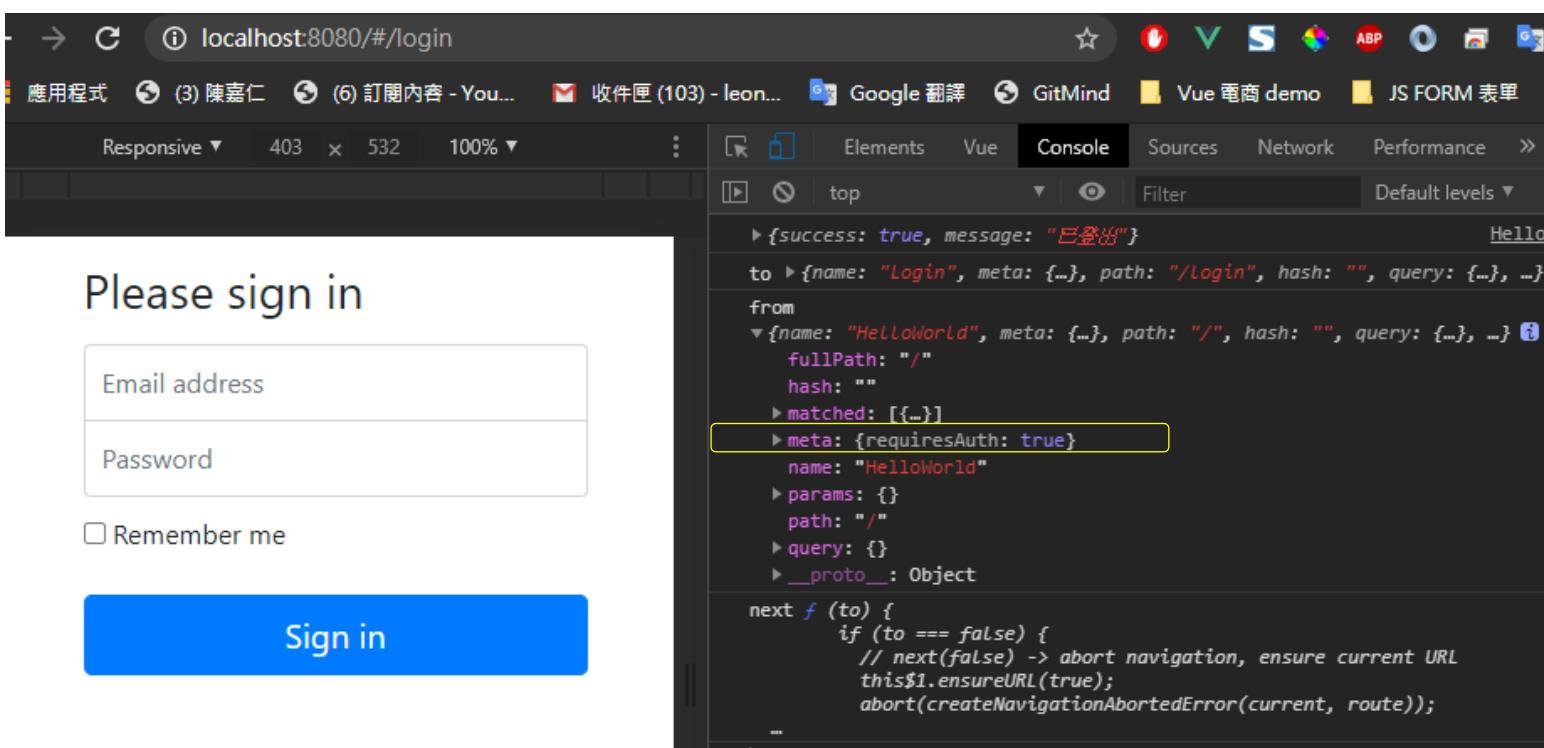
meta requireAuth (true or false 是個判斷基準)



```
product > src > router > JS index.js > ...
1 import Vue from 'vue'
2 import Router from 'vue-router'
3 import HelloWorld from '@/components/HelloWorld'
4
5 Vue.use(Router)
6
7 export default new Router({
8   routes: [
9     {
10       path: '/',
11       name: 'HelloWorld',
12       component: HelloWorld,
13       meta: { requiresAuth: true }
14     },
15     {
16       path: '/login', // 虛擬 path 一般建議打小寫
17       name: 'Login',
18       component : () => import('@/components/pages/login'), // 載入元件
19     }
20   ]
21 })
22 })
```



以下切換頁面 就可以看到剛剛加入的 meta 結果



Please sign in

Email address

Password

Remember me

Sign in

Console output:

```
▶ {success: true, message: "已登錄"}
to ▶ {name: "Login", meta: {...}, path: "/Login", hash: "", query: {...}, ...}
from
▼ {name: "HelloWorld", meta: {...}, path: "/", hash: "", query: {...}, ...}
  fullPath: "/"
  hash: ""
  ▶ matched: [{}]
  ▶ meta: {requiresAuth: true}
    name: "HelloWorld"
  ▶ params: {}
  ▶ path: "/"
  ▶ query: {}
  ▶ __proto__: Object
next f (to) {
  if (to === false) {
    // next(false) -> abort navigation, ensure current URL
    this$1.ensureURL(true);
    abort(createNavigationAbortedError(current, route));
  }
}
```

因此針對以上寫判斷式：

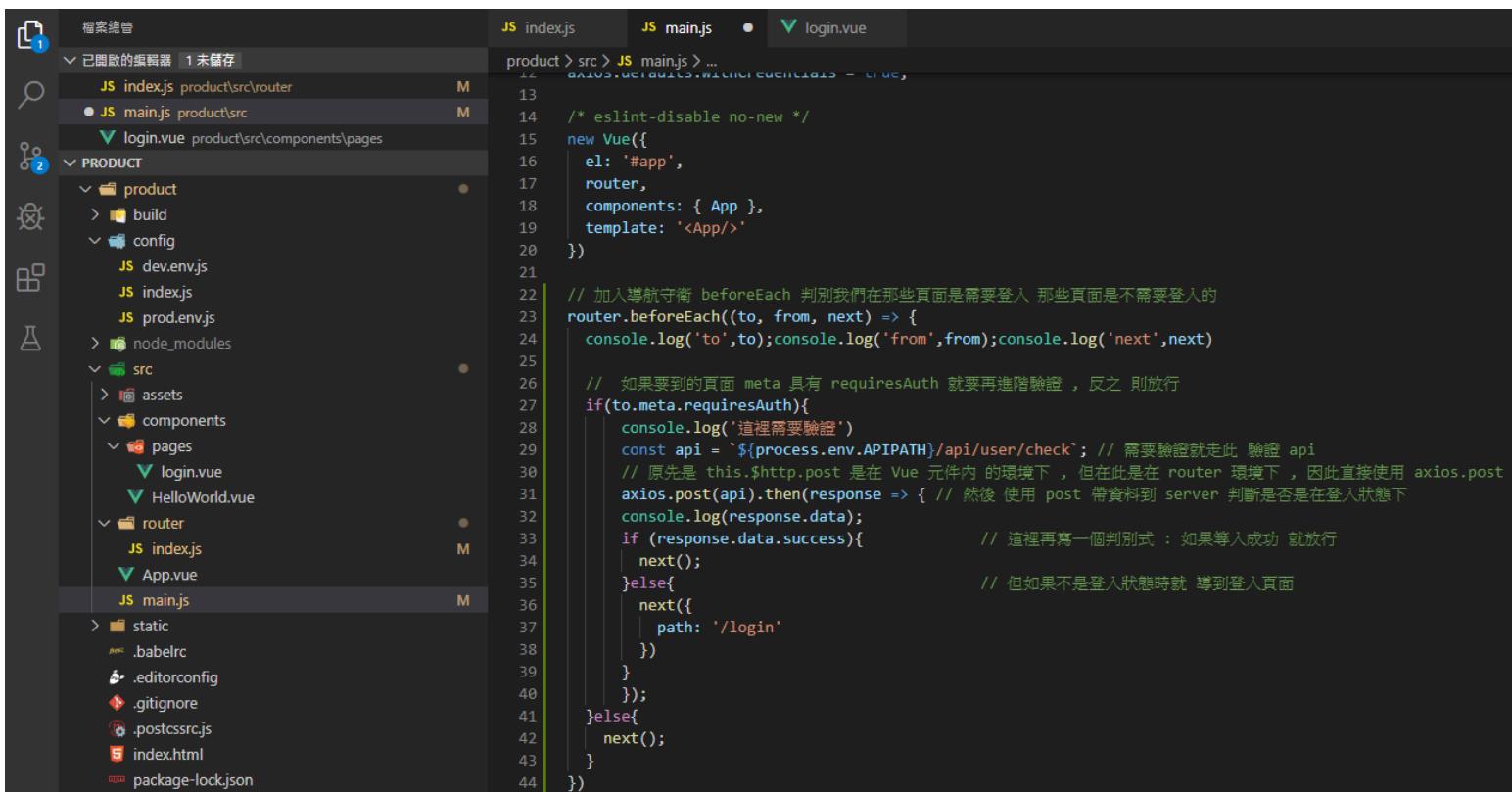
- 針對 `to.meta.requestAuth` 判別 true or false (目前都還是 false)
- True 就走驗證 api : `/api/user/check` 便會自動透過 server 寫好的驗證機制判別使用者狀態

檢查用戶是否仍持續登入

```
[API]: /api/user/check  
[方法]: post  
[說明]: 檢查用戶是否仍持續登入  
[成功回應]:  
{  
    "success": true  
}  
[失敗回應]: 如用戶已登出狀況  
{  
    "success": false,  
    "message": "請重新登入"  
}
```

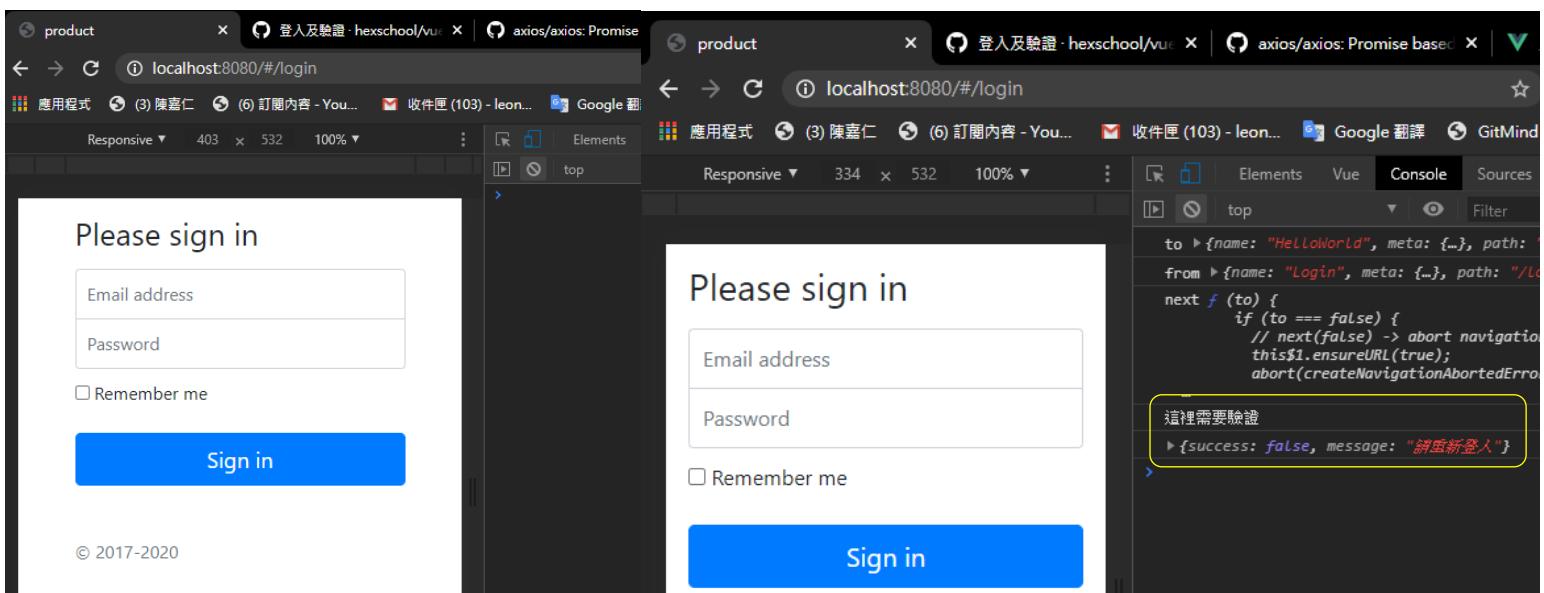
此時再辦別 如果經 server 判斷是登入狀態就是 成功 就放行，如果是未登入狀態就 導向到 登入頁面。

- False 就直接放行 `next()`;



```
product > src > JS main.js > ...  
  12   axios.defaults.withCredentials = true,  
  13  
  14  /* eslint-disable no-new */  
  15  new Vue({  
  16    el: '#app',  
  17    router,  
  18    components: { App },  
  19    template: '<App>'  
  20  })  
  21  
  22 // 加入導航守衛 beforeEach 判別我們在那些頁面是需要登入 那些頁面是不需要登入的  
  23 router.beforeEach((to, from, next) => {  
  24   console.log('to', to); console.log('from', from); console.log('next', next)  
  25  
  26   // 如果要到的頁面 meta 具有 requiresAuth 就要再進驗證 , 反之 則放行  
  27   if(to.meta.requiresAuth){  
  28     console.log('這裡需要驗證')  
  29     const api = `${process.env.APIPATH}/api/user/check`; // 需要驗證就走此 驗證 api  
  30     // 原先是 this.$http.post 是在 Vue 元件內 的環境下 , 但在此是在 router 環境下 , 因此直接使用 axios.post  
  31     axios.post(api).then(response => { // 然後 使用 post 帶資料到 server 判斷是否是在登入狀態下  
  32       console.log(response.data);  
  33       if(response.data.success){ // 這裡再寫一個判別式 : 如果登入成功 就放行  
  34         next();  
  35       }else{ // 但如果不是登入狀態時就 導到登入頁面  
  36         next({  
  37           path: '/login'  
  38         })  
  39       }  
  40     }  
  41   }else{  
  42     next();  
  43   }  
  44 }  
  45 }
```

如下 在未登入情況下 我自行在網址想要刪掉 login 到首頁，結果無法，然後 console 跳出提示訊息 (反之 登入過就沒問題，狀態會記錄在 server 服務裡)



product 登入及驗證 - hexschool/vue axios/axios: Promise

product 登入及驗證 - hexschool/vue axios/axios: Promise based

Please sign in

Email address

Password

Remember me

Sign in

Responsive 403 x 532 100% Elements top

Responsive 334 x 532 100% Elements Vue Console Sources top

```
to ▶ {name: "HelloWorld", meta: {}, path: ""}  
from ▶ {name: "Login", meta: {}, path: "/login"}  
next f (to) {  
  if (to === false) {  
    // next(false) -> abort navigation  
    this$1.ensureURL(true);  
    abort(createNavigationAbortedError());  
  }  
}  
 這裡需要驗證  
▶ {success: false, message: "請重新登入"}
```

另外在 router 增添一個預防使用者 亂打網址會白畫面的防呆機制，使其 打錯網址就導到 登入頁

```
JS main.js      JS router.js  X
product > src > JS router.js > ...
1  import Vue from 'vue';
2  import Router from 'vue-router';
3
4  import HelloWorld from '@/components/HelloWorld';
5  import Login from '@/components/pages/login';
6
7  Vue.use(Router)
8
9  export default new Router({
10    routes: [
11      {
12        path: '*',
13        redirect: 'login'          // 避免用亂打進入不存在的頁面，就導回 登入頁
14      },
15      {
16        path: '/',
17        name: 'HelloWorld',
18        component: HelloWorld,
19        meta: { requiresAuth: true } // 導航守衛的 路由訊息
20      },
21      {
22        path: '/login',           // 虛擬 path 一般建議打小寫
23        name: 'Login',
24        component: Login,         // 載入元件
25        // component : () => import('@/components/pages/login'),
26      }
27    ]
28  })

```

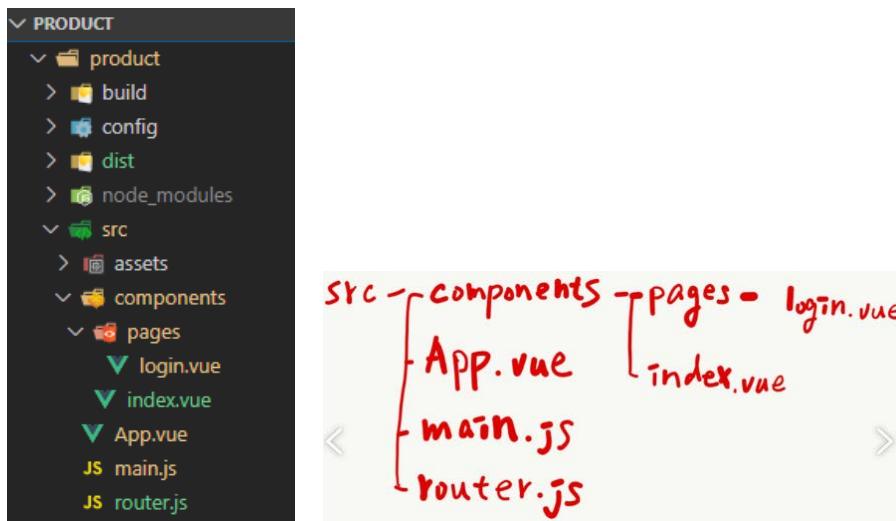
個人 單元小整理 :

這裡我大概整理上述內容：

小更動：原本的暫時性首頁 - 元件名稱 HelloWord.vue => index.vue

Router>index.js 直接改成 router.js

目錄結構 :



概述 : (子元件 >>> 父元件)

Components :

裡定義各種元件可重複利用 (目前主要做 router 為主),

檔案內容結構 <template> : 撰寫 HTML 結構

<script> : 撰寫 與 template 綁定的資料結構、方法

<style> : 撰寫 template 的 樣式

```
App.vue index.vue login.vue
product > src > components > index.vue > {} "index.vue" > script >
1  <template>
2    <div class="hello">
3      | <a href="#" @click.prevent="signout">登出</a>
4    </div>
5  </template>
6
7  <script>
8  export default {
9    name: 'Index',
10   data () {
11     return [
12       msg: 'Welcome to Your Vue.js App'
13     ]
14   },
15   methods:{
16     signout(){
17       // 依據 API 文件指出此api 址需傳入 api 不須
18       const api = `${process.env.APIPATH}/logout`;
19       const vm = this;
20       this.$http.post(api).then(response => { // 然後
21         console.log(response.data);
22         if (response.data.success){ // 這裡寫一個判別式
23           || vm.$router.push('/login')
24         }
25       });
26     }
27   }
28 }
29 </script>
30
31 <!-- Add "scoped" attribute to limit CSS to this component -->
32 <style scoped>
33 h1, h2 {
34   font-weight: normal;
35 }
36 ul {
37   list-style-type: none;
38   padding: 0;
39 }
40 li {
41   display: inline-block;
42   margin: 0 10px;
43 }
44 a {
45   color: #42b983;
46 }
47 </style>
```

↓ ↓ ↓

各種 元件 撰寫好後 就由 Router 統一管理 (所有路由的 配置檔)

首先將 各個位置的元件 import 寄來在給予 名稱

然後賦予 虛擬的 path (此 path 便是 使用者輸入網址後 會啟用元件的 導向) **輸入該網址元件就會出現在畫面上

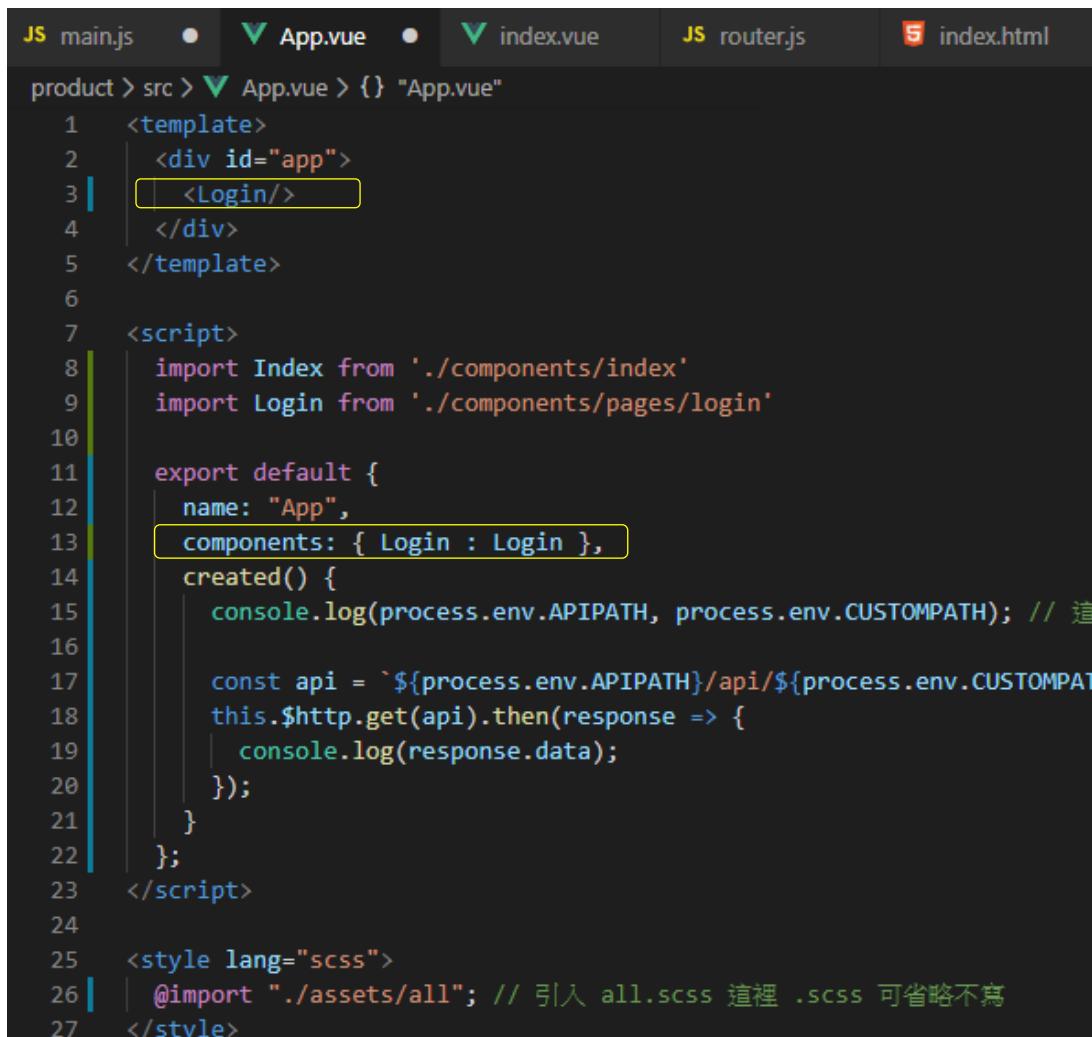
```
JS main.js      JS router.js ×
product > src > JS router.js > ...
1  import Vue from 'vue';
2  import Router from 'vue-router';
3
4  import Index from '@/components/index';
5  import Login from '@/components/pages/login';
6
7  Vue.use(Router)
8
9  export default new Router({
10    routes: [
11      {
12        path: '*',
13        redirect: 'login'          // 避免用亂打進入不存在的頁面，就導回 登入頁
14      },
15      {
16        path: '/',
17        name: 'Index',
18        component: Index,
19        meta: { requiresAuth: true } // 導航守衛的 路由訊息
20      },
21      {
22        path: '/login',           // 虛擬 path 一般建議打小寫
23        name: 'Login',
24        component: Login,        // 載入元件
25        // component: () => import('@/components/pages/login'),
26      }
27    ]
28  })

```

↓ ↓ ↓

當 Router 邊寫好後 一般來說 會將 所有的 router 元件都 統一再掛到 一個 APP.vue 元件底下

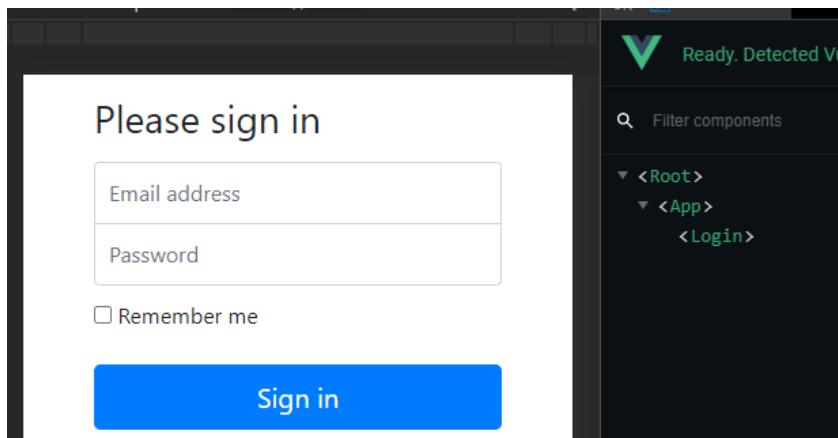
原本的概念是類似以下 將多個元件放進 <div id='app'> 裏頭，但此方法便無法 用上 router 透過 path 切換元件的效果



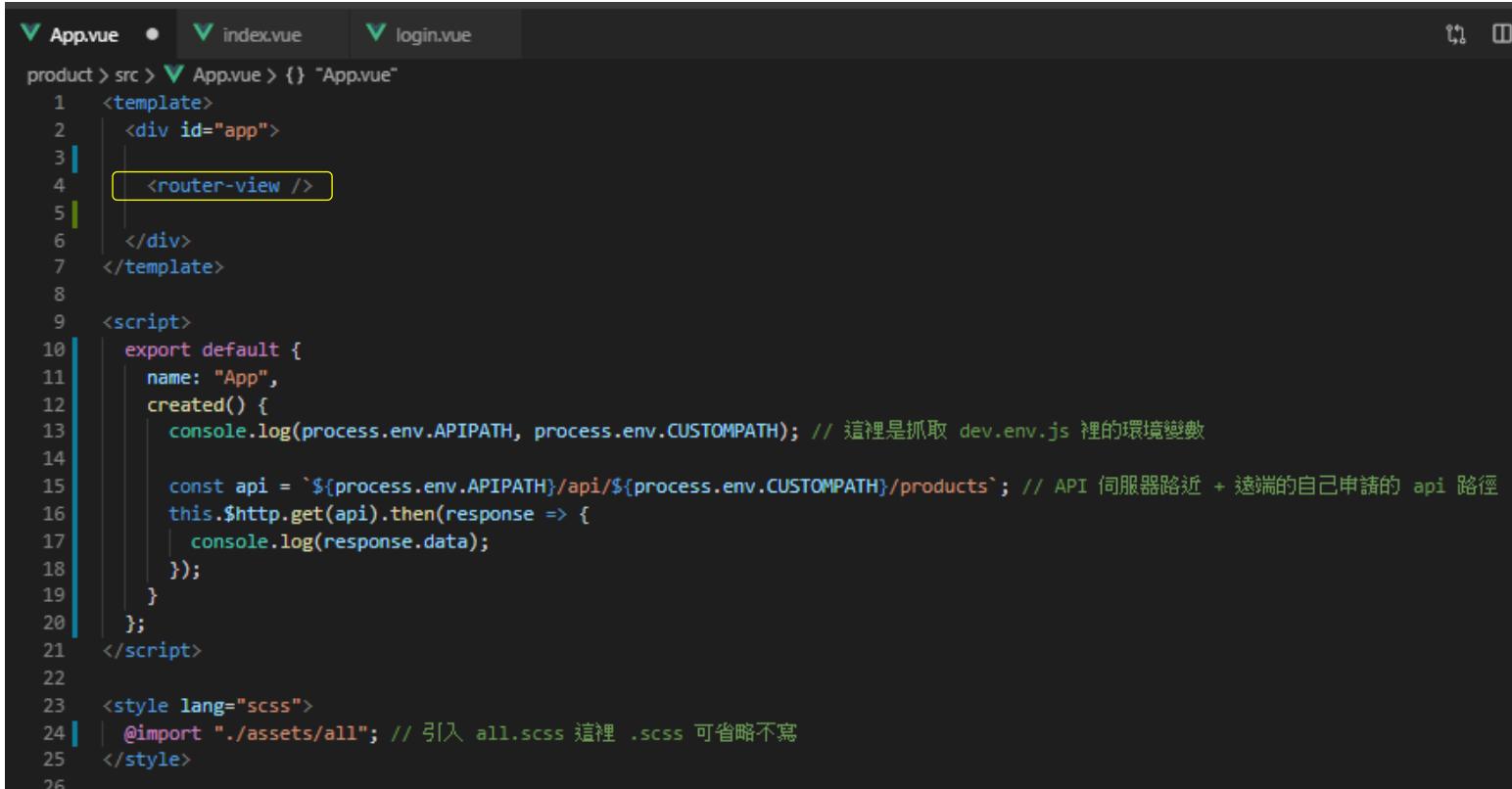
The screenshot shows the code editor with the App.vue file open. The code defines a template with a single

element containing a component. In the script section, the component is registered under the name 'Login'. The component is highlighted with a yellow box.

```
product > src > App.vue > {} "App.vue"
1  <template>
2  |   <div id="app">
3  |     <Login/>
4  |   </div>
5  </template>
6
7  <script>
8    import Index from './components/index'
9    import Login from './components/pages/login'
10
11   export default {
12     name: "App",
13     components: { Login : Login },
14     created() {
15       console.log(process.env.APIPATH, process.env.CUSTOMPATH); // 這
16
17       const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPAT
18       this.$http.get(api).then(response => {
19         console.log(response.data);
20       });
21     }
22   };
23 </script>
24
25 <style lang="scss">
26   @import "./assets/all"; // 引入 all.scss 這裡 .scss 可省略不寫
27 </style>
```



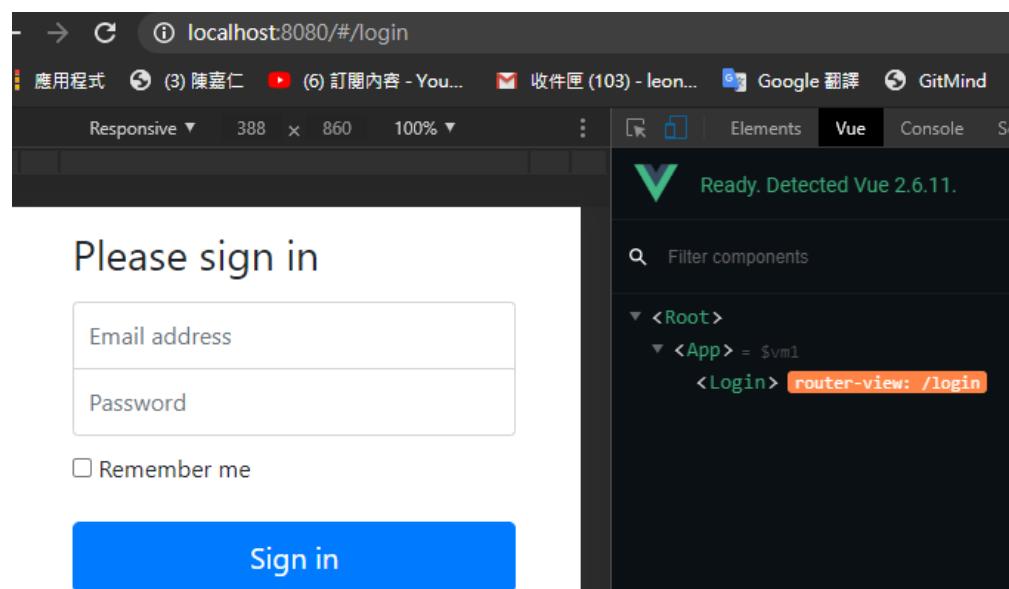
於是 要 使用 <router-view> 做掛載點 此時 當 虛擬 path 切換時 就可以在 <router-view> 上任意切換



The screenshot shows a code editor with two tabs: 'App.vue' and 'login.vue'. The 'App.vue' tab is active, displaying the following code:

```
product > src > App.vue > {} "App.vue"
1  <template>
2    <div id="app">
3      <router-view />
4    </div>
5  </template>
6
7  <script>
8    export default {
9      name: "App",
10     created() {
11       console.log(process.env.APIPATH, process.env.CUSTOMPATH); // 這裡是抓取 dev.env.js 裡的環境變數
12
13       const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/products`; // API 伺服器路徑 + 遠端的自己申請的 api 路徑
14       this.$http.get(api).then(response => {
15         console.log(response.data);
16       });
17     }
18   };
19 </script>
20
21 <style lang="scss">
22   @import "./assets/all"; // 引入 all.scss 這裡 .scss 可省略不寫
23 </style>
24
25
26
```

Vue 開發者工具可以看的出來 APP 下 掛著 Login (router-view)



最後再將以上所有檔案通通匯至我們打包的進入點 main.js：

1. APP.vue import：用元件方式掛載
 2. router.js import 掛載
 3. 最後再生成一個 main.js 裡面的 template : <APP/> 並將其 el: 掛置 index.html (經 webpack 打包完成後)

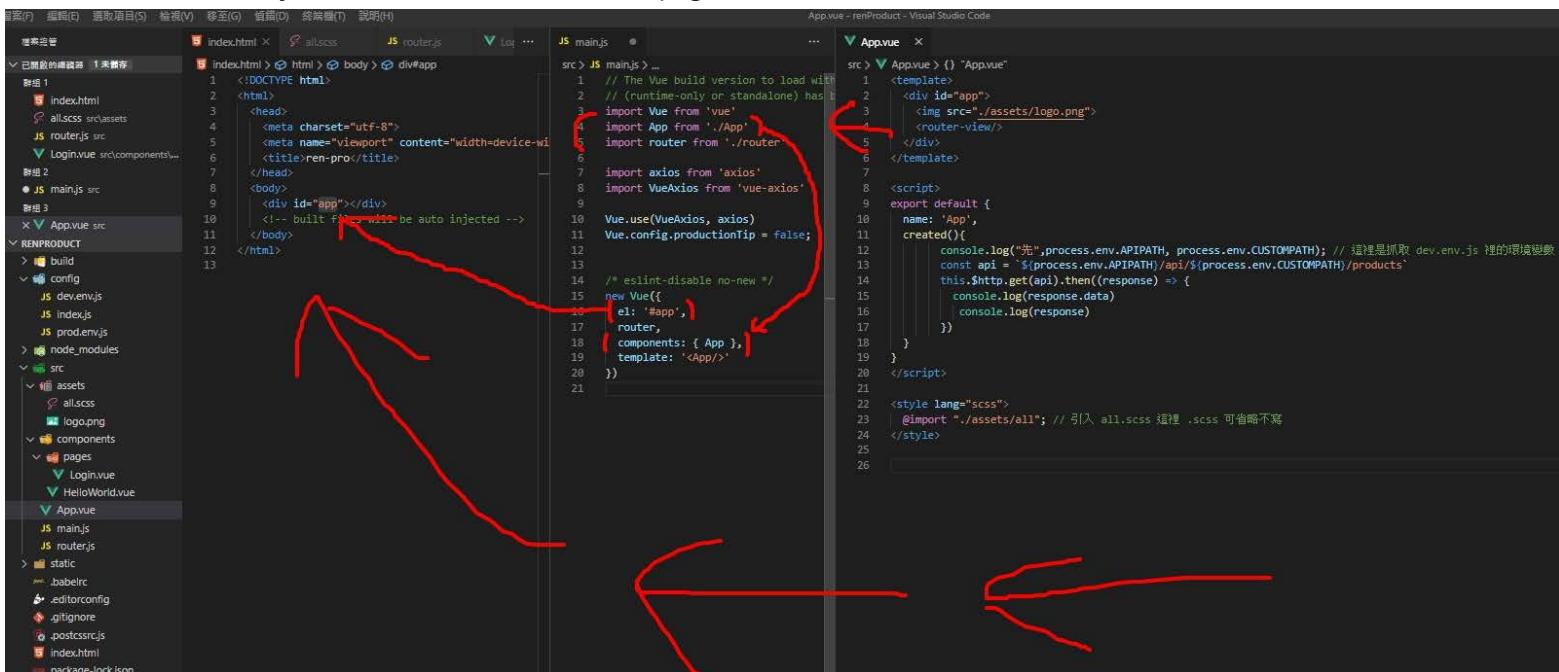
product > src > **JS** main.js

```
1 import Vue from 'vue';
2 import axios from 'axios';
3 import VueAxios from 'vue-axios';
4 
5 import App from './App'
6 import router from './router'
7 
8 Vue.use(VueAxios, axios)
9 Vue.config.productionTip = false
10 
11 axios.defaults.withCredentials = true;
12 
13 /* eslint-disable no-new */
14 new Vue({
15   el: '#app',
16   router,
17   components: { App },
18   template: '<App/>'
19 })
20 
21 router.beforeEach((to, from, next) => {           // 加入導航守衛 beforeEach 判別我們在那些頁面是需要登入 那些頁面是不需要登入的
22   console.log('to', to, 'from', from, 'next', next);
23   if(to.meta.requiresAuth){                         // 如果要到的頁面 meta 具有 requiresAuth 就要再進階驗證 , 反之 則放行
24     console.log('這裡需要驗證')
25     const api = `${process.env.APIPATH}/api/user/check`; // 需要驗證就走此 驗證 api
26     // 原先是 this.$http.post 是在 Vue 元件內 的環境下 , 但在此是在 router 環境下 , 因此直接使用 axios.post
27     axios.post(api).then(response => { // 然後 使用 post 帶資料到 server 判斷是否是在登入狀態下
28       console.log(response.data);
29       if (response.data.success){          // 這裡再寫一個判別式 : 如果登入成功 就放行
30         next();
31       }else{                            // 但如果不是登入狀態時就 導到登入頁面
32         next({
33           path: '/login'
34         })
35       }
36     });
37   }else{
38     next();
39   }
40 })
```

結論：

- 各種 component 撰寫後
 - 並編輯好 router 配置後
 - 會將所有 component 汇集在 App.vue 裡，並透過 <router-view/> 做切換。

最後在 `main.js` 上引入以上、所有所需要的 plugin



11 - 91 套用 Bootstrap Dashboard 版型

到 bootstrap 官網找 dashboard 型並且套用

<https://getbootstrap.com/docs/4.5/examples/dashboard/>

將其版型複製起來 並 複製他額外自定義的 css

新增一個後台管理員的 儀錶板管理頁面

首先增添 dashboard.scss

The screenshot shows a code editor with two tabs: '_dashboard.scss' and 'all.scss'. The '_dashboard.scss' tab contains SCSS code for a dashboard layout, including styles for the body, feather icon, sidebar, and navigation components. The 'all.scss' tab contains imports for Bootstrap and other Bootstrap components. The file structure on the left shows a 'PRODUCT' folder containing 'product', 'src', and 'assets' subfolders, with 'Dashboard.vue' also listed.

```
body {
    font-size: .875rem;
}
.feather {
    width: 16px;
    height: 16px;
    vertical-align: text-bottom;
}
.sidebar {
    position: fixed;
    top: 0;
    bottom: 0;
    left: 0;
    z-index: 100; /* Behind the navbar */
    padding: 48px 0 0; /* Height of navbar */
    box-shadow: inset -1px 0 0 #eaeaea;
}
```

Import 進去 all.scss 做統一管理

The screenshot shows a code editor with three tabs: '_dashboard.scss', 'all.scss', and 'Dashboard.vue'. The 'all.scss' tab contains imports for Bootstrap and other Bootstrap components. The file structure on the left shows a 'PRODUCT' folder containing 'product', 'src', and 'assets' subfolders, with 'Dashboard.vue' also listed.

```
@import "~bootstrap/scss/bootstrap"; // 這裡指我要找 node module
@import "~bootstrap/scss/functions"; // 這裡載入 bs 套用變數的地方
@import "./helpers/_variables"; // 這裡載入 bs 我們自定義的變數
@import "./dashboard";
```

把結構 存成一個新的元件 叫 Dashboard.vue

The screenshot shows a code editor with three tabs: 'Dashboard.vue', 'index.vue', and 'App.vue'. The 'Dashboard.vue' tab contains the template for a dashboard component, including a navbar with a search bar and a sidebar menu. The file structure on the left shows a 'PRODUCT' folder containing 'product', 'src', and 'assets' subfolders, with 'Dashboard.vue' also listed.

```
<template>
<div>
    <nav class="navbar navbar-dark sticky-top bg-dark">
        <a class="navbar-brand col-md-3 col-lg-2" href="#">
            <button
                class="navbar-toggler position-absolute
                type="button"
                data-toggle="collapse"
                data-target="#sidebarMenu"
                aria-controls="sidebarMenu"
                aria-expanded="false"
                aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <input
                class="form-control form-control-dark w-100
                type="text"
                placeholder="Search"
                aria-label="Search">
            />
        <ul class="navbar-nav px-3">
            <li class="nav-item textnowrap">
                <a class="nav-link" href="#">Sign out</a>
            </li>
        </ul>
    </nav>
```

對其進行 router 管理：

```

    ▼ index.vue          U   23   | path: '/login',           // 虛擬 path 一般建議打小寫
    ▼ App.vue           M   24   | name: 'Login',
    JS main.js          M   25   | component: Login,        // 載入元件
    JS router.js         U   26   | // component : () => import ('@/components/pages/login'),
                                |
                                | },
                                |
                                | 27   | {
                                | 28   |   path: '/admin',
                                | 29   |   name: 'Admin',
                                | 30   |   component : Dashboard,
                                | 31   |   meta: { requiresAuth: true }
                                | 32   |
                                | 33   | }
                                | 34   |
                                | 35   | })

```

以上動作後 在進行更細部拆解 元件，將原本 navbar 的結構 也另外拆成元件：

```

product > src > components > ▼ Navbar.vue > ...
1  <template>
2  <div>
3  <nav class="navbar navbar-dark sticky-...
4  <a class="navbar-brand col-md-3 col-...
5  <button
6  class="navbar-toggler position-abs...
7  type="button"
8  data-toggle="collapse"
9  data-target="#sidebarMenu"
10 aria-controls="sidebarMenu"
11 aria-expanded="false"
12 aria-label="Toggle navigation"
13 >
14 <span class="navbar-toggler-icon"></span>
15 </button>
16 <input
17 class="form-control form-control-d...
18 type="text"
19 placeholder="Search"
20 aria-label="Search"
21 />
22 <ul class="navbar-nav px-3">
23 <li class="nav-item text-nowrap">
24 | <a class="nav-link" href="#">Signi...
25 </li>
26 </ul>
27 </nav>
28 </div>
29 </template>

```

```

product > src > components > ▼ Sidebar.vue > template > div > nav#sid...
1  <template>
2  <div>
3  <nav id="sidebarMenu" class="col-md-3 col-lg-2 d-m...
4  <div class="sidebar-sticky pt-3">
5  <ul class="nav flex-column">
6  <li class="nav-item">
7  <a class="nav-link active" href="#">
8  <span data-feather="home"></span>
9  Dashboard
10 <span class="sr-only">(current)</span>
11 </a>
12 </li>
13 <li class="nav-item">
14 <a class="nav-link" href="#">
15 <span data-feather="file"></span>
16 Orders
17 </a>
18 </li>
19 <li class="nav-item">
20 <a class="nav-link" href="#">
21 <span data-feather="shopping-cart"></span>
22 Products
23 </a>
24 </li>
25 <li class="nav-item">
26 <a class="nav-link" href="#">
27 <span data-feather="users"></span>
28 Customers
29 </a>
30 </li>
31 <li class="nav-item">
32 <span data-feather="grid"></span>
33 <span>Dashboard</span>
34 </li>
35 </ul>
36 </div>
37 </nav>
38 </div>
39 </template>

```

接著透過先前組元件方式，復原我們的 dashboard 結構 (要將內容 <main> 清空 待會填入我們的內容)

```

product > src > components > ▼ Dashboard.vue > {} "Dashboard.vue"
1  <template>
2  <div>
3  <Navbar></Navbar> <!-- Navbar -->
4  <div class="container-fluid">
5  <div class="row">
6  <Sidebar></Sidebar> <!-- Sidebar -->
7  <main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-md-4">
8
9
10 </main>
11 </div>
12 </div>
13 </div>
14 </template>
15
16 <script>
17 import Sidebar from "./Sidebar";
18 import Navbar from "./Navbar";
19
20 export default {
21   components: {
22     Sidebar,
23     Navbar
24   }
25 };
26 </script>

```

加入 管理員 的產品儀表板控制頁面 然後讓 此 product 頁面依附在 dashboard.vue 之下做 router 切換：

The screenshot shows the file structure and content of `Products.vue`. The file is located at `product > src > components > pages > Products.vue`. The code contains a template with a single `<div>` element containing the text "這邊將會有 管理員的 管理產品". The file has 23 lines of code.

對 product.vue 頁面 進行路由配置

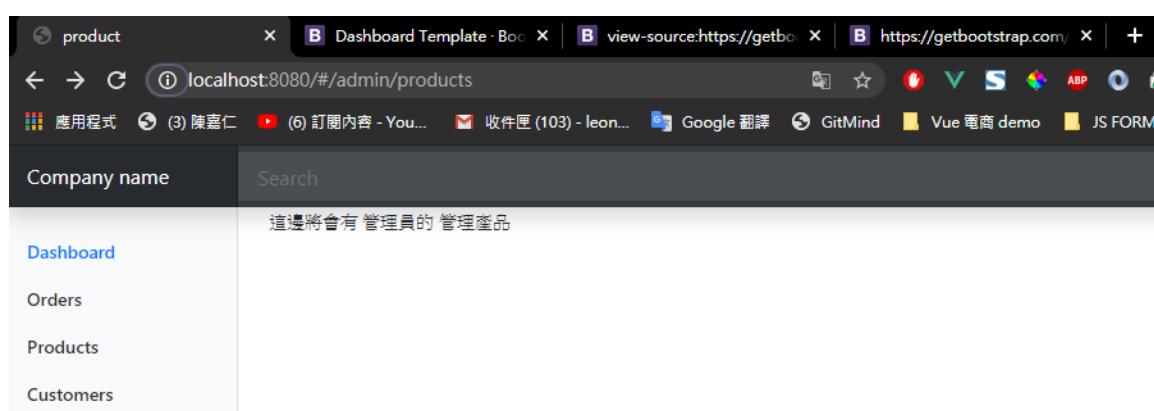
他是依附在 admin 管理員的 dashboard 之下進行路由切換

此時的 router-view 是巢狀的 因為外層的 APP.vue 還有一個 router view

The screenshot shows the `router.js` file structure and content. The file is located at `product > src > JS router.js > routes`. It defines two main routes: one for `/login` and another for `/admin`. The `/admin` route has a nested route for `/products`. A yellow box highlights the nested route configuration for `/products`.

```
product > src > JS router.js > routes
  22  },
  23  {
  24    path: '/login',           // 虛擬 path 一般建議打小寫
  25    name: 'Login',
  26    component: Login,        // 載入元件
  27    // component : () => import ('@/components/pages/login'),
  28  },
  29
  30  // 當進入 admin 時會進入另一個 router ( 巢狀 router )
  31  {
  32    path: '/admin',
  33    name: 'Admin',
  34    component : Dashboard,
  35    meta: { requiresAuth: true },
  36    children : [
  37      {
  38        path: 'products',
  39        name: 'Products',
  40        component: Products,
  41        meta: { requiresAuth: true } , // 這裡一樣加上 確保進入此頁面前是需要被驗證的
  42      },
  43    ]
  44  }
  45}
  46}
```

此時的 product 頁面也需要登入才可看到



11 - 92 製作產品列表

首先 移除先前的 index.vue 檔 的 router 功能

再 將之前再 APP.vue 檔的 ajax 取得資料 改放到 產品頁面 ,

```
JS router.js ×
product > src > JS router.js > ...
  9  Vue.use(Router)
 10 |
 11 export default new Router({
 12   routes: [
 13     {
 14       path: '*',
 15       redirect: 'login'          // 避免直接打 *
 16     },
 17     // {
 18     //   path: '/',
 19     //   name: 'Index',
 20     //   component: Index,
 21     //   meta: { requiresAuth: true } // 需要登入才能看
 22     // },
 23     {
 24       path: '/login',           // 虛擬路徑
 25       name: 'Login',
 26       component: Login,        // 載入元件
 27       // component : () => import('@/components/Login')
 28     },
 29   ],
 30 }
```

將取得產品資料的 ajax 移動至 products.vue 檔 , 並設計 data 資料結構 , 將取得的產品資訊塞進 products 紿予 template 做渲染

```
JS router.js      ▾ Products.vue ×      ▾ Index.vue
product > src > components > pages > ▾ Products.vue > {} "Products.vue"
  35  | </div>
  36  </template>
  37
  38 <script>
  39 export default {
  40   data(){
  41     return {
  42       products: [] // 所有新增的產品資料會放進這裡
  43     }
  44   },
  45   methods:{
  46     getProducts(){
  47       // 取得遠端產品資料 然後將 ajax取得的資料存進 data 的products
  48       const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/products`; // API 向服務器路徑 + 遠端的自己申請的 api 路徑
  49       const vm = this;
  50       this.$http.get(api).then(response => {
  51         console.log(response.data);
  52         vm.products = response.data.products; // 將取得的產品資訊塞進 products 紿予template 做渲染
  53       });
  54     }
  55   },
  56   created(){ // 上方的 getProducts 需要在 created 裡呼叫執行
  57     this.getProducts();
  58   }
  59 };
 60 </script>
```

在 Template 做資料綁定 剛剛 抓取道的 data 裡的 products

```
JS router.js          ▼ Products.vue ● ▼ Index.vue
product > src > components > pages > ▼ Products.vue > {} "Products.vue"
1   <template>
2     <div>
3       <div class="text-right mt-4">
4         <button class="btn btn-primary">建立新產品</button>
5       </div>
6       <table class="table mt-4">
7         <thead>
8           <th width="120">分類</th>
9           <th>產品名稱</th>
10          <th width="120">數量</th>
11          <th width="120">原價</th>
12          <th width="120">售價</th>
13          <th width="80">狀態</th>
14          <th width="80">編輯</th>
15        </thead>
16        <tbody>
17          <tr v-for="item in products" :key="item.id">
18            <td>{{item.category}}</td>
19            <td>{{item.title}}</td>
20            <td>{{item.num}}</td>
21            <td class="text-right">{{item.origin_price}}</td>
22            <td class="text-right">{{item.price}}</td>
23            <td>
24              <span v-if="item.is_enable" class="text-success">啟用</span>
25              <span v-else>未啟用</span>
26            </td>
27            <td>
28              <button class="btn btn-outline-primary btn-sm">編輯</button>
29            </td>
30          </tr>
31        </tbody>
32      </table>
33    </div>
34  </template>
```

目前畫面

Company name	Search	Sign out
Dashboard		建立新產品
Orders		
Products	分類 產品名稱 數量 原價 售價 狀態 編輯	
Customers	測試分類 Canon 1 28000 25000 未啟用	編輯
Reports		
Integrations	Nikon D750 測試的產品 2 35000 33000 未啟用	編輯
SAVED REPORTS		
Current month		
Last quarter		
Social engagement		

11 - 93 Vue 中運用 Bootstrap 及 jQuery

我預想要在 產品頁面按下 建立新產品 按鈕時 會有個 彈出視窗 (modal)

因此到 bootstrap 官網的 modal 範例進行下載樣本

<https://getbootstrap.com/docs/4.0/components/modal/>

以下 data-target 的 id 對象就是控制對象

Live demo

Toggle a working modal demo by clicking the button below. It will slide down and fade in from the top of the page.

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
  Launch demo modal
</button>

<!-- Modal -->


<div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>


```

然後 在 main.js import bootstrap, 並且要在下載 bs 相依的 jquery 與 popper.js

檔案總管

- 已開啟的編輯器 1 未儲存
 - JS router.js product\src
 - JS main.js product\src
- PRODUCT
 - product
 - build
 - config
 - dist
 - node_modules
 - src
 - assets
 - components

JS router.js JS main.js ●

```
product > src > JS main.js > router.beforeEach() callback
1   import Vue from 'vue';
2   import axios from 'axios';
3   import VueAxios from 'vue-axios';
4   import 'bootstrap';
5
6
7   import App from './App'
8   import router from './router'
```

問題 輸出 僵錯主控台 終端機

要終止批次工作嗎 (Y/N)? y

PS D:\CodePractice\product\product> npm i --save jquery popper.js

npm WARN deprecated popper.js@1.16.1: You can find the new Popper v2 at [https://popper.js.org](#)

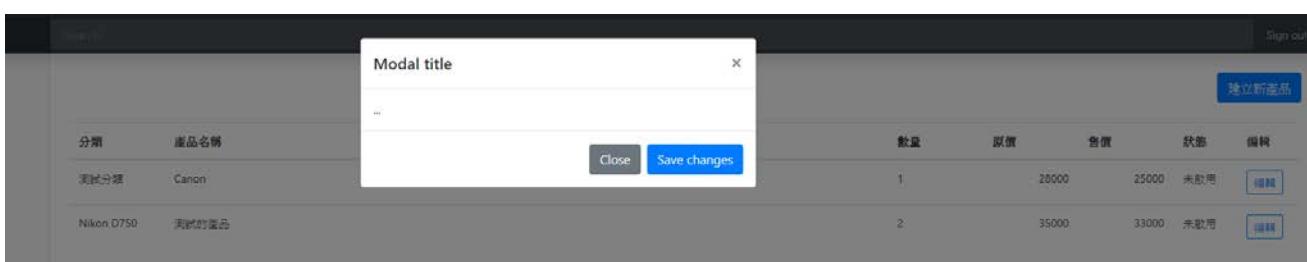
依照上述 把 bs 加入 product 頁面 並 用上 modal 操作

Products.vue

```
product > src > components > pages > Products.vue > {} > Products.vue > template > div > table.table.mt-4 > thead
```

```
1  <template>
2    <div>
3      <div class="text-right mt-4" data-toggle="modal" data-target="#productModal">
4        <button class="btn btn-primary">建立新產品</button>
5      </div>
6      <table class="table mt-4">
7        <thead>
8          <th width="120">分類</th>
9          <th>產品名稱</th>
10         <th width="120">數量</th>
11         <th width="120">原價</th>
12         <th width="120">售價</th>
13         <th width="80">狀態</th>
14         <th width="80">編輯</th>
15       </thead>
16       <tbody>
17         <tr v-for="item in products" :key="item.id">
18           <td>{{item.category}}</td>
19           <td>{{item.title}}</td>
20           <td>{{item.num}}</td>
21           <td class="text-right">{{item.origin_price}}</td>
22           <td class="text-right">{{item.price}}</td>
23           <td>
24             <span v-if="item.is_enable" class="text-success">啟用</span>
25             <span v-else>未啟用</span>
26           </td>
27           <td>
28             <button class="btn btn-outline-primary btn-sm">編輯</button>
29           </td>
30         </tr>
31       </tbody>
32     </table>
33     <!-- Modal -->
34     <div class="modal fade" id="productModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
35       <div class="modal-dialog" role="document">
36         <div class="modal-content">
37           <div class="modal-header">
38             <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
39             <button type="button" class="close" data-dismiss="modal" aria-label="Close">
40               <span aria-hidden="true">&times;</span>
41             </button>
42           </div>
43           <div class="modal-body">
44             ...
45           </div>
46           <div class="modal-footer">
47             <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
48           </div>
49         </div>
50       </div>
51     </div>
52   </div>
```

按下 建立商品 btn 便有 彈出視窗效果



目前 以上的效果是透過 bs 效果去處發 modal

以下 會改由我們自己從 按鈕觸發 methods 方法 (以確保是在取得 資料後才 有 modal 出現)

在 bs modal 文件下方有個 methods

Live demo

Toggle a working modal demo by clicking the button below. It will slide down and fade in from the top of the page.

Launch demo modal

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&amptimes
        </button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal" aria-label="Close">Close</button>
        <button type="button" class="btn btn-primary" data-dismiss="modal" aria-label="Confirm">Confirm</button>
      </div>
    </div>
  </div>
</div>
```

Copy

- Scrolling long content
- Vertically centered
- Tooltips and popovers
- Using the grid
- Varying modal content
- Remove animation
- Dynamic heights
- Accessibility
- Embedding YouTube videos
- Optional sizes
- Large modal
- Small modal
- Usage
 - Via data attributes
 - Via JavaScript
 - Options
 - Methods
 - Events

使用其中的 show 可以將 bs 的 modal 透過 methods 方法 觸發開啟

`.modal('show')`

Manually opens a modal. **Returns to the caller before the modal has actually been shown** (i.e. before the `shown.bs.modal` event occurs).

```
$('#myModal').modal('show')
```

接著在我們 products.vue 中的 methods 加上自定義的 openModal()

裏頭加入剛剛的 bs 提供語法 並給為我們自定義的部分 \$('#productModal').modal('show')

然後 要將 JQ 的 "\$" import 進來 否則 會無法執行 並 噴錯表示 \$ not defined

▼ Products.vue ●

product > src > components > pages > ▼ Products.vue > {} "Products.vue"

```
56  <script>
57  | import $ from 'jquery';
58  |
59  export default {
60  |   data() {
61  |     return {
62  |       products: [] // 所有新增的產品資料會放進這裡
63  |     };
64  |   },
65  |   methods: {
66  |     getProducts() {
67  |       // 取得遠端產品資料 然後將 ajax取得的資料存進 data 的products
68  |       const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/products`; // API 伺服器路徑 + 遠端的自己申請的 api 路徑
69  |       const vm = this;
70  |       this.$http.get(api).then(response => {
71  |         console.log(response.data);
72  |         vm.products = response.data.products; // 將取得的產品資訊塞進 products 給予template 做渲染
73  |       });
74  |     },
75  |     openModal(){
76  |       $('#productModal').modal('show')
77  |     }
78  |   },
79  |   created() {
80  |     this.getProducts(); // 上方的 getProducts 需要在 created 裡呼叫執行
81  |   }
82  |};
83 </script>
```

然後在 template 用之前 vue 的 綁定手法 加上 v-on:click 事件以觸發 openModal 方法

▼ Products.vue ●

```
product > src > components > pages > ▼ Products.vue > {} "Products.vue" > ⓘ template
1   <template>
2   |   <div>
3   |     <div class="text-right mt-4" @click="openModal">
4   |       <button class="btn btn-primary">建立新產品</button>
5   |     </div>
6   |     <table class="table mt-4">
```

透過以上作法便可以讓我們自行判斷 modal 要在何種情形打開 ex 按下按鈕後 等 ajax 取道資料後再開 等等

六角提供新增產品的模板

<https://github.com/hexschool/vue-course-api-wiki/wiki/%E8%AA%B2%E7%A8%8B%E9%83%A8%E5%88%86%E6%A8%A1%E6%9D%BF>

然後針對以上模板 做 v-model 資料綁定 、 其他事項 ,
將資料綁進 data 新建立好的 tempProduct:{} 、 新增 updateProduct 事件

```

Products.vue ×
product > src > components > pages > Products.vue > {} "Products.vue" > script > created
190 |   ...
191   </template>
192
193   <script>
194     import $ from 'jquery';
195
196     export default {
197       data() {
198         return {
199           products: [], // 所有新增的產品資料會放進這裡
200
201           tempProduct:{} // for modal 這裡會存放要送到資料庫 產品資料(透過 post 方法)
202         };
203       },
204       methods: {
205         getProducts() {
206           // 取得遠端產品資料 然後將 ajax取得的資料存進 data 的products
207           const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/products`; // API 伺服器路徑 + 遠端的自己申請的 api 路徑
208           const vm = this;
209           this.$http.get(api).then(response => {
210             console.log(response.data);
211             vm.products = response.data.products; // 將取得的產品資訊塞進 products 紿予template 做渲染
212           });
213         },
214         openModal(){
215           $('#productModal').modal('show') // bs 提供給予控制 modal 的 methods
216         },
217         updateProduct(){
218           }
219         },
220       },
221       created() {
222         this.getProducts(); // 上方的 getProducts 需要在 created 裡呼叫執行
223       }
224     };
225   </script>

```

資料欄位對應參考 API 提供的欄位即可 :

<https://github.com/hexschool/vue-course-api-wiki/wiki/%E7%AE%A1%E7%90%86%E6%8E%A7%E5%88%B6%E5%8F%B0-%5B%E9%9C%80%E9%A9%97%E8%AD%89%5D%#%E5%95%86%E5%93%81%E5%BB%BA%E7%AB%8B>

商品建立

```

[API]: /api/:api_path/admin/product
[說明]: @api_path = 'thisismycourse2'
[方法]: post
[參數]:
{
  "data": {
    "title": "[賣]動物園造型衣服3",
    "category": "衣服2",
    "origin_price": 100,
    "price": 300,
    "unit": "個",
    "image": "test.testtest",
    "description": "Sit down please 名設計師設計",
    "content": "這是內容",
    "is_enabled": 1,
    "imageUrl": <>firebase storage url>
  }
}
[成功回應]:
{
  "success": true,
  "message": "已建立產品"
}

```

將一些 input 、 textarea 欄位資料綁定

```
Products.vue
product > src > components > pages > Products.vue > {} "Products.vue" > template >
  </div>
  <!-- Modal -->
  <div class="modal fade" id="productModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog modal-lg" role="document">
      <div class="modal-content border-0">
        <div class="modal-header bg-dark text-white">
          <h5 class="modal-title" id="exampleModalLabel">
            <span>新增產品</span>
          </h5>
          <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
        </div>
        <div class="modal-body">
          <div class="row">
            <div class="col-sm-4">
              <div class="form-group">
                <label for="image">輸入圖片網址</label>
                <input type="text" class="form-control" id="image" placeholder="請輸入圖片連結" v-model="tempProduct.imageUrl" ><!-->
              </div>
              <div class="form-group">
                <label for="customFile">或 上傳圖片 </label>
                <i class="fas fa-spinner fa-spin"></i>
              </div>
              <input type="file" id="customFile" class="form-control" ref="files">
              <!-->
            </div>
            <div class="col-sm-8">
              <div class="form-group">
                <label for="title">標題</label>
                <input type="text" class="form-control" id="title" placeholder="請輸入標題" v-model="tempProduct.title" ><!-->
              </div>
              <div class="form-row">
                <div class="form-group col-md-6">
                  <label for="category">分類</label>
                  <input type="text" class="form-control" id="category" placeholder="請輸入分類" v-model="tempProduct.category" ><!-->
                </div>
                <div class="form-group col-md-6">
                  <label for="price">單位</label>
                  <input type="unit" class="form-control" id="unit" placeholder="請輸入單位" v-model="tempProduct.unit" ><!-->
                </div>
              </div>
              <div class="form-group col-md-6">
                <label for="origin_price">原價</label>
                <input type="number" class="form-control" id="origin_price" placeholder="請輸入原價" v-model="tempProduct.origin_price" ><!-->
              </div>
              <div class="form-group col-md-6">
                <label for="price">售價</label>
                <input type="number" class="form-control" id="price" placeholder="請輸入售價" v-model="tempProduct.price" ><!-->
              </div>
              <hr>
              <div class="form-group">
                <label for="description">商品描述</label>
                <textarea type="text" class="form-control" id="description" placeholder="請輸入產品描述" v-model="tempProduct.description" ><!-->
              </div>
              <div class="form-group">
                <label for="content">說明內容</label>
                <textarea type="text" class="form-control" id="content" placeholder="請輸入產品說明內容" v-model="tempProduct.content" ><!-->
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
```

input checkbox 透過 :true-value 和 false-value 與 v-model 綁定 來讓 設定的 1 與 0 做切換 (因為 API 是給 1 & 0)

```
<div class="form-group">
  <div class="form-check">
    <input
      class="form-check-input"
      type="checkbox"
      id="is_enabled"
      :true-value="1"
      :false-value="0"
      v-model="tempProduct.is_enabled"><!-->
    <label class="form-check-label" for="is_enabled">
      | 是否啟用
    </label>
  </div>
</div>
</div>
</div>
<div class="modal-footer">
  <button type="button" class="btn btn-outline-secondary" data-dismiss="modal">取消</button>
  <button
    type="button"
    class="btn btn-primary"
    @click="updateProduct">確認</button> <!-->
</div>
</div>
</div>
</div>
<div class="modal fade" id="delProductModal" tabindex="-1" role="dialog"
  aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content border-0">
      <div class="modal-header bg-danger text-white">
        <h5 class="modal-title" id="exampleModalLabel">
          | <span>刪除產品</span>
        </h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          | <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        | 是否刪除 <strong class="text-danger">{{ tempProduct.title }}</strong> 商品(刪除後將無法恢復)。
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-outline-secondary" data-dismiss="modal">取消</button>
        <button
          type="button"
          class="btn btn-danger"
          @click="delProduct"
          >確認刪除</button>
      </div>
    </div>
  </div>
</div>
</div>
</div>
</div>
</template>
```

新增 產品建立按鈕事件 :updateProduct() 依據上方 API 指示使用 post 方法，並且更改符合的 api 路徑。

(API 規則有提及要傳入的參數格式 是 {data: ''})

寫下判斷式 如果 成功推送 就將 modal 隱藏，並且重新 getProducts 一次 (重新渲染畫面商品)

```
methods: {
  getProducts() {
    // 取得遠端產品資料 然後將 ajax取得的資料存進 data 的products
    const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/products`; // API 同伺服器路近 + 遠端的自己申請的 api 路徑
    const vm = this;
    this.$http.get(api).then(response => {
      console.log(response.data);
      vm.products = response.data.products; // 將取得的產品資訊塞進 products 紿予template 做渲染
    });
  },
  openModal(){
    $('#productModal').modal('show') // bs 提供給予控制 modal 的 methods
  },
  updateProduct(){
    const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/product` // API 同伺服器路近 + 遠端的自己申請的 api 路徑
    const vm = this;
    this.$http.post(api, {data:vm.tempProduct}).then(response => {
      if(response.data.success){
        $('#productModal').modal('hide') // bs 提供給予控制 modal 的 methods
        vm.getProducts();
      }else{
        $('#productModal').modal('hide') // bs 提供給予控制 modal 的 methods
        console.log("產品建立失敗")
      }
    });
  },
  created() {
    this.getProducts(); // 上方的 getProducts 需要在 created 裡呼叫執行
  }
};
```

當我們在 `openModal` 當下 就會判定是 建立新產品 或 編輯產品，此兩種情況便分別會是 點開 的 modal 畫面是沒產品 、 有既有產品 (要編輯) 。

編輯 `openModal()`

```
openModal(isNew,item){ // 新增 參數 ( 是否是新的 , item(原有的 item) )  
  if(isNew){ // 如果是新增的時候 tempProduct就是一個空物件  
    this.tempProduct = {};  
    this.isNew = true;  
  }else{ // 否則就是編輯  
    this.tempProduct = Object.assign({},item); // 避免編輯的tempProduct 與 item 有參考的特性 ( 物件傳址特性 )  
    this.isNew = false;  
  }  
  $('#productModal').modal('show') // bs 提供給予控制 modal 的 methods  
},
```

接著 會因為 modal 開啟後 要做的動作不同 所以在 ajax 的 method 、 API 路徑 也不一樣

建立新品 : `post`

商品建立

```
[API]: /api/:api_path/admin/product  
[說明]: @api_path = 'thisismycourse2'  
[方法]: post  
[參數]:  
  {  
    "data": {  
      "title": "[賣]動物園造型衣服3",  
      "category": "衣服2",  
      "origin_price": 100,  
      "price": 300,  
      "unit": "個",  
      "image": "test.testtest",  
      "description": "Sit down please 名設計師設計",  
      "content": "這是內容",  
      "is_enabled": 1,  
      "imageUrl": <>firebase storage url>  
    }  
  }  
[成功回應]:  
  {  
    "success": true,  
    "message": "已建立產品"  
  }
```

編輯產品 : `put`

修改產品

```
[API]: /api/:api_path/admin/product/:id  
[方法]: put  
[說明]: @api_path = 'thisismycourse2'  
        @id = '-L9tH8jxVb2Ka_DYPwng'  
[參數]:  
  {  
    "data": {  
      "category": "衣服3",  
      "content": "這是內容",  
      "description": "Sit down please 名設計師設計",  
      "id": "-L9tH8jxVb2Ka_DYPwng",  
      "image": "test.testtest",  
      "is_enabled": 1,  
      "origin_price": 100,  
      "price": 600,  
      "title": "[賣]動物園造型衣服3",  
      "unit": "個",  
      "num": 1,  
      "imageUrl": <>firebase storage url>  
    }  
  }  
[成功回應]:  
  {  
    "success": true,  
    "message": "已更新產品"  
  }
```

此時 就要對 `updateProduct` 再做更動了

```
updateProduct(){  
  let api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/product` // API 伺服器路徑 + 遠端的自己申請的 api 路徑  
  let httpMethod = 'post'; // 先預設 ajax 方法是 post  
  const vm = this;  
  // 這裡的 isNew 是經由上方 openModal 所決定的結果 來判斷了  
  if(!vm.isNew){ // 如果 不是新的 我們就更改 API 內容 、 http 方法使用 put  
    api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/product/${vm.tempProduct.id}`;  
    httpMethod = 'put';  
  }  
  this.$http[httpMethod](api, {data:vm.tempProduct}).then(response => {  
    if(response.data.success){  
      $('#productModal').modal('hide') // bs 提供給予控制 modal 的 methods  
      vm.getProducts();  
    }else{  
      $('#productModal').modal('hide') // bs 提供給予控制 modal 的 methods  
      vm.getProducts();  
      console.log("產品建立失敗")  
    }  
  });  
}
```

這裡用 `true/ false` 參數帶入 判定是 新增或編輯

分別綁定在兩個 按鈕上

(編輯的要多帶入 `item` 參數) (`item` 就是當下按下的 product)

```
<button  
  class="btn btn-primary"  
  @click="openModal(true)">建立新產品</button>
```

```
<button  
  class="btn btn-outline-primary btn-sm"  
  @click="openModal(false, item)">編輯</button>
```

11 - 95. 串接上傳檔案 API

本案例是 上傳圖片 , 那 後端 是以 表單 <form> 形式做接收 ,

那我們前端 可以再前面 使用 Javascript 方式 模擬 表單傳送 (Web Api) , 將在後端組成一個 form 表單

這裡簡單介紹一下表單傳送 的幾個重點 :

<form> :

Attribute	Value	Description
action	URL	指定提交表單時將表單數據發送到的位置
enctype	application/x-www-form-urlencoded multipart/form-data text/plain	指定將表單數據提交到服務器時應如何編碼 (僅用於 method = “ post”)
method	get post	指定發送表單數據時要使用的 HTTP 方法

屬性 enctype :

enctype	Description
application/x-www-form-urlencoded	默認。所有字符在發送之前都已編碼 (空格轉換為 “ +” 符號 , 特殊字符轉換為 ASCII HEX 值)
multipart/form-data	沒有字符被編碼。當您使用具有文件上載控件的表單時 , 此值是必需的 用於 我們的表單 具有 檔案上傳控制的 需求
text/plain	空格會轉換為 “ +” 符號 , 但不會編碼任何特殊字符

然後對應到 六角 API 級的 表單模擬樣式 如下 是該有的 form 屬性 、 input 屬性 會有哪些
以及會有甚麼要的 值

上傳圖片

上傳表單 (前端測試使用)

```
<form action="/api/thisismycourse2/admin/upload" enctype="multipart/form-data"
      method="post">
    <input type="file" name="file-to-upload">
    <input type="submit" value="Upload">
</form>
```

(欄位名稱 :file-to-upload)

```
action - URL : 指定提交表單時將表單數據發送到的位置
enctype - application/x-www-form-urlencoded : 指定將表單數據提交到服務器時應如何編碼 (僅用於method =“ post”)
          | multipart/form-data
          | text/plain

method - get / post: 指定發送表單數據時要使用的HTTP方法

<form
  action="/api/thisismycourse2/admin/upload"
  enctype="multipart/form-data"
  method="post">
  <input
    type="file"
    name="file-to-upload">
  <input
    type="submit"
    value="Upload">
</form>
```

選擇檔案 未選擇任何檔案

以上 知曉 表單基本 屬性 內容 就先簡單操作一下 上傳檔案

那我們先定義上傳的事件，並綁定 Vue v-on 的 change

```
<div class="form-group">
  <label for="customFile">或 上傳圖片
    <i class="fas fa-spinner fa-spin"></i>
  </label>
  <input
    type="file"
    id="customFile"
    class="form-control"
    ref="files"
    @change="uploadFile" > <!-- 上傳圖片 -->
</div>
```

首先 console 出來 看看 丟進去的圖片檔案

```
  uploadFile(){
    console.log(this)
  }
```

丟入 football 圖片



接著到 console 查看 this 打印出來的結果： `VueComponent > $refs > files > files` (注意：這裡有兩層 files)
便會看到 圖片資源以 陣列方式 顯示

```
▼ VueComponent {_uid: 5, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: VueComponent, ...} ⓘ
  $attrs: (...)
  ▶ $children: []
  ▶ $createElement: f (a, b, c, d)
  ▶ $el: div
  $listeners: (...)
  ▶ $options: {parent: VueComponent, _parentVnode: VNode, propsData: undefined, _parentListeners: undefined}
  ▶ $parent: VueComponent {_uid: 2, _isVue: true, $options: {...}, _renderProxy: Proxy, _self: VueComponent, ...}
  ▶ $refs:
    ▼ files: input#customFile.form-control
      accept: ""
      elementRefName: ""
      elementRefValue: ""
      enterKeyHint: ""
      ▶ files: FileList
        ▶ 0: File {name: "football_PNG52759.png", lastModified: 1589903115650, lastModifiedDate: Tue Mar 10 2020 10:35:50 GMT+0800 (台灣/香港), type: "image/png", size: 100000}
        ▶ length: 1
        ▶ __proto__: FileList
      firstChild: null
      firstElementChild: null
      form: null
```

因此 要上傳的圖片是 以上的第 0 個物件

接著因應剛剛提及 我們要以 JS 來模擬傳統的 HTML 表單形式 做傳送：

可以參考 MDN 文件 提供的 [new FormData\(\)](#)

<https://developer.mozilla.org/en-US/docs/Web/API/FormData/FormData>

(MDN example)

表單傳送 action

```
[API]: /api/:api_path/admin/upload
[方法]: post
[說明]:
  @api_path = 'thisismycourse2'
[成功回傳]:
{
  "success":true,
  "imageUrl": 圖片連結 (https)
}
[錯誤回傳]: 檔案格式錯誤 (非圖片)
{
  "success": false,
  "message": "檔案格式錯誤"
}
[錯誤回傳]: 檔案過大
{
  "success": false,
  "message": {
    "code": "LIMIT_FILE_SIZE",
    "field": "file-to-upload",
    "storageErrors": []
  }
}
```

Example

The following line creates an empty `FormData` object:

```
1 | var formData = new FormData(); // Currently empty
```

You could add a key/value pair to this using `FormData.append`:

```
1 | formData.append('username', 'Chris');
```

然後使用 `new FormData()` 方式 撰寫 `uploadFile()`

```
uploadFile(){
  console.log(this)
  const uploadedFile = this.$refs.files.files[0]; // 要傳入的圖片
  const vm = this;
  const formData = new FormData(); // 透過 JS 先準備一個 Form 單
  formData.append( 'file-to-upload', uploadedFile ) // 再用 append 把欄位新增進去 form 裡面，後方參數就是要傳入的檔案
  const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/upload`;// 定義路徑
  // 以下 post 內容 ( 路徑，剛組好的 formData ，調整好格式的 FormData 物件 )
  this.$http.post( url , formData , {
    headers:{ // 將表單形式改成 FormData
      'Content-Type' : 'multipart/form-data' // multipart/form-data 屬性使用於 我們的表單 具有 檔案上傳控制的需求
    }
  }).then((response)=>{
    console.log(response.data)
  })
}
```

流程總結：

先取出 要傳送的檔案(圖片) >> 建立一個表單物件 `formData` >> 將 `uploadedFile` 加入
>> 透過 `post` 送出 (裏頭依照需求定義 `Content - Type`)

此時 上傳圖片後 可以先從 console 看出此圖片目前已被上傳 且有自己的網址了

The screenshot shows a Vue.js application interface for adding a new product. On the left, there is a form with fields for '輸入圖片網址' (Input Picture URL) containing 'https://storage.googleapis.com...', '標題' (Title) with placeholder '請輸入標題', '分類' (Category) with placeholder '請輸入分類', and '原價' (Original Price) with placeholder '請輸入原價'. Below these is a large image of a soccer ball. On the right, the browser's developer tools show the network tab with a successful POST request to '/include/postCloud.js.map'. The response body is an object with 'success: true', 'products: Array(9)', 'pagination: {}', 'messages: []', and a nested 'success: true' object with 'imageUrl: "https://storage.googleapis.com/vue-course-api.appspot.com/3bJnadsNra%30%3D"'.

以上圖片上傳 已完成 那我們要將他的圖片路徑 (網址) 存下來
且 他會對應到 tempProduct 的 imgurl 讓 表單上傳有效果

這裡有個重點 imgurl 並未包含 get 與 set 所以要強制寫入 imgurl

<https://pjchender.blogspot.com/2017/05/vue-vue-reactivity.html> ←用 set 解說

```
uploadFile(){
  console.log(this)
  const uploadedFile = this.$refs.files.files[0]; // 要傳入的圖片
  const vm = this;
  const formData = new FormData(); // 透過 JS 先準備一個 Form 單
  formData.append('file-to-upload', uploadedFile) // 再用 append 把欄位新增進去 form 裡面，後方參數就是要傳入的檔案
  const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/upload`;// 定義路徑
  // 以下 post 內容 ( 路徑，剛組好的 formData ，調整好格式的 FormData 物件 )
  this.$http.post(url, formData, {
    headers: { // 將表單形式改成 FormData
      'Content-Type': 'multipart/form-data' // multipart/form-data 屬性使用於 我們的表單 具有 檔案上傳控制的 需求
    }
  }).then((response)=>{
    console.log(response.data)
    if(response.data.success){
      // vm.tempProduct.imageUrl = response.data.imageUrl; // 此作法會讓 建立新產品的 圖片位置無法正確雙向綁定 因此要採用以下 set 做法
      // 以下 set 內容 ( 此欄位要塞進哪裡，此欄位名稱，要寫入的路徑 )
      vm.$set(vm.tempProduct, 'imageUrl', response.data.imageUrl); // 使用 set 強制寫入 才有 雙向綁定功能
      console.log(vm.tempProduct)
    }
  })
}
```

The screenshot shows the same Vue.js application interface for adding a new product. The 'imageUrl' field in the form has been highlighted with a red box. The browser's developer tools on the right show the state of the 'tempProduct' object, which now includes a 'imageUrl' property with the value 'https://storage.googleapis.com...'. The developer tools also show some warning messages about cookies and DevTools failing to load SourceMaps.

11-96. 增加使用者體驗 - 讀取中效果製作

製作 網頁載入、圖片載入的 防呆 Loading 讀取畫面

使用 vue-loading-overlay 套件 製作 全頁的 loading 效果

<https://github.com/ankurk91/vue-loading-overlay>

npm install vue-loading-overlay –save

安裝後 將相關套件設定 寫在 main.js

```
V Products.vue • JS main.js •  
product > src > JS main.js > ...  
1 import Vue from 'vue';  
2 import axios from 'axios';  
3 import VueAxios from 'vue-axios';  
4 import 'bootstrap';  
5 import Loading from 'vue-loading-overlay'; // Import component  
6 import 'vue-loading-overlay/dist/vue-loading.css'; // Import stylesheet  
7  
8  
9 import App from './App'  
10 import router from './router'  
11  
12 Vue.use(VueAxios, axios)  
13 Vue.config.productionTip = false  
14  
15 | Vue.component('Loading', Loading) // 使用全域方式啟用 (這樣在每個不同地方就就不需要依依載入)  
16
```

首先在 template 部分加入 官方提供的 tag

```
V Products.vue × JS main.js  
product > src > components > pages > V Products.vue > {} "Products.vue" > ⚡ script > ⚡ data  
1 <template>  
2   <div>  
3     <div class="text-right mt-4">  
4       <loading :active.sync="isLoading" > </loading>  
5     <button  
6       class="btn btn-primary"  
7       @click="openModal(true)">建立新產品</button> <!-- 這裡用 true 參數帶入 判定是 新增 -->  
8   </div>
```

接著 在 data 寫入 預設是 false

isLoading 如果是 true 就會啟用 loading 防呆效果

在 getProducts 取得 ajax 以前給予 true(loading 效果), 取得後就移除 (false)

```
methods: {  
  getProducts() {  
    // 取得遠端產品資料 然後將 ajax取得的資料存進 data 的 products  
    const api = `${process.env.APIPATH}/api/${process.env.VUE_APP_COLLECTION}`;  
    const vm = this;  
    vm.isLoading = true;  
    this.$http.get(api).then(response => {  
      console.log(response.data);  
      vm.isLoading = false;  
      vm.products = response.data.products; // 將取得的資料存進 products  
    });  
  },  
  data() {  
    return {  
      products: [], // 所有新增的產品資料會放進這裡  
      tempProduct: {}, // for modal 這裡會存放要送到後端的資料  
      isNew: false,  
      isLoading: false, // loading 畫面 啟用與否  
    };  
  },
```

效果如下 (中間會有個轉圈符號)

The screenshot shows a dashboard interface with a sidebar on the left containing links like Company name, Dashboard, Orders, Products, Customers, Reports, Integrations, SAVED REPORTS, Current month, Last quarter, Social engagement, and Year-end sale. The main area has a search bar at the top. Below it is a table with columns: 分類 (Category), 產品名稱 (Product Name), 原價 (Original Price), and 售價 (Sale Price). There are three rows of data: '新增產品' (New Product) under '測試分類' (Test Category), '測試的產品' (Test Product) under '測試分類' (Test Category), and another '測試的產品' (Test Product) under '測試分類' (Test Category). A large play button icon with a red box around its outline is overlaid on the screen, indicating a loading or processing state.

接著 製作局部位置的 loading 效果 , 我們使用 fontawesome 製作
以下直接使用 CDN 引入 方式

The screenshot shows a code editor with tabs for Products.vue, index.html, and main.js. The index.html tab is selected and displays the following code:

```
product > index.html > ...
1  <!DOCTYPE html>
2  <html>
3  | <head>
4  | | <meta charset="utf-8">
5  | | <meta name="viewport" content="width=device-width,initial-scale=1.0">
6  | | <title>product</title>
7  | | <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.13.1/css/all.min.css"></i>
8  |
9  | </head>
10 | <body>
11 | | <div id="app"></div>
12 | | <!-- built files will be auto injected -->
13 | </body>
14 </html>
```

The line `<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.13.1/css/all.min.css"></i>` is highlighted with a yellow box.

```
ue > {} "Products.vue" > template > div > div#productModal.modal.fade >
68
69
70
71
72
73
74
```

或 上傳圖片

</label>

```
/ export default {
/   data() {
/     return {
/       products: [], // 所有新增的產
/       tempProduct:{}, // for modal
/       isNew : false,
/       isLoading : false, // loading
/       status:{
/         fileUploading : false,
/       }
/     };
/   },
/ }
```

```
uploadFile(){
  console.log(this)
  const uploadedFile = this.$refs.files.files[0]; // 要傳入的圖片
  const vm = this;
  const formData = new FormData(); // 透過 JS 先準備一個 Form 單
  formData.append( 'file-to-upload', uploadedFile ) // 再用 append 把欄位新增進去 form 裡面，後方參數就是要傳入的檔案
  const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/upload`;// 定義路徑
  vm.status.fileUploading = true ;
  // 以下 post 內容（路徑，剛組好的 formData，調整好格式的 FormData 物件）
  this.$http.post( url , formData , {
    headers:{ // 將表單形式改成 FormData
      'Content-Type' : 'multipart/form-data' // multipart/form-data 屬性使用於 我們的表單 具有 檔案上傳控制的 需求
    }
  }).then((response)=>{
    console.log(response.data)
    vm.status.fileUploading = false ;
    if(response.data.success){
      // vm.tempProduct.imageUrl = response.data.imageUrl; // 此作法會讓 建立新產品的 圖片位置無法正確雙向綁定 因此要採用以下 set 做法
      // 以下 set 內容（此欄位要塞進哪裡，此欄位名稱，要寫入的路徑）
      vm.$set(vm.tempProduct, 'imageUrl', response.data.imageUrl); // 使用 set 強制寫入 才有 雙向綁定功能
      console.log(vm.tempProduct)
    }
  })
}
```



11 - 98. 增加使用者體驗 - 錯誤的訊息回饋

由於 alert 是比較頂層的畫面 那一般來說 我們撰寫的 component 都在底層 因此比較難呼叫，所以可以透過 Vue Event Bus 概念 直接將 Alert 掛載在 Vue 原型之下，透過直接操作 Vue 的原型來控制 alert

Event Bus 簡介：

我們都知道 vuex 可以幫助我們統一管理狀態，但是若情況沒有這麼複雜或專案規模不大的時候，是可以考慮使用 event bus 來作為組件溝通的橋樑。

前言

使用 event bus 來作為溝通橋樑的概念就像是所有組件共用相同的事件中心，可以向該中心註冊發送事件或接收事件，所有組件都可以上下平行地通知其他組件，但也就是太方便所以若不謹慎使用，會造成難以維護的災難產生，因此才需要更完善的 vuex 作為狀態管理中心，將通知的概念提升到共享狀態層次。說了這麼多，如果網站規模不大不複雜，共用狀態或多階組件互動其實並沒有這麼多的時候也可以先從 event bus 著手，因此本文就來學習一下怎麼實作 event bus 機制吧。

先抓取 六角提供的 alert 原始碼

<https://github.com/hexschool/vue-course-api-wiki/wiki/%E8%AA%B2%E7%A8%8B%E9%83%A8%E5%88%86%E6%A8%A1%E6%9D%BF>

The screenshot shows a code editor with a sidebar showing the project structure. The main pane displays the code for `AlertMessage.vue`. The code includes a template with a button to remove a message, a script section defining a data object with a `messages` array containing a single message object, and methods for updating and removing messages. A yellow box highlights the `messages` array in the data section.

```
product > src > components > AlertMessage.vue > {} "AlertMessage.vue" > template
  6   <button type="button" class="close" @click="removeMessage(i)" aria-label="Close">
  7     <span aria-hidden="true">&times;
```

接著到 Dashboard.vue 引入 該 元件

```
product > src > components > Dashboard.vue > {} "Dashboard."
1  <template>
2  |   <div>
3  |   |     <Navbar></Navbar>      <!-- Navbar -->
4  |   |     <Alert/>           <!-- Alert -->
5  |   |     <div class="container-fluid">
6  |   |       <div class="row">
7  |   |         <Sidebar></Sidebar> <!-- Sidebar -->
8  |   |         <main role="main" class="col-md-9 ml-2">
9  |   |           <router-view></router-view>
10 |   |         </main>
11 |   |       </div>
12 |   |     </div>
13 |   |   </div>
14 |   | </template>
15 |
16 <script>
17   import Sidebar from "./Sidebar";
18   import Navbar from "./Navbar";
19   import Alert from "./AlertMessage"
20
21   export default {
22     components: {
23       Sidebar,
24       Navbar,
25       Alert
26     }
27   };
28
29 </script>
```

然後啟用 Event Bus：直接在 src 底下建立一個 bus.js 檔案

並且直接掛載在 Vue 的原型之下 用一個叫 \$bus 的變數（如此便能直接對 event bus 做呼叫）

```
product > src > JS bus.js
1  import Vue from 'vue';
2
3  Vue.prototype.$bus = new Vue();
```

完成後 把 event bus 注入到 main.js 裡面

```
JS main.js > {}
product > src > JS main.js > router.beforeEach() callback
6   import 'vue-loading-overlay/dist/vue-loading.css'; // Import stylesheet
7
8
9   import App from './App';
10  import router from './router';
11  import "./bus";
12
13  Vue.use(VueAxios, axios)
14  Vue.config.productionTip = false
```

這裡的 `vm.$bus` 是直接呼叫 Vue 實體下的 bus (剛剛創建的 `bus.js`)，並使用 `on` 方式 註冊一個 `mes:push` 的方法

`$on` 作用便是要推資料進去上方宣告 `data` 的 `messages` 裡頭：

- `message` 是一個字串對應到 上方 `messages` 裡的 `message`
- `status` 便是對應到 `messages` 裡的 `bootstrap` 樣式 - 下方我們先行定義預設是 `warning`(黃色)

The screenshot shows the file structure on the left and the code for `AlertMessage.vue` on the right. The code is as follows:

```
product > src > components > AlertMessage.vue > {} "AlertMessage.vue"
46     if (item.timestamp === timestamp) {
47         vm.messages.splice(i, 1);
48     });
49 },
50 }, 5000);
51 },
52 },
53 created() {
54     const vm = this;
55     // 自定義名稱 'messsage:push'
56     // message: 傳入參數
57     // status: BS 樣式，預設值為 warning
58     vm.$bus.$on('message:push', (message, status = 'warning') => {
59         vm.updateMessage(message, status); // 最後在觸發 更新 mes
60     });
61     vm.$bus.$emit('message:push');
62 },
63 };
```

A yellow box highlights the `vm.$bus.$on` and `vm.$bus.$emit` lines at the bottom of the `created` block.

接著套 `product` 頁面，直接在 `created` 的地方 嘗試送出一個 event bus (先送出 剛剛的 `message:push`，並給他訊息內容 和 狀態(bs 樣式))，透過此方式 觸發看看 外層的 alert 提示

`this.$bus.$emit('message:push','這是一段訊息','success');` 便是 event bus 的使用方法

The screenshot shows the file structure on the left and the code for `Products.vue` on the right. The code is as follows:

```
product > src > components > pages > Products.vue > {} "Products.vue"
321 }
322 },
323 created() {
324     this.getProducts(); // 上方的 getProducts 需要在 created 之後執行
325     this.$bus.$emit('message:push','這是一段訊息','success');
326 }
327 };
328 </script>
```

A yellow box highlights the `this.$bus.$emit` line in the `created` block. To the right, a browser screenshot shows a yellow alert box with the text "這是一段訊息".

如此 便達成 外層使用 `$on` 註冊好，然後 內層則使用 `$emit` 觸發它

The screenshot shows two code editors. The left editor shows `ProductList.vue` with the following code:

```
ProductList.vue > ...
<script>
  ...
  created(){
    this.getProducts();
    this.$bus.$emit('message:push','這是一段訊息','success')
  }
</script>
```

The right editor shows `AlertMessage.vue` with the following code:

```
src > components > admin > AlertMessage.vue > {} "AlertMessage.vue"
53 created() {
54     const vm = this;
55     // 自定義名稱 'messsage:push'
56     // message: 傳入參數
57     // status: BS 樣式，預設值為 warning
58     vm.$bus.$on('message:push', (message, status = 'warning') => {
59         vm.updateMessage(message, status); // 最後在觸發 更新 mes
60     });
61     vm.$bus.$emit('message:push');
62 };
63 </script>
```

A yellow arrow points from the `this.$bus.$emit` line in `ProductList.vue` to the `vm.$bus.$on` line in `AlertMessage.vue`.

最後 就將以上 寫到 我們真正要 提示用戶的地方

以下 成功就將網址 照先前一樣 正確地將網址寫進 tempProduct 裡面

失敗 則透過 bus 跳出錯誤訊息提醒用戶 其錯誤原因

```
product > src > components > Products.vue > {} "Products.vue" > script > methods > uploadFile > headers
uploadFile(){
    console.log(this)
    const uploadedFile = this.$refs.files.files[0]; // 要傳入的圖片
    const vm = this;
    const formData = new FormData(); // 透過 JS 先準備一個 Form 範本
    formData.append('file-to-upload', uploadedFile) // 再用 append 把欄位新增進去 form 裡面，後方參數就是值
    const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/upload`;// 定義路徑
    vm.status.fileUploading = true ;
    // 以下 post 內容 ( 路徑 , 傳好的 formData , 調整好格式的 FormData 物件 )
    this.$http.post( url , formData , {
        headers:[
            'Content-Type' : 'multipart/form-data'
        ]
    }).then((response)=>{
        console.log(response.data)
        vm.status.fileUploading = false ;
        if(response.data.success){
            // vm.tempProduct.imageUrl = response.data.imageUrl; // 此作法會讓 建立新產品的 圖片位置無法正確顯示
            // 以下 set 內容 ( 此欄位要塞進哪裡 , 此欄位名稱 , 要寫入的路徑 )
            vm.$set(vm.tempProduct, 'imageUrl', response.data.imageUrl); // 使用 set 強制寫入 才有 雙向綁定功能
            console.log(vm.tempProduct)
        }else{
            this.$bus.$emit('message:push',response.data.message,'danger'); // 如果檔案有誤 跳錯誤訊息
        }
    })
},
created() {
    this.getProducts(); // 上方的 getProducts 需要在 created 裡呼叫執行
}
};
```

並且 刪除掉 剛剛在 AlertBus.js 寫的預設值

```
product > src > components > AlertMessage.vue > {} "AlertMessage.vue" > script > methods
</template>
<script>
export default {
    name: 'Navbar',
    data() {
        return {
            messages: [
                // message :'訊息內容',
                // status :'danger', // bs 樣式
                // timestamp :123,
            ],
        };
    },
    // 每當送訊息到 上方 data 的 mes 裡面時 就會觸發函式 將 自己移除的函式
    methods: {
        updateMessage(message, status) {
            const timestamp = Math.floor(new Date() / 1000);
            this.messages.push({
                message,
                status,
                timestamp,
            });
        }
    }
};
```

11 - 99. 產品列表的分頁邏輯

基本上 分頁 邏輯六角後端 API 已經幫我們寫好了

Pagination – 分頁

取得商品列表

```
[API]: /api/:api_path/admin/products?page=:page
[方法]: get
[說明]:
  @api_path = 'thisismycourse2'
  @page 當前第幾頁(分頁參數)
[成功回應]:
{
  "success": true,
  "products": [
    {
      "category": "衣服2",
      "content": "這是內容",
      "description": "Sit down please 名設計師設計",
      "id": "-L9tH8jxVb2Ka_DYPwng",
      "image": "test.testtest",
      "is_enabled": 1,
      "origin_price": 100,
      "price": 600,
      "title": "[賣]動物園造型衣服3",
      "unit": "個",
      "num": 1,
      "imageUrl": <>firebase storage url>
    }
  ],
  "pagination": {
    "total_pages": 1,
    "current_page": 1,
    "has_pre": false,
    "has_next": false,
    "category": null
  },
  "messages": []
}
```

可以在 console 裡看到

(目前預設 頁面只會出現 10 筆資料)

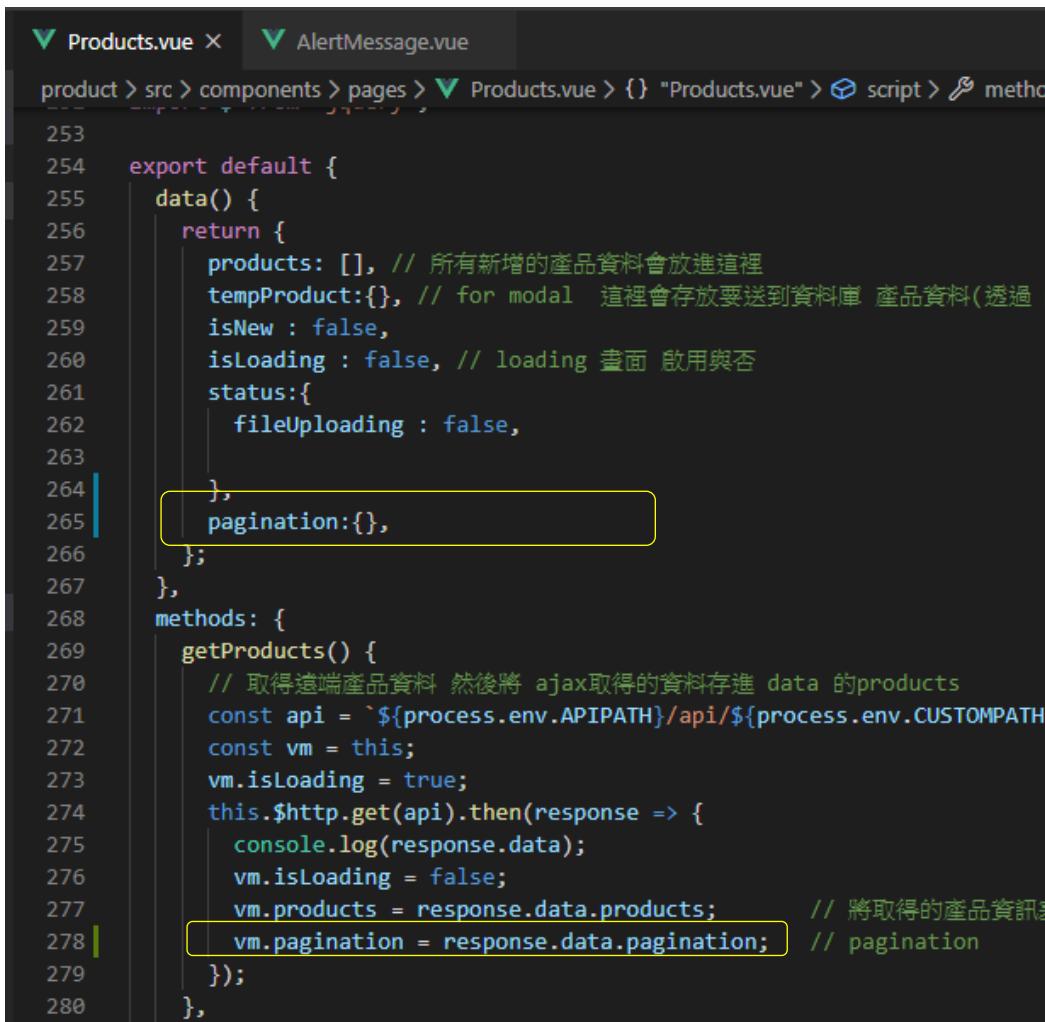
分類	產品名稱	數量	原價	售價	狀態	編輯	刪除
bbbbbbbb		1			未啟用	<button>編輯</button>	<button>刪除</button>
gggggggggggggg		2			未啟用	<button>編輯</button>	<button>刪除</button>
33333		3			未啟用	<button>編輯</button>	<button>刪除</button>
3f3f3f3f3f		4			未啟用	<button>編輯</button>	<button>刪除</button>
g3g333d3d		5			未啟用	<button>編輯</button>	<button>刪除</button>
33f3f3f		6			未啟用	<button>編輯</button>	<button>刪除</button>
ddddd		7			未啟用	<button>編輯</button>	<button>刪除</button>
sqssq		8			未啟用	<button>編輯</button>	<button>刪除</button>
		9			未啟用	<button>編輯</button>	<button>刪除</button>
xxxx		10			未啟用	<button>編輯</button>	<button>刪除</button>

```
建立新產品
VueComponent { _uid: 11, _isVue: true, $options: {<...>}, _renderProxy: Proxy, _self: VueComponent, __proto__: Object } Products.vue?bdef:312
  {success: true, imageUrl: "https://storage.googleapis.com/vue-course-opt.appspot.com/3b0CDFx4DnFEnVXNbteqIVKnW1cxTsuDeelshaQ%3D%3D?"} Products.vue?bdef:318
  ▲ Resource interpreted as Document but transferred with MIME type application/vnd.openxmlformats-officedocument.spreadsheetml.sheet: "file:///C:/Users/user/Desktop/TransactionHistory.xlsx".
Products.vue?bdef:299
  > VueComponent { _uid: 11, _isVue: true, $options: {<...>}, _renderProxy: Proxy, _self: VueComponent, __proto__: Object } Products.vue?bdef:312
  > {success: false, message: "檔案格式錯誤"} Products.vue?bdef:244
  > {success: true, products: Array(5), pagination: {<...>}, messages: Array(0)} Products.vue?bdef:244
  > {success: true, products: Array(6), pagination: {<...>}, messages: Array(0)} Products.vue?bdef:244
  > {success: true, products: Array(7), pagination: {<...>}, messages: Array(0)} Products.vue?bdef:244
  > {success: true, products: Array(8), pagination: {<...>}, messages: Array(0)} Products.vue?bdef:244
  > {success: true, products: Array(9), pagination: {<...>}, messages: Array(0)} Products.vue?bdef:244
  > {success: true, products: Array(10), pagination: {<...>}, messages: Array(0)} Products.vue?bdef:244
  ▼ {success: true, products: Array(10), pagination: {<...>}, messages: Array(0)}
  > messages: []
  > pagination:
    category: null
    current_page: 1
    has_next: true
    has_pre: false
    total_pages: 2
    > __proto__: Object
  > products: (10) [{<...>}, {<...>}, {<...>}, {<...>}, {<...>}, {<...>}, {<...>}, {<...>}, {<...>}, {<...>}]
  > success: true
  > __proto__: Object
Products.vue?bdef:244
```

接著來實作 分頁特效：

首先在 Products.vue 頁面 data 部分 創建空物件 pagination 等等會從後端取得分頁資訊 在塞進去 提供 上方 vue 做 template 綁定渲染的動作

分頁部分 放置在 getProduct 後取得頁數 資料



```
253
254  export default {
255    data() {
256      return {
257        products: [], // 所有新增的產品資料會放進這裡
258        tempProduct:{}, // for modal 這裡會存放要送到資料庫 產品資料(透過 productForm)
259        isNew : false,
260        isLoading : false, // loading 畫面 啟用與否
261        status:{},
262        fileUploading : false,
263      },
264      pagination:{},
265    };
266  },
267  methods: {
268    getProducts() {
269      // 取得遠端產品資料 然後將 ajax取得的資料存進 data 的products
270      const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}`
271      const vm = this;
272      vm.isLoading = true;
273      this.$http.get(api).then(response => {
274        console.log(response.data);
275        vm.isLoading = false;
276        vm.products = response.data.products; // 將取得的產品資訊塞進 products
277        vm.pagination = response.data.pagination; // pagination
278      });
279    },
280  },

```

Template 部分 可以先到 Bootstrap 下載官方的 分頁板型 .

<https://getbootstrap.com/docs/4.0/components/pagination/>

```
<nav aria-label="Page navigation example">
  <ul class="pagination">
    <li class="page-item">
      <a class="page-link" href="#" aria-label="Previous">
        <span aria-hidden="true">&laquo;</span>
        <span class="sr-only">Previous</span>
      </a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item"><a class="page-link" href="#">2</a></li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#" aria-label="Next">
        <span aria-hidden="true">&raquo;</span>
        <span class="sr-only">Next</span>
      </a>
    </li>
  </ul>
</nav>
```

套進 template 並綁上 我們要的資料綁定 :

目前 先完成 當前頁面(active) 、 上下頁 可否點擊 (disabled 但還無效果)

```
<nav aria-label="Page navigation example">
  <ul class="pagination">
    <li>
      <div class="page-item">
        <div :class="{'disabled': pagination.has_pre === false}">
          <a class="page-link" href="#" aria-label="Previous">
            <span aria-hidden="true">&laquo;</span>
            <span class="sr-only">Previous</span>
          </a>
        </div>
      </li>
    <li>
      <div class="page-item">
        <div v-for="page in pagination.total_pages" :key="page" :class="{'active': pagination.current_page === page}">
          <a class="page-link" href="#">{{page}}</a>
        </div>
      </li>
    <li>
      <div class="page-item">
        <div :class="{'disabled': pagination.has_next === false}">
          <a class="page-link" href="#" aria-label="Next">
            <span aria-hidden="true">&raquo;</span>
            <span class="sr-only">Next</span>
          </a>
        </div>
      </li>
    </ul>
  </nav>
```

目前頁面超過十筆資料會自動分到下一页

分類	產品名稱	數量	原價	售價	狀態	編輯	刪除
	bbbbbbbb	1			未啟用	<button>編輯</button>	<button>刪除</button>
	9999999999999999	2			未啟用	<button>編輯</button>	<button>刪除</button>
	33333	3			未啟用	<button>編輯</button>	<button>刪除</button>
	3f3f3f3f3f	4			未啟用	<button>編輯</button>	<button>刪除</button>
	g3g333d3d	5			未啟用	<button>編輯</button>	<button>刪除</button>
	33f3f3f	6			未啟用	<button>編輯</button>	<button>刪除</button>
	ddddd	7			未啟用	<button>編輯</button>	<button>刪除</button>
	sqssq	8			未啟用	<button>編輯</button>	<button>刪除</button>
		9			未啟用	<button>編輯</button>	<button>刪除</button>
	xxxx	10			未啟用	<button>編輯</button>	<button>刪除</button>

增添切換頁 效果 :

先閱讀 API 文件 提及 如需有切換頁效果 就使用此 ?page=:page 帶入參數方法

取得訂單列表

[API]: /api/:api_path/admin/orders?page=:page

[說明]：建立訂單從客戶購物進行下單

```
@api_path = 'thisismycourse2'
```

@page 當前第幾頁(分頁參數)

[方法]: get

[成功回傳]：

f

```
"success": true,
```

"orders": [

f

```
"create_at": 1523539834,  
"id": "-L9u2EUkQSoEmW7QzGLF",  
"is_paid": true,  
"message": "這是留言",  
"update_at": 1523539834
```

使用 ES6 提供的參數預設值 解決其他有呼呼叫到 `getProducts` 方法卻舖需要帶入 `page` 這個參數的問題
(這樣此次的參數添加 就不會對之前 `code` 有影響了)

參數 紿予 page 預設值是 1 如果有代數值就使用自代數值，沒有則帶入第一頁

```
Products.vue ●  
product > src > components > pages > Products.vue > {} "Products.vue" > script > data > status  
8  
9 },  
10 methods: {  
11     getProducts(page = 1) { // 給予page 預設值是 1 如果有代數值就使用自代數值 , 沒  
12         // 取得遠端產品資料 然後將 ajax取得的資料存進 data 的products  
13         const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/products?page=${page}`;  
14         const vm = this;  
15         vm.isLoading = true;  
16         this.$http.get(api).then(response => {  
17             console.log(response.data);  
18             vm.isLoading = false;  
19             vm.products = response.data.products; // 將取得的產品資訊塞進 products 紿予template 做渲染  
20             vm.pagination = response.data.pagination; // pagination  
21         });  
22     },
```

對應到的 template

```
<nav aria-label="Page navigation example">
  <ul class="pagination">
    <li>
      <a href="#" class="page-link" @click.prevent="getProducts(pagination.current_page - 1)">
        <span aria-hidden="true">&laquo;</span>
        <span class="sr-only">Previous</span>
      </a>
    </li>
    <li v-for="page in pagination.total_pages" :key="page" :class="{ 'active': pagination.current_page === page }">
      <a href="#" class="page-link" @click.prevent="getProducts(page)">{{page}}</a>
    </li>
    <li>
      <a href="#" class="page-link" @click.prevent="getProducts(pagination.current_page + 1)">
        <span aria-hidden="true">&raquo;</span>
        <span class="sr-only">Next</span>
      </a>
    </li>
  </ul>
</nav>
```

以上完成後 分頁效果 即可正常運作

		操作	操作
33f3f3f	6	未啟用	<button>編輯</button> <button>刪除</button>
ddddddd	7	未啟用	<button>編輯</button> <button>刪除</button>
sqssq	8	未啟用	<button>編輯</button> <button>刪除</button>
	9	未啟用	<button>編輯</button> <button>刪除</button>
xxxx	10	未啟用	<button>編輯</button> <button>刪除</button>

分頁效果示例：

「<」、「1」、「2」、「>」、「>>」

11 - 100. 套用價格的 Filter 技巧

使用 filter 加上 “\$” 字號

套用六角 寫好的 filter 程式碼 將它存在 filters > currency.js (此程式作用是將數值轉為千分號 、 加上 \$ 字號)

The screenshot shows a code editor with the following file structure:

- product > src > filters > JS currency.js > ...
- JS currency.js (selected)
- JS main.js
- JS products.vue
- JS main.js
- PRODUCT
- product
- build
- config
- dist
- node_modules
- src
- assets
- components
- pages
- Login.vue
- Products.vue
- AlertMessage.vue
- Dashboard.vue
- Index.vue
- Navbar.vue
- Sidebar.vue
- filters
- JS currency.js
- App.vue

The content of currency.js is:

```
// 將金錢數字轉為千分號與加上 $ 字號
export default function (num) {
  const n = Number(num);
  return `$$${n.toFixed(0).replace(/./g, (c, i, a) => {
    const currency = (i && c !== '.' && ((a.length - i) % 3 === 0) ? ` ${c}`.replace(/\s/g, '') : c);
    return currency;
  })}`;
}
```

並且將它引入在全域

The screenshot shows the main.js file with the following content:

```
JS main.js X JS currency.js ● V Products.vue
product > src > JS main.js > router.beforeEach() callback

1 import Vue from 'vue';
2 import axios from 'axios';
3 import VueAxios from 'vue-axios';
4 import 'bootstrap';
5 import Loading from './vue-loading-overlay'; // Import component
6 import 'vue-loading-overlay/dist/vue-loading.css'; // Import stylesheet
7
8
9 import App from './App';
10 import router from './router';
11 import "./bus";
12 import currencyFilter from './filters/currency';
13
14 Vue.use(VueAxios, axios)
15 Vue.config.productionTip = false
16
17 Vue.component('Loading', Loading) // 使用全域方式啟用 (這樣在每個不同地方就可直接使用)
18 Vue.filter('currency', currencyFilter)
19
20 axios.defaults.withCredentials = true;
21
22 /* eslint-disable no-new */
23 new Vue({
24   el: '#app',
25   router,
26   components: { App },
27   template: '<App/>'
28 })
29
30 router.beforeEach((to, from, next) => { // 加入導航守衛 beforeEach
  if (to.meta.requiresAuth) {
    if (!localStorage.getItem('token')) {
      next({ name: 'Login' });
    } else {
      next();
    }
  } else {
    next();
  }
})
```

最後 只需要在需要套用此 filter 效果的地方加上 currency 即可套用

JS main.js JS currency.js ● Products.vue X

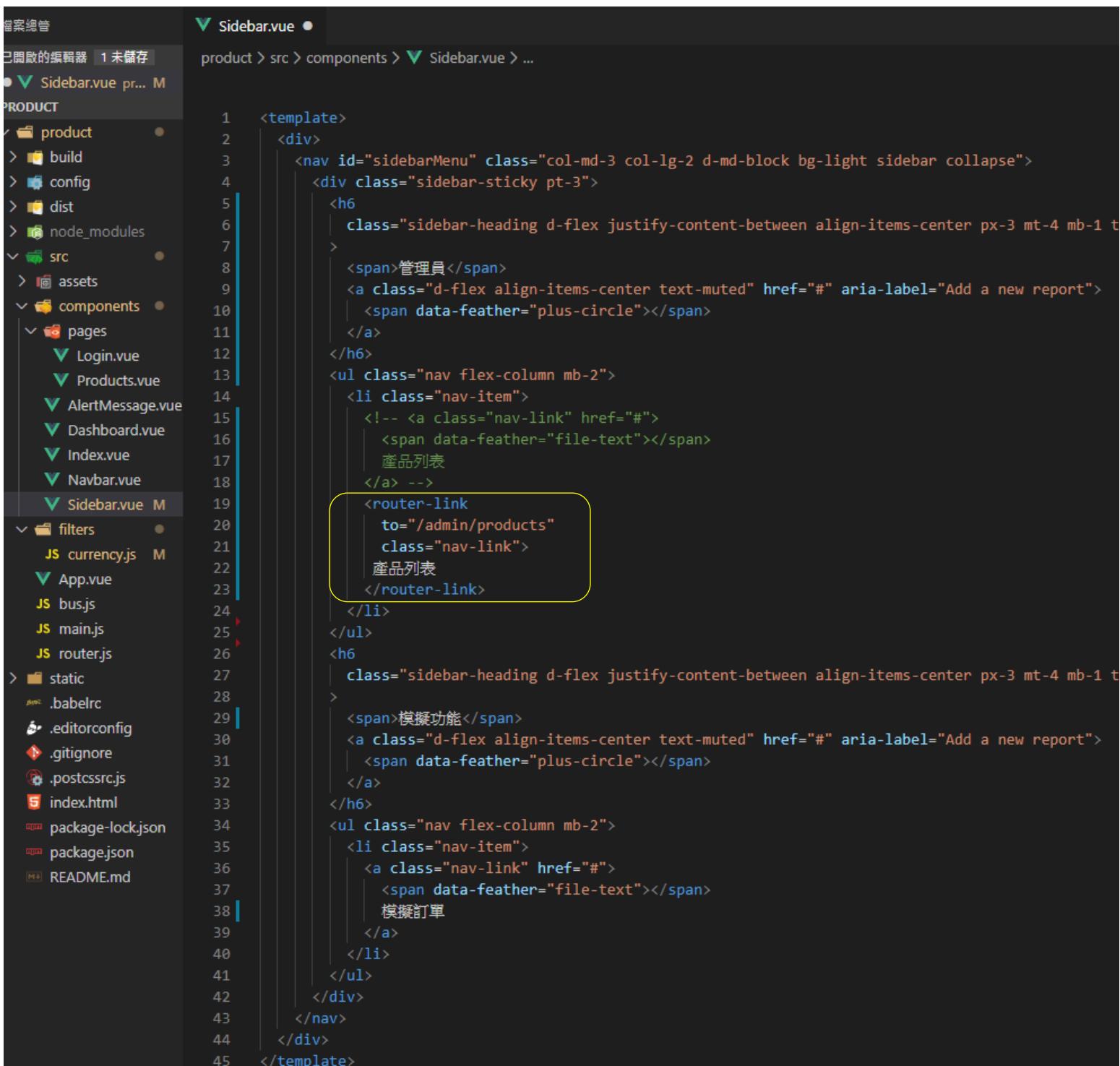
```
product > src > components > pages > Products.vue > {} "Products.vue" > template > div > table.table.mt-4 > tbody
  19   </thead>
  20   <tbody>
  21     <tr
  22       v-for="item in products"
  23         :key="item.id">
  24           <td>{{item.category}}</td>
  25           <td>{{item.title}}</td>
  26           <td>{{item.num}}</td>
  27           <td class="text-right">
  28             {{item.origin_price | currency}}
  29           </td>
  30           <td class="text-right">
  31             {{item.price | currency}}
  32           </td>
  33           <td>
  34             <span
  35               v-if="item.is_enabled">
```

分類	產品名稱	數量	原價	售價	狀態	編輯	刪除
攸	啟用測試	1	\$123	\$444	啟用	<button>編輯</button>	<button>刪除</button>
	bbbbbbbbbb	2	\$23,232,323	\$4,444	未啟用	<button>編輯</button>	<button>刪除</button>

11-101. 中場休息說明，準備進入下半場囉

將 sidebar template 設定成符合我們需要的 頁面

目前先切出產品列表 並給予 router link (管理員產品列是需要登入的狀態)



```
<template>
  <div>
    <nav id="sidebarMenu" class="col-md-3 col-lg-2 d-md-block bg-light sidebar collapse">
      <div class="sidebar-sticky pt-3">
        <h6>
          |   class="sidebar-heading d-flex justify-content-between align-items-center px-3 mt-4 mb-1 t
        >
          |   <span>管理員</span>
          |   <a class="d-flex align-items-center text-muted" href="#" aria-label="Add a new report">
          |       |   <span data-feather="plus-circle"></span>
          |   </a>
        </h6>
        <ul class="nav flex-column mb-2">
          <li class="nav-item">
            |   <!-- <a class="nav-link" href="#">
            |       |   <span data-feather="file-text"></span>
            |       |   產品列表
            |   </a> -->
            |   <router-link
            |       to="/admin/products"
            |       class="nav-link">
            |       產品列表
            |   </router-link>
          </li>
        </ul>
        <h6>
          |   class="sidebar-heading d-flex justify-content-between align-items-center px-3 mt-4 mb-1 t
        >
          |   <span>模擬功能</span>
          |   <a class="d-flex align-items-center text-muted" href="#" aria-label="Add a new report">
          |       |   <span data-feather="plus-circle"></span>
          |   </a>
        </h6>
        <ul class="nav flex-column mb-2">
          <li class="nav-item">
            |   <a class="nav-link" href="#">
            |       |   <span data-feather="file-text"></span>
            |       |   模擬訂單
            |   </a>
          </li>
        </ul>
      </div>
    </nav>
  </div>
</template>
```

Sidebar 畫面目前長這樣：

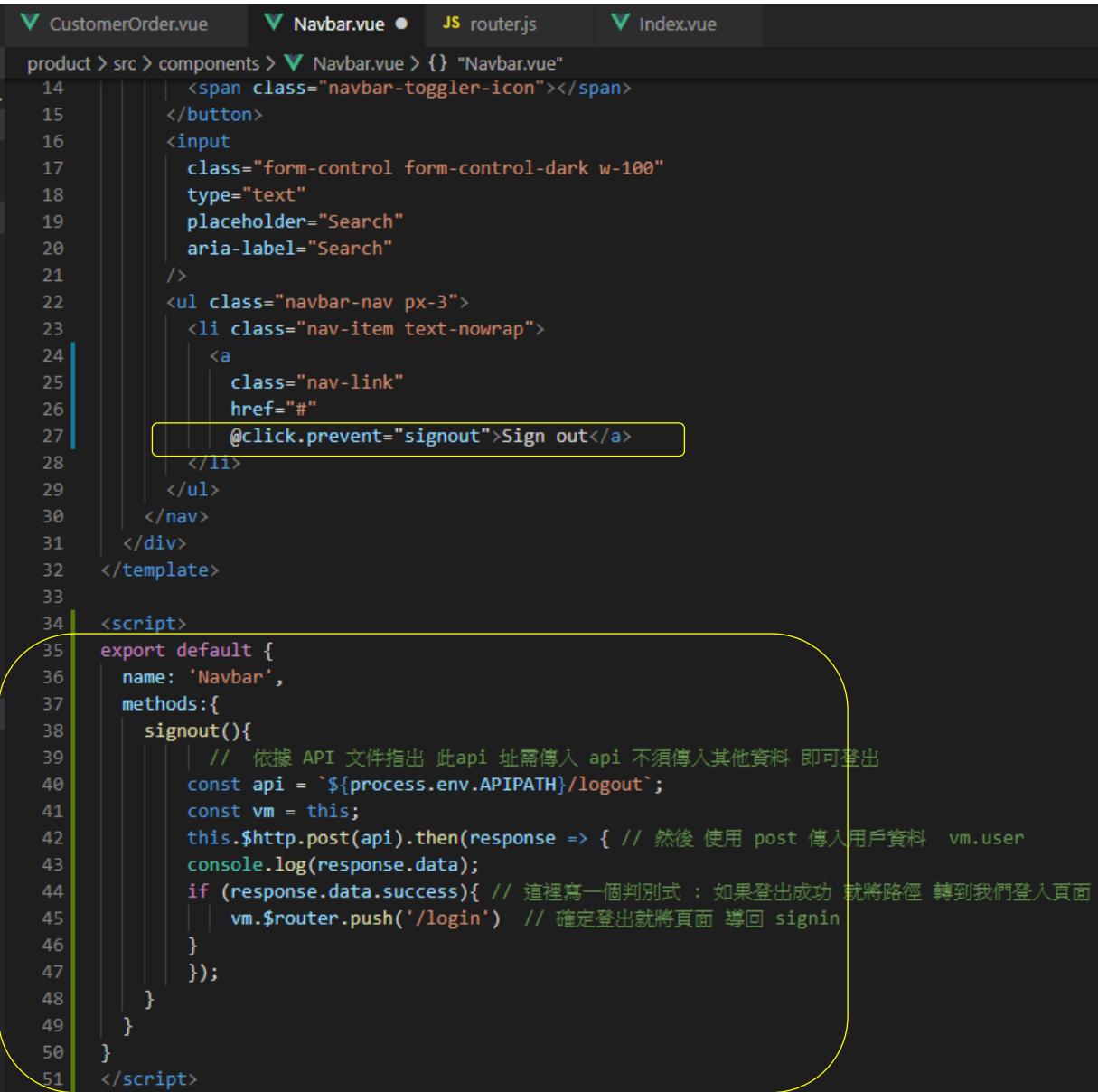


分類	產品名稱
启用测试	bbbbbbbb

此章節要實作模擬訂單 的產品卡片 畫面

首先先補上 登出 功能

在 navbar 部分補上登出



```

檔案總管          V CustomerOrder.vue          V Navbar.vue          JS router.js          V Index.vue
已開啟的編輯器 1 未儲存
CustomerOrder.vue product>src>components>Navbar.vue
Navbar.vue product\src\components\Navbar.vue
router.js product\src
Index.vue product\src\com...
PRODUCT
product
build
config
dist
node_modules
src
assets
components
pages
CustomerOrder.vue
Login.vue
Products.vue
AlertMessage.vue
Dashboard.vue
Index.vue
Navbar.vue M
Sidebar.vue M
filters
currency.js
App.vue
bus.js
main.js
router.js
static
.babelrc
.editorconfig
.gitignore
.postcssrc.js

product > src > components > V Navbar.vue > {} "Navbar.vue"
14   <span class="navbar-toggler-icon"></span>
15   </button>
16   <input
17     class="form-control form-control-dark w-100"
18     type="text"
19     placeholder="Search"
20     aria-label="Search"
21   />
22   <ul class="navbar-nav px-3">
23     <li class="nav-item textnowrap">
24       <a
25         class="nav-link"
26         href="#"
27         @click.prevent="signout">Sign out</a>
28     </li>
29   </ul>
30   </nav>
31 </div>
32 </template>
33
34 <script>
35 export default {
36   name: 'Navbar',
37   methods: {
38     signout() {
39       // 依據 API 文件指出 此api 址需傳入 api 不須傳入其他資料 即可登出
40       const api = `${process.env.APIPATH}/logout`;
41       const vm = this;
42       this.$http.post(api).then(response => { // 然後 使用 post 傳入用戶資料 vm.user
43         console.log(response.data);
44         if (response.data.success){ // 這裡寫一個判別式：如果登出成功 就將路徑 轉到我們登入頁面
45           vm.$router.push('/login') // 確定登出就將頁面 導回 signin
46         }
47       });
48     }
49   }
50 }
51 </script>

```

此時這裡的 登出功能就會在 按下後登出 並取 導到 登入畫面

Sign out

建立新產品

售價

狀態

編輯

刪除

首先建立一個新的元件頁面 CustomerOrder.vue (先使用六角提供的 BS 產品圖卡模板)

然後完成與 vue methods 的資料綁定

The screenshot shows a code editor with two tabs: 'CustomerOrder.vue' and 'router.js'. The 'CustomerOrder.vue' tab displays the following template code:

```
<template>
  <div>
    <loading :active.sync="isLoading"></loading>
    <div>
      <div class="row mt-4">
        <div v-for="item in products" :key="item.id">
          <div class="card border-0 shadow-sm">
            <div style="height: 500px; background-size: cover; background-position: center" :style="{backgroundImage: `url(${item.imageUrl})`}"></div>
            <div class="card-body">
              <span class="badge badge-secondary float-right ml-2">{{item.category}}</span>
              <h5 class="card-title">
                <a href="#" class="text-dark">{{item.title}}</a>
              </h5>
              <p class="card-text">{{item.content}}</p>
              <div class="d-flex justify-content-between align-items-baseline">
                <div class="h5 v-if="!item.price">{{item.origin_price}}</div>
                <del class="h6 v-if="item.price">{{item.origin_price}}</del>
                <div class="h5 v-if="item.price">{{item.price}}</div>
              </div>
            </div>
            <div class="card-footer d-flex">
              <button type="button" class="btn btn-outline-secondary btn-sm">
                <i class="fas fa-spinner fa-spin"></i>
                查看更多
              </button>
              <button type="button" class="btn btn-outline-danger btn-sm ml-auto">
                <i class="fas fa-spinner fa-spin"></i>
                加到購物車
              </button>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
```

The 'router.js' tab contains the following script code:

```
import $ from 'jquery';

export default {
  data() {
    return {
      products: [],
      isLoading: false,
    };
  },
  methods: {
    getProducts() {
      const vm = this;
      const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/products`;
      vm.isLoading = true;
      this.$http.get(url).then((response) => {
        vm.products = response.data.products;
        console.log(response);
        vm.isLoading = false;
      });
    },
    created() {
      this.getProducts();
    },
  },
}</script>
```

The screenshot shows a code editor with three tabs: 'Sidebar.vue', 'CustomerOrder.vue', and 'router.js'. The 'CustomerOrder.vue' tab displays the following template code:

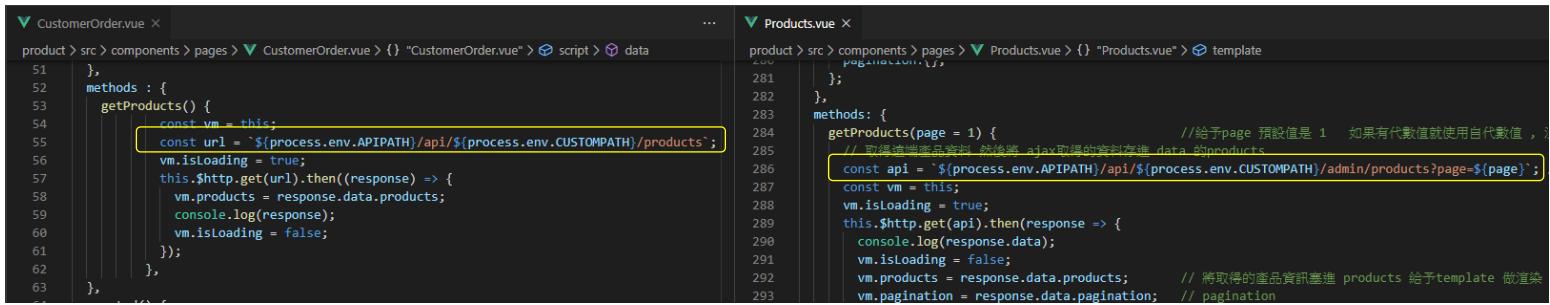
```
<template>
  <div>
    <loading :active.sync="isLoading"></loading>
    <div>
      <div class="row mt-4">
        <div v-for="item in products" :key="item.id">
          <div class="card border-0 shadow-sm">
            <div style="height: 500px; background-size: cover; background-position: center" :style="{backgroundImage: `url(${item.imageUrl})`}"></div>
            <div class="card-body">
              <span class="badge badge-secondary float-right ml-2">{{item.category}}</span>
              <h5 class="card-title">
                <a href="#" class="text-dark">{{item.title}}</a>
              </h5>
              <p class="card-text">{{item.content}}</p>
              <div class="d-flex justify-content-between align-items-baseline">
                <div class="h5 v-if="!item.price">{{item.origin_price}}</div>
                <del class="h6 v-if="item.price">{{item.origin_price}}</del>
                <div class="h5 v-if="item.price">{{item.price}}</div>
              </div>
            </div>
            <div class="card-footer d-flex">
              <button type="button" class="btn btn-outline-secondary btn-sm">
                <i class="fas fa-spinner fa-spin"></i>
                查看更多
              </button>
              <button type="button" class="btn btn-outline-danger btn-sm ml-auto">
                <i class="fas fa-spinner fa-spin"></i>
                加到購物車
              </button>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>
```

The 'router.js' tab contains the following script code:

```
import $ from 'jquery';

export default {
  data() {
    return {
      products: [],
      isLoading: false,
    };
  },
  methods: {
    getProducts() {
      const vm = this;
      const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/products`;
      vm.isLoading = true;
      this.$http.get(url).then((response) => {
        vm.products = response.data.products;
        console.log(response);
        vm.isLoading = false;
      });
    },
    created() {
      this.getProducts();
    },
  },
}</script>
```

以上要注意 getProduct 資料的 api 路徑 是不一樣的 一個是用戶 一個是管理員

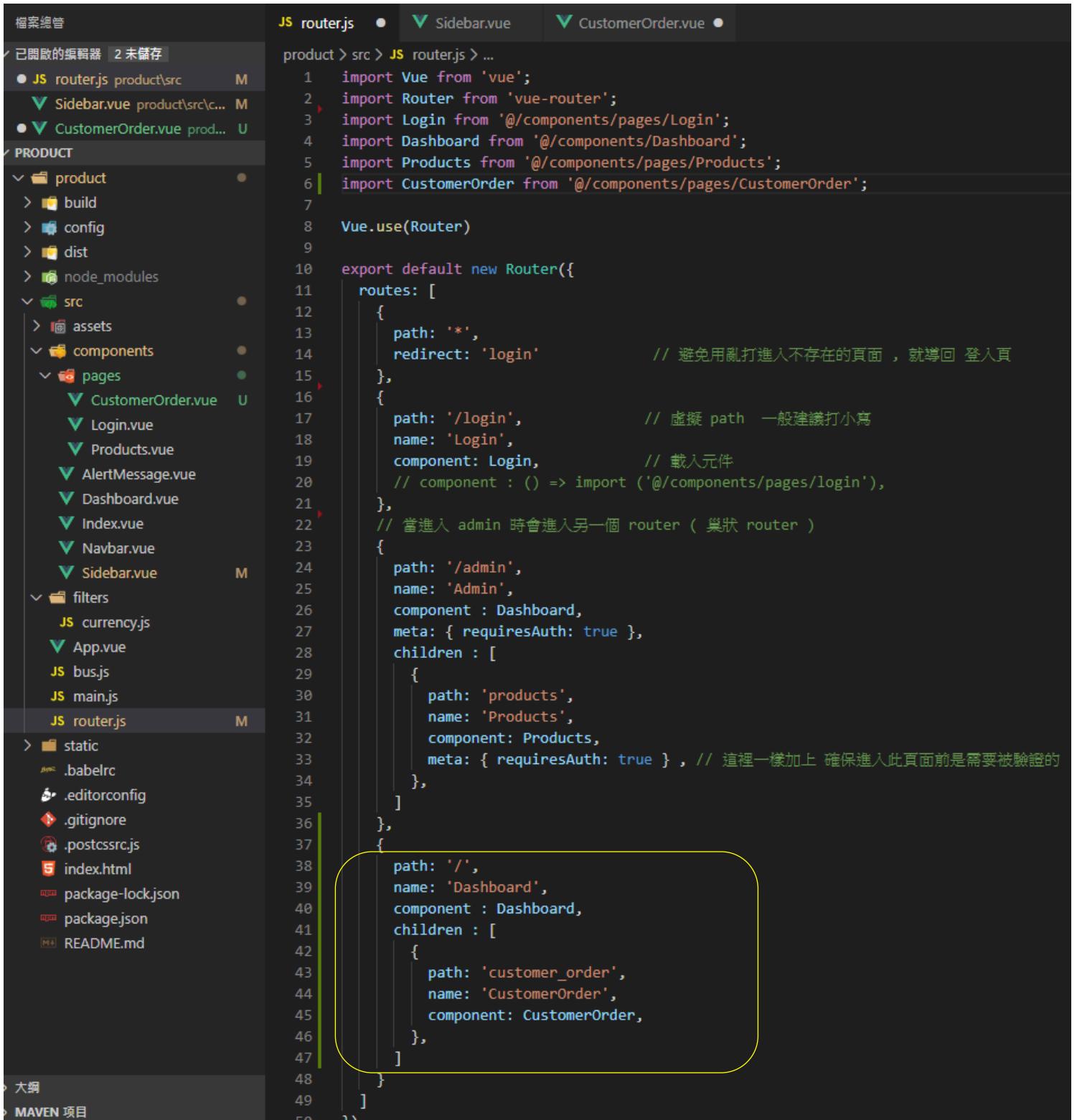


```
CustomerOrder.vue
...
product > src > components > pages > CustomerOrder.vue > () "CustomerOrder.vue" > script > data
51 },
52 methods: {
53   getProducts() {
54     const vm = this;
55     const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/products`;
56     vm.isLoading = true;
57     this.$http.get(url).then((response) => {
58       vm.products = response.data.products;
59       console.log(response);
60       vm.isLoading = false;
61     });
62   },
63 },
```

```
Products.vue
...
product > src > components > pages > Products.vue > () "Products.vue" > template
280   pagination,
281   ],
282 },
283 },
284 methods: {
285   getProducts(page = 1) { // 給予page 預設值是 1 如果有代數值就使用自代數值
286     // 取得適當產品資料 然後將 ajax取得的資料塞進 data 的products
287     const api = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/products?page=${page}`;
288     const vm = this;
289     vm.isLoading = true;
290     this.$http.get(api).then((response => {
291       console.log(response.data);
292       vm.isLoading = false;
293       vm.products = response.data.products; // 將取得的產品資訊塞進 products 紙於template 做渲染
294       vm.pagination = response.data.pagination; // pagination
295     })
296   }
297 }
```

最後將新增的頁面加入到 router 裡面

Import 後在下方新增相對應路徑 類似於 admin 路由，但 `meta: { requiresAuth: true }` 可以刪除
因為他不需要經過驗證就可以進入



檔案總管

- 已開啟的編輯器 2 未儲存
- JS router.js product\src
- Sidebar.vue product\src\...
- CustomerOrder.vue product\src\...

PRODUCT

- product
 - build
 - config
 - dist
 - node_modules
- src
 - assets
 - components
 - pages
 - CustomerOrder.vue
 - Login.vue
 - Products.vue
 - AlertMessage.vue
 - Dashboard.vue
 - Index.vue
 - Navbar.vue
 - Sidebar.vue
 - filters
 - currency.js
 - App.vue
 - bus.js
 - main.js
 - router.js
- static
 - .babelrc
 - .editorconfig
 - .gitignore
 - .postcssrc.js
 - index.html
 - package-lock.json
 - package.json
 - README.md

JS router.js ●

```
product > src > JS router.js > ...
1 import Vue from 'vue';
2 import Router from 'vue-router';
3 import Login from '@/components/pages/Login';
4 import Dashboard from '@/components/Dashboard';
5 import Products from '@/components/pages/Products';
6 import CustomerOrder from '@/components/pages/CustomerOrder';
7
8 Vue.use(Router)
9
10 export default new Router({
11   routes: [
12     {
13       path: '*',
14       redirect: 'login' // 避免用亂打進入不存在的頁面，就導回 登入頁
15     },
16     {
17       path: '/login', // 虛擬 path 一般建議打小寫
18       name: 'Login',
19       component: Login, // 載入元件
20       // component : () => import('@/components/pages/login'),
21     },
22     // 當進入 admin 時會進入另一個 router ( 套狀 router )
23     {
24       path: '/admin',
25       name: 'Admin',
26       component: Dashboard,
27       meta: { requiresAuth: true },
28       children : [
29         {
30           path: 'products',
31           name: 'Products',
32           component: Products,
33           meta: { requiresAuth: true } , // 這裡一樣加上 確保進入此頁面前是需要被驗證的
34         },
35       ]
36     },
37     {
38       path: '/',
39       name: 'Dashboard',
40       component: Dashboard,
41       children : [
42         {
43           path: 'customer_order',
44           name: 'CustomerOrder',
45           component: CustomerOrder,
46         },
47       ]
48     }
49   ]
50 })
```

然後將以上路徑 放入 sidebar.vue 裏頭

The screenshot shows a code editor interface with several tabs at the top: 'Sidebar.vue' (active), 'CustomerOrder.vue', 'JS router.js (Working Tree)', and 'JS router.js'. The left sidebar displays the project structure:

- 已開啟的編輯器:
 - Sidebar.vue
 - CustomerOrder.vue
 - JS router.js (Working Tree)
 - JS router.js
- PRODUCT:
 - product
 - build
 - config
 - dist
 - node_modules
 - src
 - assets
 - components
 - CustomerOrder.vue
 - Login.vue
 - Products.vue
 - AlertMessage.vue
 - Dashboard.vue
 - Index.vue
 - Navbar.vue
 - Sidebar.vue
 - filters
 - currency.js
 - App.vue
 - bus.js
 - main.js
 - router.js
 - static
 - .babelrc
 - .editorconfig
 - .gitignore
 - .postcssrc.js
 - index.html
 - package-lock.json
 - package.json
 - README.md
- 大綱
- MAVEN 项目

The main content area shows the template for 'Sidebar.vue':

```
<template>
  <div>
    <nav id="sidebarMenu" class="col-md-3 col-lg-2 d-md-block bg-light sidebar collapse" aria-expanded="false" aria-label="Toggle sidebar">
      <div class="sidebar-sticky pt-3">
        <h6>
          <span>管理員</span>
          <a class="d-flex align-items-center text-muted" href="#" aria-label="Add a new product">
            <span data-feather="plus-circle"></span>
          </a>
        </h6>
        <ul class="nav flex-column mb-2">
          <li class="nav-item">
            <!-- <a class="nav-link" href="#">
            <span data-feather="file-text"></span>
            產品列表
            </a> -->
            <router-link
              to="/admin/products"
              class="nav-link">
              產品列表
            </router-link>
          </li>
        </ul>
        <h6>
          <span>模擬功能</span>
          <a class="d-flex align-items-center text-muted" href="#" aria-label="Add a new customer order">
            <span data-feather="plus-circle"></span>
          </a>
        </h6>
        <ul class="nav flex-column mb-2">
          <li class="nav-item">
            <!-- <a class="nav-link" href="#">
            <span data-feather="file-text"></span>
            模擬訂單
            </a> -->
            <router-link
              to="/customer_order"
              class="nav-link">
              模擬訂單
            </router-link>
          </li>
        </ul>
      </div>
    </nav>
  </div>
</template>
```

A yellow rounded rectangle highlights the `<router-link to="/customer_order" class="nav-link">` line at line 44.

<https://github.com/Wcc723/vue-course-record/commit/742ee45730ad518fd6a0c89d418f930fae1d1d28>

- * 新增 訂單列表、優惠券頁面
- * 新增 date filter 紿予 上述兩個頁面使用
- * 新增以上例各 頁面至 Vue Router
- * 提供完整 將 Pagination 給元件化

首先 針對之前的 分頁 Pagination 給元件化：

新增 Pagination.vue

The screenshot shows a code editor with a sidebar containing a project tree. The tree includes a 'Pagination.vue' file under 'src/components'. The main pane displays the code for 'Pagination.vue'.

```

<template>
  <nav aria-label="Page navigation example">
    <ul class="pagination justify-content-center">
      <li
        class="page-item"
        :class="{'disabled': !pages.has_pre}"
        <a
          class="page-link"
          href="#"
          aria-label="Previous"
          @click.prevent="updatePage(pages.current_page - 1)">
            <span aria-hidden="true"></span>
            <span class="sr-only">Previous</span>
          </a>
        </li>
      <li
        class="page-item"
        v-for="page in pages.total_pages"
        :key="page"
        :class="{'active': pages.current_page == page}">
        <a
          class="page-link"
          href="#"
          @click.prevent="updatePage(page)">
            {{ page }}</a>
        </li>
      <li
        class="page-item"
        :class="{'disabled': !pages.has_next}"
        <a
          class="page-link"
          href="#"
          aria-label="Next"
          @click.prevent="updatePage(pages.current_page + 1)">
            <span aria-hidden="true">&raquo;</span>
            <span class="sr-only">Next</span>
          </a>
        </li>
      </ul>
    </nav>
</template>

<script>
// :pages="{ 頁碼資訊 }" // @emitPages="更新頁面事件"
export default {
  name: 'Layout',
  props: [ 'pages' ],
  data() {
    return {
      msg: 'Welcome to Your Vue.js App',
      uid: '',
    };
  },
  methods: {
    updatePage(page) {
      this.$emit('emitPages', page);
    },
  },
};
</script>

```

接著 修改原本 Products.vue 的 分頁部分 (將其改成用 元建置入)

檔案總管

```

product > src > components > pages > Products.vue > {} "Products.vue" > template
200   <script>
201     import $ from 'jquery';
202     import Pagination from '../Pagination';
203
204     export default {
205       data() {
206         return {
207           products: [], // 所有新增的產品資料會放進這裡
208           tempProduct:{}, // for modal 這裡會存放要送到資料庫 產品資料(透過 post 方法)
209           isNew : false,
210           isLoading : false, // loading 畫面 啟用與否
211           status:{},
212             fileUploading : false,
213           },
214           pagination:{},
215         },
216         components: {
217           Pagination,
218         },
219         methods: {
220           getProducts(page = 1) { //給予page 預設值是 1 如果沒有
221             // 取得遠端產品資料 然後將 ajax取得的資料存進 data 的products
222             const api = `http://127.0.0.1:8000/api/v1/processes/only_CUSTOMDATA/admin/proc

```

檔案總管

```

product > src > components > pages > Products.vue > {} "Products.vue" > template
51   <!-- <nav aria-label="Page navigation example">
52     <ul class="pagination">
53       <li
54         class="page-item"
55         :class="{'disabled':pagination.has_pre === false}"
56       <a
57         class="page-link"
58         href="#"
59         aria-label="Previous"
60         @click.prevent="getProducts(pagination.current_page - 1)"
61           <span aria-hidden="true"></span>
62           <span class="sr-only">Previous</span>
63         </a>
64       </li>
65       <li
66         class="page-item"
67         v-for="page in pagination.total_pages"
68         :key="page"
69         :class="{'active': pagination.current_page === page}" >
70         <a
71           class="page-link"
72           href="#"
73           @click.prevent="getProducts(page)">{{page}}</a>
74       </li>
75       <li
76         class="page-item"
77         :class="{'disabled':pagination.has_next === false}"
78       <a
79         class="page-link"
80         href="#"
81         aria-label="Next"
82         @click.prevent="getProducts(pagination.current_page + 1)"
83           <span aria-hidden="true">&raquo;</span>
84           <span class="sr-only">Next</span>
85         </a>
86       </li>
87     </ul>
88   </nav> -->
89   <Pagination :pages="pagination" @emitPages="getProducts"></Pagination>

```

ProductList.vue

```

components > admin > ProductList.vue > {} "ProductList.vue" > template > div > div#product
32   </tbody>
33 </table>
34
35   <Pagination :pages="pagination" @emitPages="getProducts"/>
36
37   <!-- Modal -->
38   <div class="modal fade" id="productModal" tabindex="-1" role="dialog">
39     <div class="modal-dialog modal-lg" role="document">
40       <div class="modal-content border-0">
41         <div class="modal-header bg-dark text-white">
42           <h5 class="modal-title" id="exampleModalLabel">
43             新增產品
44           </h5>
45           <button type="button" class="close" data-dismiss="modal"
46             aria-hidden="true">&times;</span>
47           </button>
48         </div>
49         <div class="modal-body">

```

Pagination.vue

```

src > components > component > Pagination.vue > {} "Pagination.vue"
40   </template>
41   <script>
42   // :pages="{ 資訊 }" // @emitPages="更新頁面事件"
43   export default {
44     name: 'Layout',
45     props: ['pages'],
46     data() {
47       return {
48         msg: 'Welcome to Your Vue.js App',
49         uid: '',
50       };
51     },
52     methods: {
53       updatePage(page) {
54         this.$emit('emitPages', page);
55       },
56     },
57   };
58   </script>

```

再來 開始新增 新頁面

首先 先把 sidebar 加入 要新增頁面的新連結

檔案總管 Sidebar.vue

```
product > src > components > Sidebar.vue > ...
10   <a class="d-flex align-items-center text-muted" href="#" aria-label="Add a new report">
11     <span data-feather="plus-circle"></span>
12   </a>
13   </h6>
14   <ul class="nav flex-column mb-2">
15     <li class="nav-item">
16       <!-- <a class="nav-link" href="#"><span data-feather="file-text"></span>產品列表</a> -->
17       <router-link
18         to="/admin/products"
19         class="nav-link">
20         產品列表
21       </router-link>
22     </li>
23     <li class="nav-item">
24       <router-link
25         to="/admin/orders"
26         class="nav-link">
27         訂單列表
28       </router-link>
29     </li>
30     <li class="nav-item">
31       <router-link
32         to="/admin/coupons"
33         class="nav-link">
34         優惠券
35       </router-link>
36     </li>
37   </ul>
38   <h6 class="sidebar-heading d-flex justify-content-between align-items-center px-3 mt-4 mb-1 text-muted">
39     <span>模擬功能</span>
40     <a class="d-flex align-items-center text-muted" href="#" aria-label="Add a new report">
41       <span data-feather="plus-circle"></span>
42     </a>
43   </h6>
44   <ul class="nav flex-column mb-2">
45     <li class="nav-item">
46       <!-- <a class="nav-link" href="#"><span data-feather="file-text"></span>模擬訂單</a> -->
47       <router-link
48         to="/customer_order"
49         class="nav-link">
50         模擬訂單
51       </router-link>
52     </li>
53   </ul>
54   </div>
55 </nav>
56 </div>
57 </template>
```

開始新增頁面：

優惠券 頁面 - Coupons.vue

Coupons.vue ×

```
product > src > components > pages > Coupons.vue > {} "Coupons.vue" > template > div > table.table.mt-4 > tbody
1  <template>
2    <div>
3      <div class="text-right">
4        <button class="btn btn-primary" @click="openCouponModal(true)">
5          建立新的優惠券
6        </button>
7      </div>
8      <table class="table mt-4">
9        <thead>
10       <tr>
11         <th>名稱</th>
12         <th>折扣百分比</th>
13         <th>到期日</th>
14         <th>是否啟用</th>
15         <th>編輯</th>
16       </tr>
17     </thead>
18     <tbody>
19       <tr v-for="(item, key) in coupons" :key="key">
20         <td>{{ item.title }}</td>
21         <td>{{ item.percent }}%</td>
22         <td>{{ item.due_date | date }}</td>
23         <td>
24           <span v-if="item.is_enabled === 1" class="text-success">啟用</span>
25           <span v-else class="text-muted">未起用</span>
26         </td>
27         <td>
28           <button class="btn btn-outline-primary btn-sm"
29             @click="openCouponModal(false, item)">
30             編輯</button>
31         </td>
32       </tr>
33     </tbody>
34   </table>
35   <div class="modal fade" id="couponModal" tabindex="-1" role="dialog"
36     aria-labelledby="exampleModalLabel" aria-hidden="true">
37     <div class="modal-dialog" role="document">
38       <div class="modal-content">
39         <div class="modal-header">
40           <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>
41           <button type="button" class="close" data-dismiss="modal" aria-label="Close">
42             <span aria-hidden="true">&times;</span>
43           </button>
44         </div>
45         <div class="modal-body">
46           <div class="form-group">
47             <label for="title">標題</label>
48             <input type="text" class="form-control" id="title" v-model="tempCoupon.title"
49               placeholder="請輸入標題">
50           </div>
51           <div class="form-group">
52             <label for="coupon_code">優惠碼</label>
53             <input type="text" class="form-control" id="coupon_code" v-model="tempCoupon.code"
54               placeholder="請輸入優惠碼">
55           </div>
56           <div class="form-group">
57             <label for="due_date">到期日</label>
58             <input type="date" class="form-control" id="due_date"
59               v-model="due_date">
60           </div>
```

Coupons.vue ×

```
product > src > components > pages > Coupons.vue > {"Coupons.vue"} > script > watch > due_date
 61      <div class="form-group">
 62          <label for="price">折扣百分比</label>
 63          <input type="number" class="form-control" id="price"
 64              v-model="tempCoupon.percent" placeholder="請輸入折扣百分比">
 65      </div>
 66      <div class="form-group">
 67          <div class="form-check">
 68              <input class="form-check-input" type="checkbox"
 69                  :true-value="1"
 70                  :false-value="0"
 71                  v-model="tempCoupon.is_enabled" id="is_enabled">
 72              <label class="form-check-label" for="is_enabled">
 73                  是否啟用
 74              </label>
 75          </div>
 76      </div>
 77      </div>
 78      <div class="modal-footer">
 79          <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
 80          <button type="button" class="btn btn-primary"
 81              @click="updateCoupon">更新優惠券</button>
 82      </div>
 83  </div>
 84  </div>
 85  </div>
 86  </div>
 87 </template>
 88
 89 <script>
 90 import $ from 'jquery';
 91 export default {
 92     props: {
 93         config: Object,
 94     },
 95     data() {
 96         return {
 97             coupons: {},
 98             tempCoupon: {
 99                 title: '',
100                 is_enabled: 0,
101                 percent: 100,
102                 due_date: 0,
103                 code: '',
104             },
105             due_date: new Date(),
106             isNew: false,
107         };
108     },
109     watch: {
110         due_date() {
111             const vm = this;
112             const timestamp = Math.floor(new Date(vm.due_date) / 1000);
113             vm.tempCoupon.due_date = timestamp;
114         },
115     },
116 };
```

Coupons.vue ×

product > src > components > pages > Coupons.vue > {} "Coupons.vue" > script

```
116     methods: {
117       openCouponModal(isNew, item) {
118         const vm = this;
119         $('#couponModal').modal('show');
120         vm.isNew = isNew;
121         if (vm.isNew) {
122           vm.tempCoupon = {};
123         } else {
124           vm.tempCoupon = Object.assign({}, item);
125           const dateAndTime = new Date(vm.tempCoupon.due_date * 1000).toISOString().split('T');
126           vm.due_date = dateAndTime[0];
127         }
128       },
129       getCoupons() {
130         const vm = this;
131         const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/coupons`;
132         this.$http.get(url, vm.tempProduct).then((response) => {
133           vm.coupons = response.data.coupons;
134           console.log(response);
135         });
136       },
137       updateCoupon() {
138         const vm = this;
139         if (vm.isNew) {
140           const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/coupon`;
141           this.$http.post(url, { data: vm.tempCoupon }).then((response) => {
142             console.log(response, vm.tempCoupon);
143             $('#couponModal').modal('hide');
144             this.getCoupons();
145           });
146         } else {
147           const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/coupon/${vm.tempCoupon.id}`;
148           vm.due_date = new Date(vm.tempCoupon.due_date * 1000);
149           this.$http.put(url, { data: vm.tempCoupon }).then((response) => {
150             console.log(response);
151             $('#couponModal').modal('hide');
152             this.getCoupons();
153           });
154         }
155       },
156     },
157     created() {
158       this.getCoupons();
159     },
160   };
161 </script>
```

訂單 頁面 - Orders.vue

Orders.vue ×

```
product > src > components > pages > Orders.vue > {} "Orders.vue" > template
1  <template>
2    <div>
3      <Loading :active.sync="isLoading"></Loading>
4      <table class="table mt-4">
5        <thead>
6          <tr>
7            <th>購買時間</th>
8            <th>Email</th>
9            <th>購買款項</th>
10           <th>應付金額</th>
11           <th>是否付款</th>
12         </tr>
13       </thead>
14       <tbody>
15         <tr v-for="(item, key) in sortOrder" :key="key"
16           v-if="orders.length"
17           :class="{'text-secondary': !item.is_paid}">
18           <td>{{ item.create_at | date }}</td>
19           <td><span v-text="item.user.email" v-if="item.user"></span></td>
20           <td>
21             <ul class="list-unstyled">
22               <li v-for="(product, i) in item.products" :key="i">
23                 {{ product.product.title }} 數量：{{ product.qty }}
24                 {{ product.product.unit }}
25               </li>
26             </ul>
27           </td>
28           <td class="text-right">{{ item.total | currency }}</td>
29           <td>
30             <strong v-if="item.is_paid" class="text-success">已付款</strong>
31             <span v-else class="text-muted">尚未起用</span>
32           </td>
33         </tr>
34       </tbody>
35     </table>
36
37     <Pagination :pages="pagination" @emitPages="getOrders"></Pagination>
38   </div>
39 </template>
40
41 <script>
42 import Pagination from '../Pagination';
43 export default {
44   data() {
45     return {
46       orders: {},
47       isNew: false,
48       pagination: {},
49       isLoading: false,
50     };
51   },
52   components: {
53     Pagination,
54   },
55 };
```

product > src > components > pages > Orders.vue > "Orders.vue" > template

```
55  methods: {
56    getOrders(currentPage = 1) {
57      const vm = this;
58      const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/admin/orders?page=${currentPage}`;
59      vm.isLoading = true;
60      this.$http.get(url, vm.tempProduct).then((response) => {
61        vm.orders = response.data.orders;
62        vm.pagination = response.data.pagination;
63        vm.isLoading = false;
64        console.log(response);
65      });
66    },
67  },
68  computed: {
69    sortOrder() {
70      const vm = this;
71      let newOrder = [];
72      if (vm.orders.length) {
73        newOrder = vm.orders.sort((a, b) => {
74          const aIsPaid = a.is_paid ? 1 : 0;
75          const bIsPaid = b.is_paid ? 1 : 0;
76          return bIsPaid - aIsPaid;
77        });
78      }
79      return newOrder;
80    },
81  },
82  created() {
83    this.getOrders();
84    console.log(process.env.APIPATH);
85  },
86};
87</script>
```

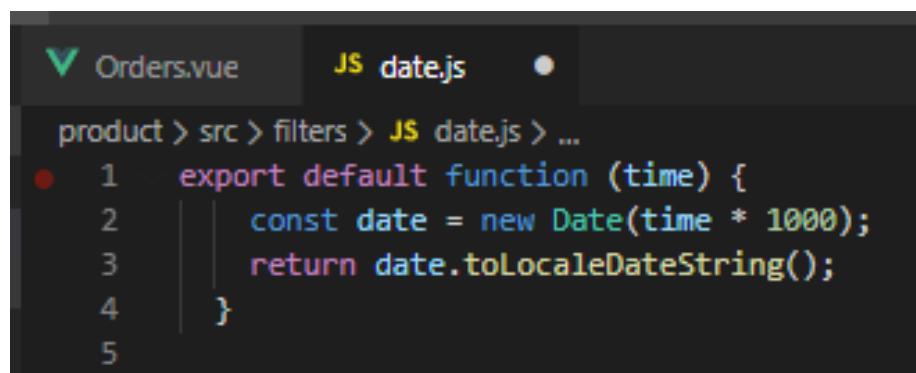
完成上述 兩個頁面 記得要在 router 中嵌入

The screenshot shows a code editor with two tabs: 'Orders.vue' and 'router.js'. The 'Orders.vue' tab is currently active, displaying a component definition. The 'router.js' tab is visible in the background.

```
Orders.vue
product > src > JS router.js ●
1 import Vue from 'vue';
2 import Router from 'vue-router';
3
4 // import Index from '@/components/Index';
5 import Login from '@/components/pages/Login';
6 import Dashboard from '@/components/Dashboard';
7 import Products from '@/components/pages/Products';
8 import Coupons from '@/components/pages/Coupons';
9 import Orders from '@/components/pages/Orders';
10 import CustomerOrder from '@/components/pages/CustomerOrder';
11
12
13 Vue.use(Router)
14
15 export default new Router({
16   linkActiveClass: 'active',
17   routes: [
18     {
19       path: '*',
20       redirect: 'login'          // 避免用亂打進入不存在的頁面，就導回 登入頁
21     },
22     // ...
23   ],
24   // ...
25 }
26
27 // 當進入 admin 時會進入另一個 router ( 嵌狀 router )
28 {
29   path: '/admin',
30   name: 'Admin',
31   component : Dashboard,
32   meta: { requiresAuth: true },
33   children : [
34     {
35       path: 'products',
36       name: 'Products',
37       component: Products,
38       meta: { requiresAuth: true } , // 這裡一樣加上 確保
39     },
40     {
41       path: 'coupons',
42       name: 'Coupons',
43       component: Coupons,
44       meta: { requiresAuth: true },
45     },
46     {
47       path: 'orders',
48       name: 'Orders',
49       component: Orders,
50       meta: { requiresAuth: true },
51     },
52   ],
53 },
54 ],
55
56
57
58
59   ],
60 ]
61 },
```

接著加入 date filter 紿予 上述兩個頁面使用

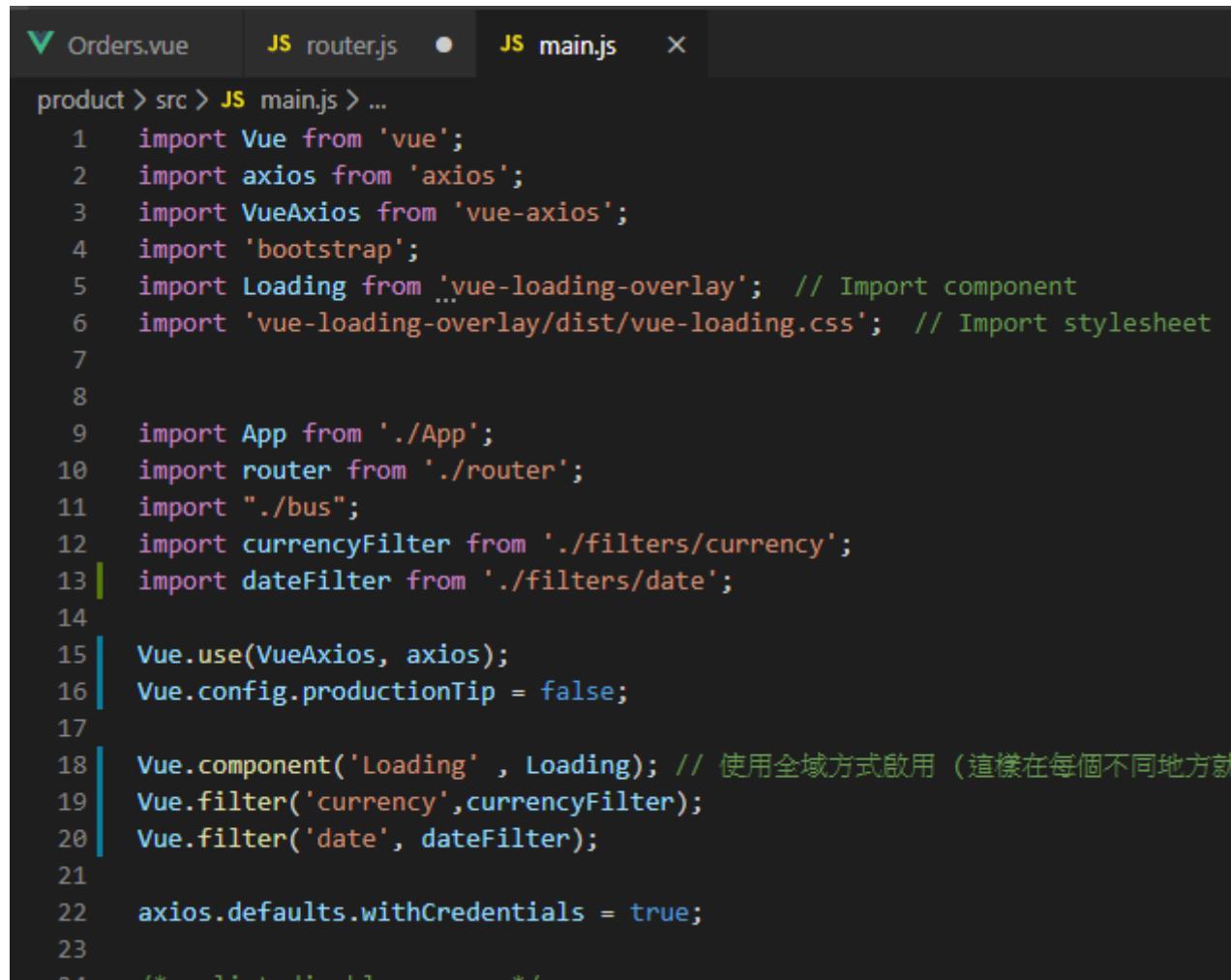
新增 date.js



product > src > filters > JS date.js > ...

```
1  export default function (time) {
2    const date = new Date(time * 1000);
3    return date.toLocaleDateString();
4  }
5
```

在 main.js 中引入



product > src > JS main.js > ...

```
1  import Vue from 'vue';
2  import axios from 'axios';
3  import VueAxios from 'vue-axios';
4  import 'bootstrap';
5  import Loading from 'vue-loading-overlay'; // Import component
6  import 'vue-loading-overlay/dist/vue-loading.css'; // Import stylesheets
7
8
9  import App from './App';
10 import router from './router';
11 import './bus';
12 import currencyFilter from './filters/currency';
13 import dateFilter from './filters/date';
14
15 Vue.use(VueAxios, axios);
16 Vue.config.productionTip = false;
17
18 Vue.component('Loading', Loading); // 使用全域方式啟用 (這樣在每個不同地方就
19 Vue.filter('currency', currencyFilter);
20 Vue.filter('date', dateFilter);
21
22 axios.defaults.withCredentials = true;
23
24 /* eslint-disable no-console */
```

11- 103. 取得單一產品

目前 按下 查看更多會 透過 彈出 modal (在那之前會先取得 那筆單筆資料 資料) 來顯示我們的產品

```
[API]: /api/:api_path/product/:id
[說明]: @api_path = 'thisismycourse2'
          @id = '-L9tH8jxVb2Ka_DYPwng'

[方法]: get
[成功回應]:
{
  "success": true,
  "product": {
    "category": "衣服3",
    "content": "這是內容",
    "description": "Sit down please 名設計師設計",
    "id": "-L9tH8jxVb2Ka_DYPwng",
    "imageUrl": "test.testtest",
    "is_enabled": 1,
    "num": 1,
    "origin_price": 100,
    "price": 600,
    "title": "[賣]動物園造型衣服3",
    "unit": "個"
  }
}
[失敗回應]: 產品未啟用
{
  "success": false,
  "message": "產品未啟用"
}
[失敗回應]: 找不到產品
{
  "success": false,
  "message": "找不到產品"
}
```

點擊後 才是將完整的資料取出

另定義一個 product 來存放 modal 要開啟的那筆 單一筆 資料

另定義一個 status 解決原本 loading icon 會在畫面每個地方都有 的問題

(loadingItem >> 存放 點擊的產品 ID , 其 主要是用來判斷目前畫面上次哪一個元素正在讀取中)

Sidebar.vue CustomerOrder.vue Products.vue router.js

```
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue" > template > div
```

```
1  <template>
2    <div>
3      <loading :active.sync="isLoading"></loading>
4    <div
5      class="row mt-4"
6    <div
7      class="col-md-4 mb-4"
8      v-for="item in products"
9      :key="item.id"
10     <div class="card border-0 shadow-sm">
11       <div
12         style="height: 500px; background-size: cover; background-position: center"
13         :style="{backgroundImage: `url(${item.imageUrl})`}"></div>
14       <div class="card-body">
15         <span class="badge badge-secondary float-right ml-2">{{item.category}}</span>
16         <h5 class="card-title">
17           <a href="#" class="text-dark">{{item.title}}</a>
18         </h5>
19         <p class="card-text">{{item.content}}</p>
20         <div class="d-flex justify-content-between align-items-baseline">
21           <div class="h5" v-if="!item.price">{{item.origin_price}}</div>
22           <del class="h6" v-if="item.price">{{item.origin_price}}</del>
23           <div class="h5" v-if="item.price">{{item.price}}</div>
24         </div>
25       </div>
26       <div class="card-footer d-flex">
27         <button
28           type="button"
29           class="btn btn-outline-secondary btn-sm"
30           @click="getProduct(item.id)">
31           <!-- 如果 目前的 loading Id 和 item id 相符 才顯示此讀取效果 ( 解決原本全頁面都有 loading icon 的問題 ) -->
32           <i
33             class="fas fa-spinner fa-spin"
34             v-if="status.loadingItem === item.id"></i>
35           查看更多
36         </button>
37         <button type="button" class="btn btn-outline-danger btn-sm ml-auto">
38           <!-- 如果 目前的 loading Id 和 item id 相符 才顯示此讀取效果 ( 解決原本全頁面都有 loading icon 的問題 ) -->
39           <i
40             class="fas fa-spinner fa-spin"
41             v-if="status.loadingItem === item.id"></i>
42           加到購物車
43         </button>
44       </div>
45     </div>
46   </div>
47 </div>
```

Sidebar.vue CustomerOrder.vue Products.vue router.js

```
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue" > template > div > div.row.mt-4 > div.col-md-4.mb-4
```

```
48   <div class="modal fade" id="productModal" tabindex="-1" role="dialog"
49     aria-labelledby="exampleModalLabel" aria-hidden="true">
50     <div class="modal-dialog" role="document">
51       <div class="modal-content">
52         <div class="modal-header">
53           <h5 class="modal-title" id="exampleModalLabel">{{ product.title }}</h5>
54           <button type="button" class="close" data-dismiss="modal" aria-label="Close">
55             <span aria-hidden="true">&times;</span>
56           </button>
57         </div>
58         <div class="modal-body">
59           
60           <blockquote class="blockquote mt-3">
61             <p class="mb-0">{{ product.content }}</p>
62             <footer class="blockquote-footer text-right">{{ product.description }}</footer>
63           </blockquote>
64           <div class="d-flex justify-content-between align-items-baseline">
65             <div class="h4" v-if="!product.price">{{ product.origin_price }} 元</div>
66             <del class="h6" v-if="product.price">原價 {{ product.origin_price }} 元</del>
67             <div class="h4" v-if="product.price">現在只要 {{ product.price }} 元</div>
68           </div>
69           <select name="" class="form-control mt-3" v-model="product.num">
70             <option :value="num" v-for="num in 10" :key="num">
71               選購 {{num}} {{product.unit}}
72             </option>
73           </select>
74         </div>
75         <div class="modal-footer">
76           <div class="text-muted text nowrap mr-3">
77             小計 <strong>{{ product.num * product.price }}</strong> 元
78           </div>
79           <button type="button" class="btn btn-primary">
80             加到購物車
81           </button>
82         </div>
83       </div>
84     </div>
85   </div>
86 </div>
```

Sidebar.vue CustomerOrder.vue ● Products.vue JS router.js

```

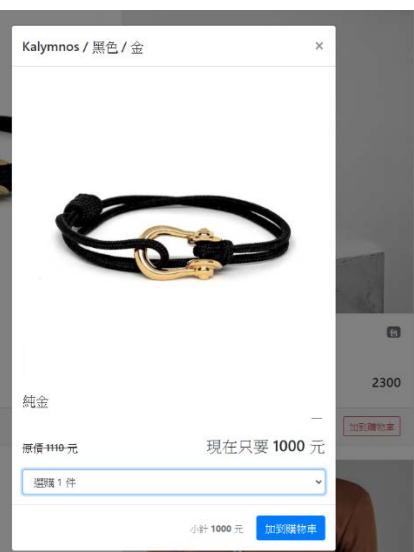
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue"
88
89  <script>
90  import $ from 'jquery';
91
92  export default {
93    data() {
94      return {
95        products: [],
96        product: {}, // 這裡是用來存 modal 的資料
97        status: { // 新增一狀態 判斷目前畫面是哪個 查看更多按鈕 正在讀取中 ( 解決原本全頁面都有 loading icon 的問題 )
98          loadingItem: '' // 將會存放 點擊的產品 ID
99        },
100        isLoading: false,
101      }
102    },
103    methods: {
104      getProducts() {
105        const vm = this;
106        const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/products`;
107        vm.isLoading = true;
108        this.$http.get(url).then((response) => {
109          vm.products = response.data.products;
110
111          console.log(response);
112          vm.isLoading = false;
113        });
114      },
115      getProduct(id) { // 取得單一筆 資料，並且依照 API 需求要帶上 參數 id
116        const vm = this;
117        const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/product/${id}`; // 多加 參數 id
118        vm.status.loadingItem = id; // 將原本的 vm.isLoading = true; ** 改為由 id 決定是哪個 查看更多被觸發
119        this.$http.get(url).then((response) => {
120          vm.product = response.data.product;
121          vm.product.num = 1;
122          $('#productModal').modal('show'); // 啟用 modal
123          console.log(response);
124          vm.status.loadingItem = ''; // 將原本的 vm.isLoading = false; ** 改成 如果讀取完要改成 空的
125        });
126      },
127    },
128    created() {
129      this.getProducts();
130    },
131  },
132  </script>

```

以下是原本 全部都有 loading icon



修改後 完成畫面



11 - 104. 選購產品及加入購物車

目前畫面邏輯是 點擊 查看更多 會跳出 modal 讓你有 select 可以選數量 、 點擊加到購物車則是直接加一筆數量

<https://github.com/hexschool/vue-course-api-wiki/wiki/%E5%AE%A2%E6%88%B6%E8%B3%BC%E7%89%A9-%E5%85%8D%E9%A9%97%E8%AD%89%5D>

從以下 API 得知 要傳入 參數有：產品 ID 以及 產品數量 (但要注意是不傳入產品價格的，因為訪客如果可以傳入產品價格 那他將可能有可以串改價格的 風險)

所以通常只上傳 ID 與數量，金額部分則由 後端來做計算

```
加入購物車

[API]: /api/:api_path/cart
[方法]: post
[參數]: { "data": { "product_id": "-L9tH8jxVb2Ka_DYPwng", "qty": 1 } }
[成功回傳]:
{
  "success": true,
  "message": "已加入購物車",
  "data": {
    "product_id": "-L9tH8jxVb2Ka_DYPwng",
    "qty": 1,
    "coupon_code": "",
    "id": "-LA15v_2MhWeh3linQxx",
    "total": 600,
    "final_total": 600,
    "product": {
      "category": "衣服3",
      "content": "這是內容",
      "description": "Sit down please 名設計師設計",
      "id": "-L9tH8jxVb2Ka_DYPwng",
      "imageUrl": "test.testtest",
      "is_enabled": 1,
      "num": 1,
      "origin_price": 500,
      "price": 600,
      "title": "[賣]動物園造型衣服3",
      "unit": "個"
    }
  }
}
```

取得購物車列表

```
取得購物車列表

[API]: /api/:api_path/cart
[方法]: get
[成功回傳]:
{
  "success": true,
  "data": {
    "carts": [
      {
        "coupon": {
          "code": "testCode",
          "due_date": 6547658,
          "id": "-L9uIs5EfPibJpwvTMhN",
          "is_enabled": 1,
          "percent": 60,
          "title": "超級特惠價格"
        },
        "id": "-LATwKkmvteWHjkmtp6m",
        "product": {
          "category": "衣服3",
          "content": "這是內容",
          "description": "Sit down please 名設計師設計",
          "id": "-L9tH8jxVb2Ka_DYPwng",
          "imageUrl": "test.testtest",
          "is_enabled": 1,
          "num": 1,
          "origin_price": 500,
          "price": 600,
          "title": "[賣]動物園造型衣服3",
          "unit": "個"
        },
        "product_id": "-L9tH8jxVb2Ka_DYPwng",
        "qty": 2
      },
      {
        "coupon": {
          "code": "testCode",
          "due_date": 6547658,
          "id": "-L9uIs5EfPibJpwvTMhN",
          "is_enabled": 1,
          "percent": 60,
          "title": "超級特惠價格"
        }
      }
    ]
  }
}
```

```

methods : {
  getProducts() {
    const vm = this;
    const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/products`;
    vm.isLoading = true;
    this.$http.get(url).then((response) => {
      vm.products = response.data.products;

      console.log(response);
      vm.isLoading = false;
    });
  },
  getProduct(id) { // 取得單一筆 資料，並且依照 API 需求要帶上 參數 id
    const vm = this;
    const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/product/${id}`; // 多加 參數 id
    vm.status.loadingItem = id; // 將原本的 vm.isLoading = true; ** 改為由 id 決定是哪個 查看更多被觸發
    this.$http.get(url).then((response) => {
      vm.product = response.data.product;
      vm.product.num = 1;
      $('#productModal').modal('show'); // 啟用 modal
      console.log(response);
      vm.status.loadingItem = '';
    });
  },
  addToCart(id , qty=1){ // 這裡將傳入裡參數 產品id、 數量(數量要給預設值 至少要 1 件)
    const vm = this;
    const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/cart`;
    vm.status.loadingItem = id; // 將原本的 vm.isLoading = true; ** 改為由 id 決定是哪個 查看更多被觸發
    const cart = { // 定義資料結構，將在下方的 post 做傳送到後端 (這裡便是要傳入的兩個參數)
      product_id : id ,
      qty : qty,
    }
    this.$http.post(url , {data : cart}).then((response) => {
      console.log(response);
      vm.status.loadingItem = ''; // 將原本的 vm.isLoading = false; ** 改成 如果讀取完要改成 空的
    });
  }
},

```

```

<div class="card-footer d-flex">
  <button
    type="button"
    class="btn btn-outline-secondary btn-sm"
    @click="getProduct(item.id)">
    <i
      class="fas fa-spinner fa-spin"
      v-if="status.loadingItem === item.id" ></i> 
    查看更多
  </button>
  <button
    type="button"
    class="btn btn-outline-danger btn-sm ml-auto"
    @click="addToCart(item.id)">
    <i
      class="fas fa-spinner fa-spin"
      v-if="status.loadingItem === item.id" ></i> 
    加到購物車
  </button>
</div>

```

The screenshot shows a product detail page for a watch named "Yosemite / 黑色 / 手鍊". The price is listed as 2300. A modal window is open, showing the JSON data sent to the server via a POST request to the "/api/jiaren-products/cart" endpoint:

```

{
  "data": {},
  "status": 200,
  "statusText": "",
  "headers": {},
  "config": {
    "url": "https://vue-course-api.hexschool.io/api/jiaren-products/cart",
    "method": "post",
    "data": {}
  },
  "data": {
    "final_total": 1400,
    "id": "-MCVH0Jzavy5kVtQxZc",
    "product": {
      "category": "手鍊",
      "content": "編織",
      "descripion": "酷",
      "id": "-MCNX5ZEvj2o7DoZtJX1",
      "image": "https://vue-course-api.hexschool.io/api/jiaren-products/cart/-MCVH0Jzavy5kVtQxZc/product_id/-MCNX5ZEvj2o7DoZtJX1/image",
      "qty": 1,
      "total": 1400
    },
    "message": "已加入購物車",
    "success": true
  },
  "headers": {
    "Content-Type": "application/json; charset=utf-8"
  },
  "request": XMLHttpRequest {
    readyState: 4,
    timeout: 0,
    withCredentials: true,
    upload: XMLHttpRequestUpload
  },
  "status": 200,
  "statusText": ""
}

```

Modal 加入

```
CustomerOrder.vue ×
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue" > template > div
49
50  <div class="modal fade" id="productModal" tabindex="-1" role="dialog"
51    aria-labelledby="exampleModalLabel" aria-hidden="true">
52    <div class="modal-dialog" role="document">
53      <div class="modal-content">
54        <div class="modal-header">
55          <h5 class="modal-title" id="exampleModalLabel">{{ product.title }}</h5>
56          <button type="button" class="close" data-dismiss="modal" aria-label="Close">
57            <span aria-hidden="true">&times;</span>
58          </button>
59        </div>
60        <div class="modal-body">
61          
62          <blockquote class="blockquote mt-3">
63            <p class="mb-0">{{ product.content }}</p>
64            <footer class="blockquote-footer text-right">{{ product.description }}</footer>
65          </blockquote>
66          <div class="d-flex justify-content-between align-items-baseline">
67            <div class="h4" v-if="!product.price">{{ product.origin_price }} 元</div>
68            <del class="h6" v-if="product.price">原價 {{ product.origin_price }} 元</del>
69            <div class="h4" v-if="product.price">現在只要 {{ product.price }} 元</div>
70          </div>
71          <select name="" class="form-control mt-3" v-model="product.num">
72            <option :value="num" v-for="num in 10" :key="num">
73              選購 {{num}} {{product.unit}}
74            </option>
75          </select>
76        </div>
77        <div class="modal-footer">
78          <div class="text-muted text nowrap mr-3">
79            小計 <strong>{{ product.num * product.price }}</strong> 元
80          </div>
81          <button
82            type="button"
83            class="btn btn-primary"
84            @click="addToCart(product.id , product.num)">
85            加到購物車
86          </button>
87        </div>
88      </div>
89    </div>
90  </div>
91
92 </div>
93 </template>
```

11-103.5 購物車列表 (課程未講解)

CustomerOrder.vue ● JS dev.env.js

```
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue" > script
```

```
89     |     </div>
90     |   </div>
91
92     <div class="my-5 row justify-content-center">
93       <div class="my-5 row justify-content-center">
94         <table class="table">
95           <thead>
96             <th></th>
97             <th>品名</th>
98             <th>數量</th>
99             <th>單價</th>
100
101           <tbody>
102             <tr v-for="item in cart.carts" :key="item.id" v-if="cart.carts">
103               <td class="align-middle">
104                 <button type="button" class="btn btn-outline-danger btn-sm">
105                   <i class="far fa-trash-alt"></i>
106                 </button>
107               </td>
108               <td class="align-middle">
109                 {{ item.product.title }}
110               </td>
111               <td class="align-middle">{{ item.qty }}/{{ item.product.unit }}</td>
112               <td class="align-middle text-right">{{ item.final_total }}</td>
113             </tr>
114           </tbody>
115           <tfoot>
116             <tr>
117               <td colspan="3" class="text-right">總計</td>
118               <td class="text-right">{{ cart.total }}</td>
119             </tr>
120           </tfoot>
121         </table>
122       </div>
123     </div>
124
125   </div>
126 </template>
```

CustomerOrder.vue ×

```
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue"
```

```
187   },
188   getCart(){
189     const vm = this;
190     const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/cart`;
191     vm.isLoading = true;
192     this.$http.get(url).then((response) => {
193       vm.cart = response.data.data;
194       // vm.products = response.data.products;
195       console.log("取得購物車",response);
196       vm.isLoading = false;
197     });
198   }
199 },
200 created() {
201   this.getProducts();
202   this.getCart();
203 },
204 }
</script>
```



Chain 防潑水長肩帶托特包

包

尼龍

4000

2000

查看更多

加到購物車

品名	數量	單價
Kalymnos / 黑色 / 金	1/件	1000
Yosemite / 黑色 / 亮黑	1/件	1400
極簡單綠包	1/件	2300
極簡單綠包	4/件	9200

```
top | Filter | Defa
⚠ Issues detected. The new Issues tab displays information about deprecations, breaking changes, and more.
▶ {data: {...}, status: 200, statusText: "", headers: {...}, config: {...}, ...} ⓘ
取得購物車
▼ {data: {...}, status: 200, statusText: "", headers: {...}, config: {...}, ...} ⓘ
  ▶ config: {url: "https://vue-course-api.hexschool.io/api/jiaren-products/cart"}
  ▶ data:
    ▶ carts: Array(24)
      ▶ 0: {...}
      ▶ 1: {...}
      ▶ 2: {...}
      ▶ 3: {...}
      ▶ 4: {...}
      ▶ 5: {...}
      ▶ 6: {...}
      ▶ 7: {...}
      ▶ 8: {...}
      ▶ 9: {...}
      ▶ 10: {...}
      ▶ 11: {...}
      ▶ 12: {...}
      ▶ 13: {...}
      ▶ 14: {...}
      ▶ 15: {...}
      ▶ 16: {...}
      ▶ 17: {...}
      ▶ 18: {...}
      ▶ 19: {...}
      ▶ 20: {...}
      ▶ 21: {...}
      ▶ 22: {...}
      ▶ 23: {...}
      length: 24
    ▶ __ob__: Observer {value: Array(24), dep: Dep, vmCount: 0}
    ▶ __proto__: Array
    final_total: (...)

  total: (...)

  ▶ __ob__: Observer {value: {...}, dep: Dep, vmCount: 0}
  ▶ get carts: f reactiveGetter()
  ▶ set carts: f reactiveSetter(newVal)
  ▶ get final_total: f reactiveGetter()
```

11 - 105. 刪除購物車品項及新增優惠碼

刪除某筆特定資料的 API

(路徑：將 API 路由帶入後 + 購物車 + ID)

刪除某一筆購物車資料

[API]: /api/:api_path/cart/:id

[方法]: delete

[成功回傳]:

```
{  
  "success": true,  
  "message": "已刪除"  
}
```

套用優惠券

參數是要上船的優惠馬，如果產品套用上優惠券 則價格也會有所調整。

套用後 整個購物車資料也是必須再重新取得

套用優惠券

percent 會轉數字格式

[API]: /api/:api_path/coupon

[方法]: post

[說明]: Coupon 套用時，全部統一套用

[參數]: @api_path: 'thisismycourse2'

```
{  
  "data": {  
    "code": "testCode"  
  }  
}
```

[成功回應]:

```
{  
  "success": true,  
  "message": "已套用優惠券:testCode",  
  "data": {  
    "final total": 2160  
  }  
}
```

[失敗回應]: 找不到優惠券

```
{  
  "success": false,  
  "message": "找不到優惠券!"  
}
```

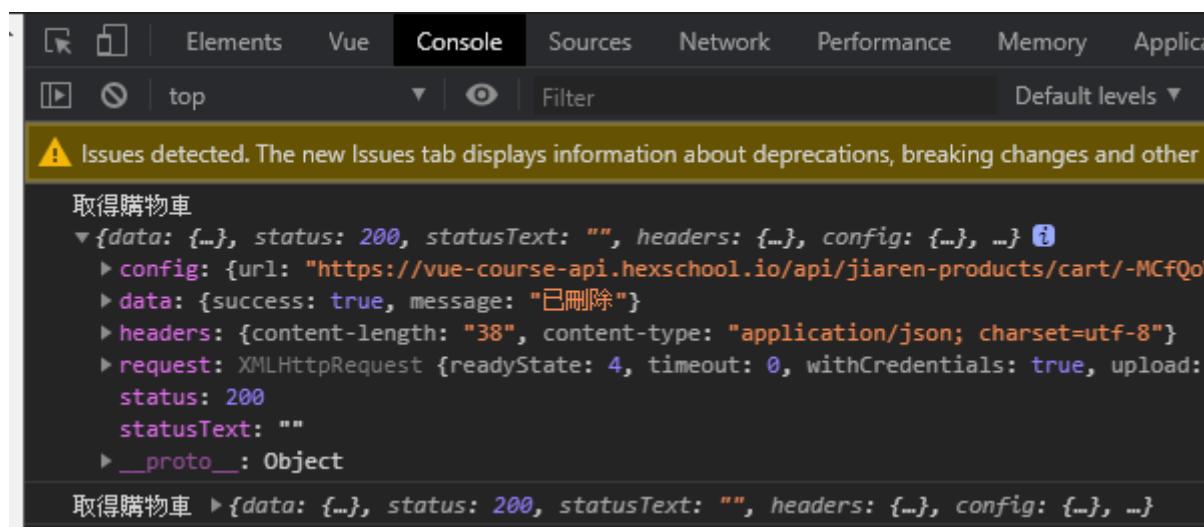
[失敗回應]: 過期

```
{  
  "success": false,  
  "message": "優惠券無法使用或已過期"  
}
```

刪除特定購物車內容：

使用 delete HTTP method 送出刪除的行為，後端便會將此資料給刪除

```
▼ CustomerOrder.vue ●
product > src > components > pages > ▼ CustomerOrder.vue > {} "CustomerOrder.vue"
  ...
191     getCart(){
192         const vm = this;
193         const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/cart`;
194         vm.isLoading = true;
195         this.$http.get(url).then((response) => {
196             |   vm.cart = response.data.data;
197             // vm.products = response.data.products;
198             console.log("取得購物車",response);
199             vm.isLoading = false;
200         });
201     },
202     removeCartItem(id){
203         const vm = this;
204         const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/cart/${id}`;
205         vm.isLoading = true;
206         this.$http.delete(url).then((response) => {
207             |   vm.getCart();                                // 刪除後重新取得購物車內容
208             |   console.log("取得購物車",response);
209             |   vm.isLoading = false;                      // 然後關閉讀取 loading 畫面
210         });
211     }
212 },
213 created() {
214     this.getProducts();
215     this.getCart();
216 },
217 }
218 </script>
```



以下 `<tr>` 的 `v-if` 會有紅字，主要是 `v-for` 與 `v-if` 共用會有的提示訊息。

```
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue" > template > div > div
  92 |   <div
  93 |     class="my-5 row justify-content-center"
  94 |     v-if="cart.total !== 0" <!-- 這裡判斷如果都沒加進購物車 則 此表格隱藏 -->
  95 |       <div class="my-5 row justify-content-center">
  96 |         <table class="table">
  97 |           <thead>
  98 |             <th></th>
  99 |             <th>品名</th>
 100 |             <th>數量</th>
 101 |             <th>單價</th>
 102 |           </thead>
 103 |           <tbody>
 104 |             <tr
 105 |               v-for="item in cart.carts"
 106 |               :key="item.id"
 107 |               v-if="cart.carts">
 108 |                 <td class="align-middle">
 109 |                   <button
 110 |                     type="button"
 111 |                     class="btn btn-outline-danger btn-sm"
 112 |                     @click="removeCartItem(item.id)">
 113 |                       <i class="far fa-trash-alt"></i>
 114 |                     </button>
 115 |                 </td>
 116 |                 <td class="align-middle">
 117 |                   {{ item.product.title }}
 118 |                 </td>
 119 |                 <td class="align-middle">{{ item.qty }}/{{ item.product.unit }}</td>
 120 |                 <td class="align-middle text-right">{{ item.final_total }}</td>
 121 |               </tr>
 122 |             </tbody>
 123 |             <tfoot>
 124 |               <tr>
 125 |                 <td colspan="3" class="text-right">總計</td>
 126 |                 <td class="text-right">{{ cart.total }}</td>
 127 |               </tr>
 128 |             </tfoot>
 129 |           </table>
 130 |         </div>
 131 |       </div>
```

品名	數量	單價
 Chain 防潑水長肩帶托特包	1/件	2000
 Chain 防潑水長肩帶托特包	1/件	2000
	總計	4000

接著進行優惠碼部分：

首先加上 優惠碼 template

```
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue" > template
92  <div
93    class="my-5 row justify-content-center"
94    v-if="cart.total !== 0"      <!-- 這裡判斷如果都沒加進購物車 則 此表格隱藏 --&gt;
95    &lt;div class="my-5 row justify-content-center"&gt;
96      &lt;table class="table"&gt;
97        &lt;thead&gt;
98          &lt;th&gt;&lt;/th&gt;
99          &lt;th&gt;品名&lt;/th&gt;
100         &lt;th&gt;數量&lt;/th&gt;
101         &lt;th&gt;單價&lt;/th&gt;
102       &lt;/thead&gt;
103       &lt;tbody&gt;
104         &lt;tr
105           v-for="item in cart.carts"
106           :key="item.id"
107           v-if="cart.carts"&gt;
108             &lt;td class="align-middle"&gt;
109               &lt;button
110                 type="button"
111                 class="btn btn-outline-danger btn-sm"
112                 @click="removeCartItem(item.id)"&gt;
113                   &lt;i class="far fa-trash-alt"&gt;&lt;/i&gt;
114                 &lt;/button&gt;
115             &lt;/td&gt;
116             &lt;td class="align-middle"&gt;
117               {{ item.product.title }}
118               &lt;div class="text-success" v-if="item.coupon"&gt;
119                 已套用優惠券
120               &lt;/div&gt;
121             &lt;/td&gt;
122             &lt;td class="align-middle"&gt;{{ item.qty }}/{{ item.product.unit }}&lt;/td&gt;
123             &lt;td class="align-middle text-right"&gt;{{ item.final_total }}&lt;/td&gt;
124           &lt;/tr&gt;
125         &lt;/tbody&gt;
126         &lt;tfoot&gt;
127           &lt;tr&gt;
128             &lt;td colspan="3" class="text-right"&gt;總計&lt;/td&gt;
129             &lt;td class="text-right"&gt;{{ cart.total }}&lt;/td&gt;
130           &lt;/tr&gt;
131           &lt;tr v-if="cart.final_total"&gt;
132             &lt;td colspan="3" class="text-right text-success"&gt;折扣價&lt;/td&gt;
133             &lt;td class="text-right text-success"&gt;{{ cart.final_total }}&lt;/td&gt;
134           &lt;/tr&gt;
135         &lt;/tfoot&gt;
136       &lt;/table&gt;
137
138       &lt;div class="input-group mb-3 input-group-sm"&gt;
139         &lt;input type="text" class="form-control" placeholder="請輸入優惠碼"&gt;
140         &lt;div class="input-group-append"&gt;
141           &lt;button class="btn btn-outline-secondary" type="button"&gt;
142             套用優惠碼
143           &lt;/button&gt;
144         &lt;/div&gt;
145       &lt;/div&gt;
146
147     &lt;/div&gt;
148   &lt;/div&gt;
149
150 &lt;/div&gt;
151 &lt;/template&gt;</pre>
```

套上後 目前畫面是如下 ,



品名	數量	單價
Chain 防潑水長肩帶托特包	1/件	2000
Chain 防潑水長肩帶托特包	1/件	2000
	總計	4000
	折扣價	4000

請輸入優惠碼

套用優惠碼

因為 還會正式加入優惠碼 , 所以 價格還是跟 總計一樣 ,
但目前先 寫入 v-if 判斷 讓 總計如果跟折扣價一樣時 , 折扣價 就不顯示

```
<tr v-if="cart.final_total !== cart.total">
  <td colspan="3" class="text-right text-success">折扣價</td>
  <td class="text-right text-success">{{ cart.final_total }}</td>
</tr>
```

然後開始補上優惠碼 部分：

API 顯示 要傳到後端的參數是 data 裡面包著 所提供的優惠碼

套用優惠券

percent 會轉數字格式

```
[API]: /api/:api_path/coupon  
[方法]: post  
[說明]: Coupon 套用時，全部統一套用  
[參數]: @api_path: 'thisismycourse2'  
{  
    "data": {  
        "code": "testCode"  
    }  
}  
[成功回應]:  
{  
    "success": true,  
    "message": "已套用優惠券:testCode",  
    "data": {  
        "final_total": 2160  
    }  
}  
[失敗回應]: 找不到優惠券  
{  
    "success": false,  
    "message": "找不到優惠券!"  
}  
[失敗回應]: 過期  
{  
    "success": false,  
    "message": "優惠券無法無法使用或已過期"  
}
```

首先 增加 優惠碼資料型態 再根據 api 寫入

```
CustomerOrder.vue <--  
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue"  
160 |  
161 <script>  
162 import $ from 'jquery';  
163  
164 export default {  
165   data() {  
166     return {  
167       products: [],  
168       product: {}, // 這裡是用來存 modal 的資料  
169       status: { // 新增一狀態 判斷目前畫面是哪個 查看更多按鈕 正在讀取中 (  
170         loadingItem: '' // 將會存放 點擊的產品 ID  
171       },  
172       cart: {},  
173       isLoading: false,  
174       coupon_code: '',  
175     }  
176   },  
177   methods: {  
178     getProducts() { ...  
179     },  
180     getProduct(id) { // 取得單一筆 資料，並且依照 API  
181     },  
182     addToCart(id, qty=1){ // 這裡將傳入裡參數 產品id、 數量  
183     },  
184     getCart(){ ...  
185     },  
186     removeCartItem(id){ ...  
187     },  
188     addCouponCode(){  
189       const vm = this;  
190       const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/coupon`;  
191       const coupon = {  
192         code: vm.coupon_code,  
193       };  
194       vm.isLoading = true;  
195       this.$http.post(url, {data: coupon}).then((response) => {  
196         vm.getCart(); // 刪除後重新取得購物車內容  
197         console.log("折扣碼", response);  
198         vm.isLoading = false; // 然後關閉讀取 loading 畫面  
199       });  
200     }  
201   }  
202 }  
203 </div>
```

Template 繩上 v-model 、 click 事件

```
139 <div class="input-group mb-3 input-group-sm">  
140   <input  
141     type="text"  
142     class="form-control"  
143     placeholder="請輸入優惠碼"  
144     v-model="coupon_code">  
145   <div class="input-group-append">  
146     <button  
147       class="btn btn-outline-secondary"  
148       type="button"  
149       @click="addCouponCode">  
150       套用優惠碼  
151     </button>  
152   </div>  
153 </div>
```

接著測試 優惠券效果：

如果隨便輸入會顯示 找不到優惠券

品名	數量	單價
Chain 防潑水長肩帶托特包	1/件	2000
Chain 防潑水長肩帶托特包	1/件	2000
	總計	4000

sss22

套用優惠碼

```
折扣碼
CustomerOrder.vue?155c:243
{
  data: {},
  status: 200,
  statusText: '',
  headers: {},
  config: {
    url: "https://vue-course-api.hexschool.io/api/jiaren-coupon"
  },
  data: {
    message: "找不到優惠券!",
    success: false
  }
}
```

如果到優惠券頁面 編輯優惠券 便可以正常使用了

The screenshot shows a modal window titled 'Modal title' with the following fields:

- 標題: 第一波優惠券
- 優惠碼: leon123
- 到期日: 2020/07/21
- 折扣百分比: 3
- 是否啟用

At the bottom right of the modal are two buttons: 'Close' and '更新優惠券' (Update Coupon).

The background shows the main 'Coupon Management' page with a sidebar menu and a table listing existing coupons.

The screenshot shows the 'Coupon Management' page with a table listing a single coupon:

名稱	折扣百分比	到期日	是否啟用	編輯
第一波優惠券	3%	Invalid Date	啟用	編輯

以下正常套用優惠券

品名	數量	單價
Chain 防潑水長肩帶托特包 已套用優惠券	1/件	60
Chain 防潑水長肩帶托特包 已套用優惠券	1/件	60
	總計	4000
	折扣價	120

leon123

套用優惠碼

```
⚠ Issues detected. The new Issues tab displays information about deprecations, breaking changes and other potential problems.
Go to Issues
CustomerOrder.vue?155c:243
{
  data: {},
  status: 200,
  statusText: '',
  headers: {},
  config: {
    url: "https://vue-course-api.hexschool.io/api/jiaren-products/coupon",
    method: "post",
    data: {
      final_total: 120
    }
  },
  data: {
    message: "已套用優惠券:leon123"
  }
}
```

以上有 以套用優惠券 綠色字樣

是 template 有寫入 v-if 效果：

```
<td class="align-middle">
  {{ item.product.title }}
  <div class="text-success" v-if="item.coupon">
    已套用優惠券
  </div>
</td>
```

另外 在 getCart 後 可以看出 console 裡面的細節，

有套上有惠碼的 購物車商品 將會多出 coupon 的增添細節

The screenshot shows a product card for a 'Chain 防潑水 長肩帶托特包' (Nylon) with a price of 2000. Below it is a summary table:

品名	數量	單價
Chain 防潑水長肩帶托特包 已套用優惠券	1/件	60
Chain 防潑水長肩帶托特包 已套用優惠券	1/件	60
	總計	4000
	折扣價	120

The screenshot shows the Vue DevTools inspecting the state of the 'carts' array in the 'CustomerOrder.vue' component. The array contains two items, each representing a product in the cart. The first item's product object has a 'coupon' key, indicating it has applied a coupon.

```
CustomerOrder.vue?155c:223
  ▶ {data: {...}, status: 200, statusText: "", headers: {...}, config: {...}, ...}
CustomerOrder.vue?155c:184
  ▶ {data: {...}, status: 200, statusText: "", headers: {...}, config: {...}, ...}
  ▶ config: {url: "https://vue-course-api.hexschool.io/api/jia..."}
  ▶ data:
    ▶ data:
      ▶ carts: Array(2)
        ▶ 0:
          ▶ coupon: Object
            final_total: 60
            id: "-MCFPLxIq550UXy8mzib"
          ▶ product: Object
            product_id: "-MCNVNIxPJzllH2Yom4e"
            qty: 1
            total: 2000
          ▶ __ob__: Observer {value: {...}, dep: Dep, vmCount: 0}
          ▶ get coupon: f reactiveGetter()
          ▶ set coupon: f reactiveSetter(newValue)
          ▶ get final_total: f reactiveGetter()
          ▶ set final_total: f reactiveSetter(newValue)
          ▶ get id: f reactiveGetter()
          ▶ set id: f reactiveSetter(newValue)
          ▶ get product: f reactiveGetter()
          ▶ set product: f reactiveSetter(newValue)
          ▶ get product_id: f reactiveGetter()
          ▶ set product_id: f reactiveSetter(newValue)
          ▶ get qty: f reactiveGetter()
          ▶ set qty: f reactiveSetter(newValue)
          ▶ get total: f reactiveGetter()
          ▶ set total: f reactiveSetter(newValue)
          ▶ __proto__: Object
        ▶ 1: {...}
          length: 2
        ▶ __ob__: Observer {value: Array(2), dep: Dep, vmCount: 0}
        ▶ __proto__: Array
        final_total: (...)
        total: (...)
      ▶ __ob__: Observer {value: f, dep: Dep, vmCount: 0}
```

11 - 106. *建立訂單及表單驗證技巧

建立訂單步驟：

1. 將會把原本所選的購物車內容全部刪掉
2. 然後成立成一筆新訂單，
3. 必且在訂單送出前 將用戶相關資訊給儲存下來
4. 用戶資訊也要填寫正確 才可送出

使用 vee-validate 這個套件 可以幫我們做 表單驗證 (此套件還有中文語系檔案) --- 先行下載安裝
npm i vee-validate@2.1.0-beta.9 –save (新版的有些問題)

```
[API]: /api/:api_path/order
[方法]: post
[說明]: 建立訂單後會把所選的購物車資訊刪除, message 欄位為必填, user 物件為必要
[參數]: @api_path: 'thisismycourse2'
{
  "data": {
    "user": {
      "name": "test",
      "email": "test@gmail.com",
      "tel": "0912346768",
      "address": "kaohsiung"
    },
    "message": "這是留言"
  }
}
[成功回傳]:
{
  "success": true,
  "message": "已建立訂單",
  "total": 100,
  "create_at": 1523539519,
  "orderId": "-L9tH8jxVb2Ka_DYPwng"
}
[失敗回傳]: message 未填寫
{
  "success": false,
  "messages": "說明欄位為必填"
}
[失敗回傳]: 無 user 物件
{
  "success": false,
  "message": "尚無用戶資料"
}
```

```
CustomerOrder.vue ● JS main.js ✘ Orders.vue
product > src > JS main.js > router.beforeEach() callback
1  import Vue from 'vue';
2  import axios from 'axios';
3  import VueAxios from 'vue-axios';
4  import 'bootstrap';
5  import Loading from 'vue-loading-overlay'; // Import component
6  import 'vue-loading-overlay/dist/vue-loading.css'; // Import stylesheet
7  import VeeValidate from 'vee-validate';
8  import zhTWValidate from 'vee-validate/dist/locale/zh_TW';
9
10
11 import App from './App';
12 import router from './router';
13 import './bus';
14 import currencyFilter from './filters/currency';
15 import dateFilter from './filters/date';
16
17 Vue.use(VueAxios, axios);
18 Vue.config.productionTip = false;
19
20 Vue.component('Loading', Loading); // 使用全域方式啟用 (這樣在每個不同地方就就不需要依依載入)
21 Vue.filter('currency', currencyFilter);
22 Vue.filter('date', dateFilter);
23
24 VeeValidate.Validator.localize('zh_TW', zhTWValidate);
25 Vue.use(VeeValidate);
26
27 axios.defaults.withCredentials = true;
```

```
CustomerOrder.vue X JS main.js Orders.vue
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue"
211
212  export default {
213    data() {
214      return {
215        products: [],
216        product : {}, // 這裡是用來存 modal 的資料
217        status:{}, // 新增一狀態 判斷目前畫面是哪個 查看更多按鈕 正在讀取中
218        loadingItem : '' // 將會存放 點擊的產品 ID
219      },
220      form: {
221        user: {
222          name: '',
223          email: '',
224          tel: '',
225          address: ''
226        },
227        message: '',
228      },
229      cart: {},
230      isLoading: false,
231      coupon_code : ''
232    }
233  },
234  methods : {
235    getProducts() { ... },
236    getProduct(id) { // 取得單一筆 資料，並且依照 API
237    },
238    addToCart(id , qty=1){ // 這裡將傳入裡參數 產品id、 數量
239    },
240    getCart(){...},
241    removeCartItem(id){...},
242    addCouponCode(){...},
243    createOrder() {
244      const vm = this;
245      const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/order`;
246      const order = vm.form;
247      // vm.isLoading = true;
248      this.$validator.validate().then((result) => {
249        if (result) {
250          this.$http.post(url, { data: order }).then((response) => {
251            console.log('訂單已建立', response);
252            // vm.getCart();
253            vm.isLoading = false;
254          });
255        } else {
256          console.log('欄位不完整');
257        }
258      });
259    },
260  },
261  created() {
262    this.getProducts();
263    this.getCart();
264  },
265}
266</script>
```

CustomerOrder.vue ● JS main.js Orders.vue

product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue" > template > div

```
154      </div>
155    </div>
156  </div>
157
158  <div class="my-5 row justify-content-center">
159    <form class="col-md-6" @submit.prevent="createOrder" >!-- 清除預設的 submit 行為 -->
160    <div class="form-group">
161      <label for="useremail">Email</label>
162      <input type="email" class="form-control" name="email" id="useremail"
163        v-validate="'required|email'"
164        :class="{'is-invalid': errors.has('email')}"
165        v-model="form.user.email" placeholder="請輸入 Email">
166      <span class="text-danger" v-if="errors.has('email')">
167        {{ errors.first('email') }}
168      </span>
169    </div>
170
171    <div class="form-group">
172      <label for="username">收件人姓名</label>
173      <input type="text" class="form-control" name="name" id="username"
174        :class="{'is-invalid': errors.has('name')}"
175        v-model="form.user.name" v-validate="'required'" placeholder="輸入姓名">
176      <span class="text-danger" v-if="errors.has('name')">姓名必須輸入</span>
177    </div>
178
179    <div class="form-group">
180      <label for="usertel">收件人電話</label>
181      <input type="tel" class="form-control" id="usertel"
182        v-model="form.user.tel" placeholder="請輸入電話">
183    </div>
184
185    <div class="form-group">
186      <label for="useraddress">收件人地址</label>
187      <input class="form-control" name="address"
188        :class="{'is-invalid': errors.has('address')}"
189        id="useraddress" v-model="form.user.address" v-validate="'required'"
190        placeholder="請輸入地址">
191      <span class="text-danger" v-if="errors.has('address')">地址欄位不得留空</span>
192    </div>
193
194    <div class="form-group">
195      <label for="usermessage">留言</label>
196      <textarea name="" id="" class="form-control" cols="30" rows="10"
197        v-model="form.message"></textarea>
198    </div>
199    <div class="text-right">
200      <button class="btn btn-danger">送出訂單</button>
201    </div>
202  </form>
203 </div>
204 </div>
205 </template>
```

品名	數量	單價
 Chain 防潑水長肩帶托特包	1/件	2000
	總計	2000

請輸入優惠碼

Email

v0416v@yahoo.com.tw

收件人姓名

ddd

收件人電話

0987654321

收件人地址

rrr

留言

rrrr

```
訂單已建立 CustomerOrder.vue?155c:313
{
  data: {...}, status: 200, statusText: "", headers: {...}, config: {...}, ...
  ▶ config: {url: "https://vue-course-api.hexschool.io/api/jiaren-..."}
  ▶ data:
    create_at: 1595294829
    message: "已建立訂單"
    orderId: "-MCiyoflC5dk-TrFznBx"
    success: true
    total: 2000
    ▶ __proto__: Object
  ▶ headers: {content-type: "application/json; charset=utf-8"}
  ▶ request: XMLHttpRequest {readyState: 4, timeout: 0, withCreden...
    status: 200
    statusText: ""
    ▶ __proto__: Object
}
```

結帳頁面

Is_paid : 確認是否付款了

取得某一筆訂單

```
[API]: /api/:api_path/order/:order_id
[參數]:
  @api_path: 'thisismycourse2'
  @order_id: '-L9u2EUkQSoEmW7QzGLF'

[方法]: get
[成功回傳]:
  {
    "success": true,
    "order": {
      "create_at": 1523539834,
      "id": "-L9u2EUkQSoEmW7QzGLF",
      "is_paid": false,
      "message": "這是留言",
      "payment_method": "credit_card",
      "products": [
        {
          "id": "L8nBrq8Ym4ARI1Kog4t",
          "product_id": "-L8moRfP1DZZ2e-1ritQ",
          "qty": "3"
        }
      ],
      "total": 100,
      "user": {
        "address": "kaohsiung",
        "email": "test@gmail.com",
        "name": "test",
        "tel": "0912346768"
      }
    }
  }
```

結帳付款

```
[API]: /api/:api_path/pay/:order_id
[方法]: post
[參數]:
  @api_path: 'thisismycourse2'
  @order_id: 訂單編號

[成功回傳]:
  {
    "success": true,
    "message": "付款完成"
  }
```

CustomerCheckout.vue

product > src > components > pages > CustomerCheckout.vue > {} "CustomerCheckout.vue" > template > e

```
1  <template>
2    <div class="my-5 row justify-content-center">
3      <form class="col-md-6" @submit.prevent="payOrder">
4        <table class="table">
5          <thead>
6            <th>品名</th>
7            <th>數量</th>
8            <th>單價</th>
9          </thead>
10         <tbody>
11           <tr v-for="item in order.products" :key="item.id">
12             <td class="align-middle">{{ item.product.title }}</td>
13             <td class="align-middle">{{ item.qty }}/{{ item.product.unit }}</td>
14             <td class="align-middle text-right">{{ item.final_total }}</td>
15           </tr>
16         </tbody>
17         <tfoot>
18           <tr>
19             <td colspan="2" class="text-right">總計</td>
20             <td class="text-right">{{ order.total }}</td>
21           </tr>
22         </tfoot>
23       </table>
24
25       <table class="table">
26         <tbody>
27           <tr>
28             <th width="100">Email</th>
29             <td>{{ order.user.email }}</td>
30           </tr>
31           <tr>
32             <th>姓名</th>
33             <td>{{ order.user.name }}</td>
34           </tr>
35           <tr>
36             <th>收件人電話</th>
37             <td>{{ order.user.tel }}</td>
38           </tr>
39           <tr>
40             <th>收件人地址</th>
41             <td>{{ order.user.address }}</td>
42           </tr>
43           <tr>
44             <th>付款狀態</th>
45             <td>
46               <span v-if="!order.is_paid">尚未付款</span>
47               <span v-else class="text-success">付款完成</span>
48             </td>
49           </tr>
50         </tbody>
51       </table>
52       <div class="text-right" v-if="order.is_paid === false">
53         <button class="btn btn-danger">確認付款去</button>
54       </div>
55     </div>
56   </template>
57
58
```

CustomerCheckout.vue ×

```
product > src > components > pages > CustomerCheckout.vue > {} "CustomerCheckout.vue" > template > div.my-5.row.jus
59  <script>
60  export default {
61    data() {
62      return {
63        order: {
64          user: {},
65        },
66        orderId: '',
67      };
68    },
69    methods: {
70      getOrder() {
71        const vm = this;
72        const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/order/${vm.orderId}`;
73        vm.isLoading = true;
74        this.$http.get(url).then((response) => {
75          vm.order = response.data.order;
76          console.log(response);
77          vm.isLoading = false;
78        });
79      },
80      payOrder() {
81        const vm = this;
82        const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/pay/${vm.orderId}`;
83        vm.isLoading = true;
84        this.$http.post(url).then((response) => {
85          console.log(response);
86          if (response.data.success) {
87            vm.getOrder();
88          }
89          vm.isLoading = false;
90        });
91      },
92    },
93    created() {
94      this.orderId = this.$route.params.orderId;
95      this.getOrder();
96      console.log(this.orderId);
97    },
98  };
99  </script>
```

CustomerCheckout.vue

CustomerOrder.vue

```
product > src > components > pages > CustomerOrder.vue > {} "CustomerOrder.vue" > script > data
290 >     addCouponCode(){...}
291
292     },
293     createOrder() {
294         const vm = this;
295         const url = `${process.env.APIPATH}/api/${process.env.CUSTOMPATH}/order`;
296         const order = vm.form;
297         // vm.isLoading = true;
298         this.$validator.validate().then((result) => {
299             if (result) {
300                 this.$http.post(url, { data: order }).then((response) => {
301                     console.log('訂單已建立', response);
302                     if (response.data.success) {
303                         vm.$router.push(`/customer_checkout/${response.data.orderId}`);
304                     }
305                     vm.isLoading = false;
306                 });
307             } else {
308                 console.log('欄位不完整');
309             }
310         });
311     },
312 },
313 created() {
314     this.getProducts();
315     this.getCart();
316 },
317 }
318 </script>
```

CustomerCheckout.vue

CustomerOrder.vue

JS router.js

product > src > JS router.js > ...

```
1 import Vue from 'vue';
2 import Router from 'vue-router';
3
4 // import Index from '@/components/Index';
5 import Login from '@/components/pages/Login';
6 import Dashboard from '@/components/Dashboard';
7 import Products from '@/components/pages/Products';
8 import Coupons from '@/components/pages/Coupons';
9 import Orders from '@/components/pages/Orders';
10 import CustomerOrder from '@/components/pages/CustomerOrder';
11 import CustomerCheckout from '@/components/pages/CustomerCheckout';
12
```

A screenshot of a code editor showing the `router.js` file. The file contains configuration for Vue.js routes. The code defines a main route for the dashboard and two nested routes under it: `CustomerOrder` and `CustomerCheckout`. The `CustomerCheckout` route includes a dynamic segment `:orderId`.

```
product > src > JS router.js > routes
  62   },
  63   {
  64     path: '/',
  65     name: 'Dashboard',
  66     component : Dashboard,
  67     children : [
  68       {
  69         path: 'customer_order',
  70         name: 'CustomerOrder',
  71         component: CustomerOrder,
  72       },
  73       {
  74         path: 'customer_checkout/:orderId',
  75         name: 'CustomerCheckout',
  76         component: CustomerCheckout,
  77       },
  78     ]
  79   }
  80 }
  81 })
```