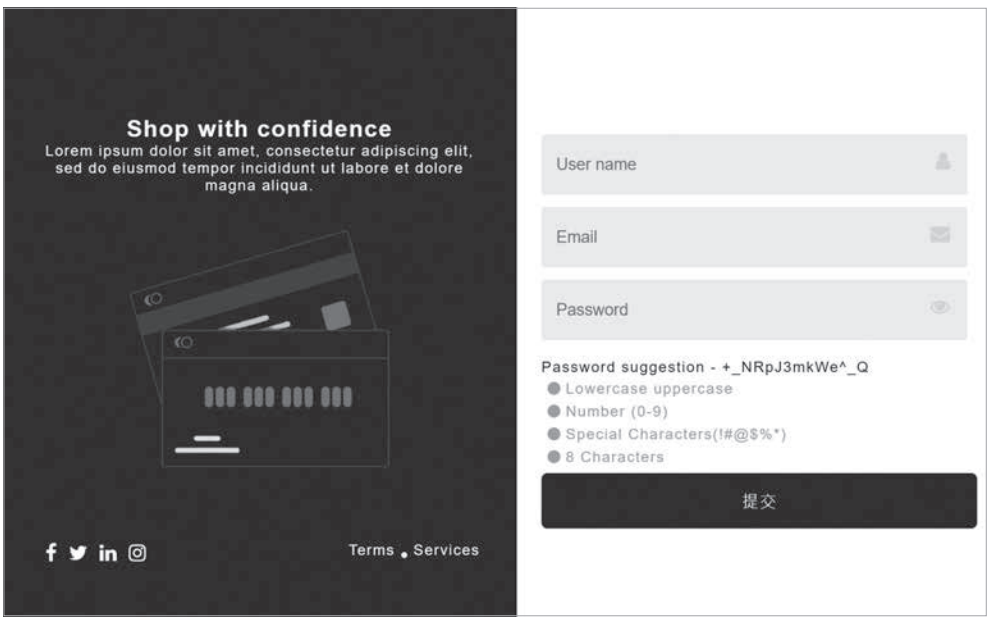
2

站在巨人肩膀上前進

本書認為很多情況下，程式碼不用自己一個字一個從頭寫起，而應採取「站在巨人的肩膀上」進行改寫。雖然上一章對此部分有改寫過，本章將進行更大幅度的改寫說明。

本章要用來說明改寫的範例採自 https://bbbootstrap.com/snippets/login-form-password-strength-checker-and-password-suggestion-47851744# 網頁的程式碼：



藉由改寫這個範例，恰好能將上一章所學做個回顧與複習。

2-1 結構分析

此範例很純粹地僅使用 HTML、CSS 及 JavaScript，因此，首先要準備一支 .html 檔或者使用本書準備的 vue02-01-001-01.html。

STEP 1 開啟各位自行準備的 .html 或是本書的 vue02-01-001-01.html。

STEP 2 切換到網站的 HTML 頁籤後，將其原始碼複製（註：可點選程式碼右上角的 copy 按鈕）到 <body> 標籤之後。

STEP 3 在 </head> 標籤前加入 <style></style> 標籤，接著切換到網站的 CSS 頁籤後，將其原始碼複製（註：可點選程式碼右上角的 copy 按鈕）到 <style> 標籤之後（詳 vue02-01-001-02.html）。

STEP 4 在 </body> 標籤前加入 <script></script> 標籤，接著切換到網站的 JAVASCRIPT 頁籤後，將其原始碼複製（註：可點選程式碼右上角的 copy 按鈕）到 <script> 標籤之後（詳 vue02-01-001-03.html）。

STEP 5 切換到網站的 RESOURCES 頁籤後，可以發現有二個 CDN，分別是 jQuery 及 Font Awesome。由於其 Font Awesome 是 4.7.0，因此，本例改寫成用第一章所提及的 6.4.0 版本，而相應的 HTML 標就必須同步修改（詳 vue02-01-001-04.html）。

```
<link
  rel="stylesheet"
  href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css"
integrity="sha512-iecdLmaskl7CVkqkXNQ/ZH/XLlvWZOJyj7Yy7tcenmpD1ypASozpmT/
E0iPtmFIB46ZmdtAc9eNBvH0H/ZpiBw==" crossorigin="anonymous" referrerpolicy="no-
referrer"
 />
```

social icons對應的原始碼，其中加上**<!-- -->**註解的是原4.7.1範例

```
<ul class="social-icons">
    <!-- <li><a href="#"><i class="fa fa-facebook"></i></a></li> -->
    <li><a href="#"><i class="fa-brands fa-facebook-f"></i></a></li>
    <!-- <li><a href="#"><i class="fa fa-twitter"></i></a></li> -->
    <li><a href="#"><i class="fa-brands fa-twitter"></i></a></li>
    <!-- <li><a href="#"><i class="fa fa-linkedin"></i></a></li> -->
    <li><a href="#"><i class="fa-brands fa-linkedin-in"></i></a></li>
    <!-- <li><a href="#"><i class="fa fa-instagram"></i></a></li> -->
    <li><a href="#"><i class="fa-brands fa-instagram"></i></a></li>
</ul>
```

如果各位選擇使用與範例相同的是 4.7.0，則可直接使用其中的 CDN 並保持原程式碼。

至於文字框右側的部分圖示，請參照前述的做法自行決定是否修改及改的方式。

STEP 6 原程式碼中關於 JavsScript 程式碼的部分，其結構如下。

① 利用 JavaScript 取得 HTML 物件及二個變數。除了其中的二個變數的位置改寫時不會被更動外，利用 JavaScript 取得 HTML 物件的程式碼後移置相關的 Vue 實例中。

```
const password_input = document.querySelector("#password_input");
const password_eye = document.querySelector("#password_eye");
let loweruppercase = document.querySelector(".loweruppercase i");
let loweruppercasetext = document.querySelector(".loweruppercase span");

let numbercase = document.querySelector(".numbercase i");
let numbercasetext = document.querySelector(".numbercase span");
let specialcase = document.querySelector(".specialcase i");
let specialcasetext = document.querySelector(".specialcase span");
```

```
let numcharacter = document.querySelector(".numcharacter i");
let numcharactertext = document.querySelector(".numcharacter span");

var password = document.getElementById('password_input');
let random_password = document.querySelector('#random_password');
var passwordLength = 14;
var passwordVal = "";
```

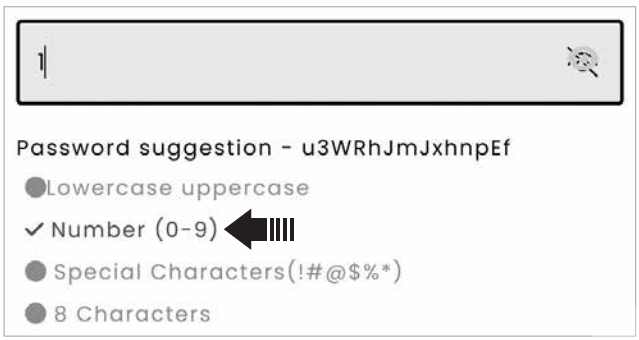
② 事件驅動處理程序的設置。這些事件處理程序係關於使用者介面中關於密碼輸入之用。

```
password_eye.addEventListener('click', () => {
});

password_input.addEventListener('keyup', function () {
});

password.addEventListener('focus', function () {
});
random_password.addEventListener('click', function () {
});
```

③ 搭配密碼的輸入，原範例設計有 passStrength(pass) 函式用以偵測使用目前輸的字元是否起過指定位數、是否輸入數字、是否輸入大寫及是否輸入特殊符。

```
function passStrength(pass) {
};
```

例如，使用者輸入數字 1，透過此函式則會在使用者介面出現相應的提示：



④ window.onload() 方法用於在網頁載入完畢後立刻執行的操作，此操作透過 loadPassword() 函式來達成。

```
window.onload = function loadPassword() {
  let randomGenerateChars = "B&vp3hSMQQsu#sR2+mTJx6kf6kHhHk^nNceWW_$=tEG#";

  for (var i = 0; i < passwordLength; i++) {
     let randomNumber=Math.floor(Math.random() * randomGenerateChars.length);
    passwordVal +=randomGenerateChars.substring(randomNumber, randomNumber + 1);
  }
  random_password.innerHTML="Password suggestion - " + passwordVal;
};
```

此函式的目的在於網頁載入之後，自動出現建議的隨機密碼：

2-1-1 改寫

STEP 7 設置 Vue 實例的掛載點（詳 vue02-01-001-05.html）。

```
<div id="app" class="container">
  <!-- Content here -->
  <div class="section">
    <div class="container">
      <div id="app" class=" form">
        <div class="left-side">
        </div>
```

```
            <div class="right-side">
            </div>
          </div>
        </div>
      </div>
    </div>
```

STEP 8 建構 Vue 實例及掛載（詳 vue02-01-001-06.html）。

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<!-- Vue 實例的程式碼 -->
<script>
  const app = Vue.createApp({

  })

  app.mount("#app")
</script>
```

STEP 9 將 window.onload() 方法的 loadPassword() 函式的程式碼寫到 Vue 實例的 mounted() 中，完成後再將原來的 window.onload() 方法對應的程式碼刪除（詳 vue02-01-001-07.html）。

```
<!-- Vue 實例的程式碼 -->
<script>
  const app = Vue.createApp({
    mounted() {
      let randomGenerateChars = "B&vp3hSMQQsu#sR2+mTJx6kf6kHhHk^nNceWW_$=tEG#";

      for (var i = 0; i < passwordLength; i++) {
        let randomNumber = Math.floor(Math.random() * randomGenerateChars.length);
        passwordVal += randomGenerateChars.substring(randomNumber, randomNumber + 1);
      }
      random_password.innerHTML = "Password suggestion - " + passwordVal;
    }
  })

  app.mount("#app")
</script>
```

上面程式中，用框線框起來的是 HTML 中用來提示建議密碼的 HTML 標籤物件。由於要由程式碼提「資料」給該 HTML 標籤，因此，有三件事要做：(一) Vue 實例中加入 data() 函式的 return 物件加入資料；(二) 在 HTML 標籤綁定該資料；(三) 改寫上述的程式碼。

```
<!-- Vue 實例的程式碼 -->
<script>
  const app = Vue.createApp({
    data() {
      return {
        random_password_innerHTML: ""
      }
    },
    mounted() {
      let randomGenerateChars = "B&vp3hSMQQsu#sR2+mTJx6kf6kHhHk^nNceWW_$=tEG#";

      for (var i = 0; i < passwordLength; i++) {
       let randomNumber = Math.floor(Math.random() * randomGenerateChars.length);
       passwordVal += randomGenerateChars.substring(randomNumber, randomNumber+ 1);
        console.log("passwordVal -- ", passwordVal)
      }
      // random_password.innerHTML = "Password suggestion - " + passwordVal;
      this.random_password_innerHTML = "Password suggestion - " + passwordVal;
    }
  })

  app.mount("#app")
</script>
```

接著再將原先的 進行綁定：

```
 <p id="random_password" class="random_password">
{{random_password_innerHTML }}
</p>
```

STEP 10 原程式碼中有關事件處理程序及用來偵測密碼輸入的 passStrength(pass) 函式都改寫到 Vue 實例中的 methods() 函式中，亦即複製原先的程式

到相應之 methods() 中，複製完之後再將這些原本的程式碼刪除。這些 methods 中函式的名稱係採用原程式碼所用的物件名稱，再搭配其事件處理程序的名稱為之（詳 vue02-01-001-08.html）。

```
<!-- Vue 實例的程式碼 -->
<script>
  const app = Vue.createApp({
    data() {
      return {
        random_password_innerHTML: ""
      }
    },
    methods: {
      password_eyeClick() {
      },
      password_inputKeyUp() {
      },
      passwordFocus() {
      },
      random_passwordClick() {
      },
      passStrength(pass) {
      }
    },
    mounted() {
    }
  })

  app.mount("#app")
</script>
```

接下來即是這些原本的程式碼及相應的 HTML 標籤進行改寫的的過程。

STEP 11 password_eyeClick() 的改寫（詳 vue02-01-001-09.html）。下面是原先的程式碼：

```
password_eyeClick() {
    if (password_input.type == "password") {
      password_input.type = "text";
```

```
        password_eye.classList.add("fa-eye");
        password_eye.classList.remove("fa-eye-slash");


    } else if (password_input.type == "text") {
        password_input.type = "password";
        password_eye.classList.add("fa-eye-slash");
        password_eye.classList.remove("fa-eye");
    }
}
```

從程式可知，其中使用了二個 HTML 標籤物件：password_input 及 password_eye，因此，將原程式碼中用來取得此二物件的程式碼複製進來：

```
password_eyeClick() {
    const password_input = document.querySelector("#password_input");
    const password_eye = document.querySelector("#password_eye");
    if (password_input.type == "password") {
        password_input.type = "text";
        password_eye.classList.add("fa-eye");
        password_eye.classList.remove("fa-eye-slash");


    } else if (password_input.type == "text") {
        password_input.type = "password";
        password_eye.classList.add("fa-eye-slash");
        password_eye.classList.remove("fa-eye");
    }
}
```

由於此 password_eyeClick() 係對應到 password_eye 物件的 Click 物件，因此，找出 HTML 中 id 是 password_eye 的位置，然後加上 v-on:click 或 @click。

```
<div class="form-inputs">
    <input id="password_input"
        class="password-input"
        autocomplete='chrome-off'
        type="password"
        placeholder="Password">
```

3 資料的呈現

網頁作為資料視覺化的載具，資料的呈現是網頁的基本，本章除了繼續延伸 data() 函式的說明外，會再談談 Options API 中的 template、computed、filters 及 watch 等屬性及搭配的 v-for 指令（directive）；至於用來控制 HTML 標籤做條件式資料呈現的 v-if 指令與 v-show 指令會在下一章說明。

3-1 選項物件的 template 屬性

前二章的例子中，都將 Vue 實例想要實現的使用者介面透過掛載點內的 HTML 標籤來呈現。不過，除了這個方式之外，Vue 實例想要呈現的 UI 內容也可以設計在 Vue 實例選項物件的 template 屬性，二者可以對比如下頁上方的對照。

下面程式碼對照的左側即為現行的做法，而右側則是將使用者介面移到 Vue 實例中的 template 屬性中進行設定：

使用掛載點

```
<!-- Vue實例的掛載點 -->
<div id='app'  >
    <h1>{{ message }}</h1>
</div>
```

```
const app = Vue.createApp({
    data() {
        return {
            message: "Vue.js 3"
        }
    }

})

app.mount("#app")
```

使用 template 屬性

```
<!-- Vue實例的掛載點 -->
<div id='app'  >

</div>
```

```
const app = Vue.createApp({
    data() {
        return {
            message: "Vue.js 3"
        }
    },
    template: '<h1>{{ message }}</h1>'
})

app.mount("#app")
```

接下來即是利用原先的 vue01-02-001.html 的程式碼進行改寫的過程。

STEP 1 複製 vue01-02-001.html 為 vue03-01-001-01.html。

STEP 2 將原先寫在 id 為 app 這個掛載點內的沒有使用 HTML 標籤的 {{ message }} 「註解掉」（詳 vue03-01-001-01.html）：

```
<!-- Vue 實例的掛載點 -->
<div id="app">
  <!-- {{ message }} -->
</div>
```

STEP 3 為 Vue 實例選項物件加上 template 屬性，並設定內含 {{ }} 鬍子語法的 <h1> 標籤構成的字串（詳 vue03-01-001-02.html）：

```
<!-- Vue 實例的程式碼 -->
<script>
  const app = Vue.createApp({
    data() {
      return {
        message: "Vue.js 3"
      }
    },
    template: '<h1>{{ message }}</h1>'
  })
```

此時若在瀏覽器開啟 vue03-01-002.html 網頁，便能從執行結果知道，原先寫在 id 為 app 這個掛載點內的沒有使用 {{ message }} 鬍子模版語法（在 Step2 時已被註解掉了），原先該語法的內容完成被 template 中指定的 <h1> 標籤的 <h1>{{ message }}</h1> 字串內容取代了！

上個範例只是使用簡單的一列字串來表示 template 的內容，如果要表達「多列」的字串內容的話，必須使用一對「``」，這個符號在鍵盤左上角數字 1 的左側。接下來這個範例利用多列的字完成一個 Bootstrap 5 的 Card 元件。

STEP 1 複製含有 Bootstrap 5 範本的 vue01-03-001-01.html 為 vue03-01-002-01.html。

STEP 2 加入與 Vue 相關的基本結構（詳 vue03-01-002-02.html）。

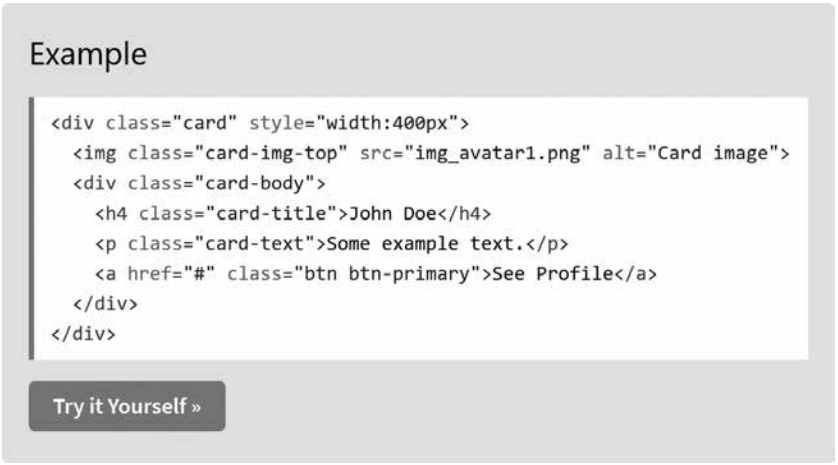
```
<body>
  <!-- Vue 實例的掛載點 -->
  <div id="app" class="container">
    <!-- Content here -->
  </div>

  <script src=……（省略）></script>

  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <!-- Vue 實例的程式碼 -->
  <script>
    const app = Vue.createApp({

    })

    app.mount("#app")
  </script>
</body>
```

STEP 3 準備一支用以作為大頭貼的圖檔，或者開啟 https://www.flaticon.com/free-icon/avatar_147144 網站下載免費的大頭貼圖案到與此 .html 檔相同的資夾中。

STEP 4 在 Vue 實例中加入 template 屬性，並設定其值為一對「``」（詳 vue03-01-002-03.html）。

```
<!-- Vue 實例的程式碼 -->
<script>
  const app = Vue.createApp({
    data() {
      return {
        message: "Vue.js 3"
      }
    },
    template: `

      `
  })

  app.mount("#app")
</script>
```

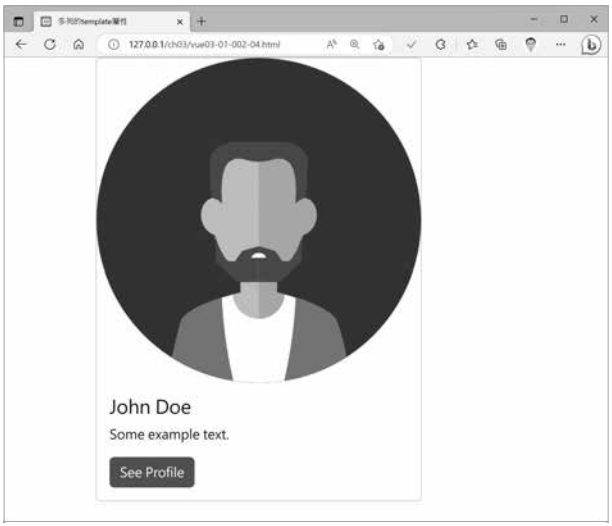
STEP 5 開啟 W3Schools 網站關於 Bootstraps 5 的 Cards 元件的範例程式碼網頁 https://www.w3schools.com/bootstrap5/bootstrap_cards.php，「複製」其範例程式碼（詳 vue03-01-002-04.html）：



然後「貼上」template 屬性的「``」中，最後將原來的 <img> 標中的 scr 屬性中的圖檔名稱修改為前面步驟下載的檔案名稱：

```
<!-- Vue 實例的程式碼 -->
<script>
  const app = Vue.createApp({
    template: `
  <div class="card" style="width:400px">
   <img class="card-img-top" src="147144.png" alt="Card image">
   <div class="card-body">
    <h4 class="card-title">John Doe</h4>
    <p class="card-text">Some example text.</p>
    <a href="#" class="btn btn-primary">See Profile</a>
   </div>
  </div>
        `

  })

  app.mount("#app")
</script>
```

完成後開啟 vue03-01-002-04.html，其執行結果如下：



如果要再加入一個 Card 元件的話，可以複製相同的程式碼到原先的 標之後，形成二個 構成的 Card（詳 vue03-01-002-05.html）：

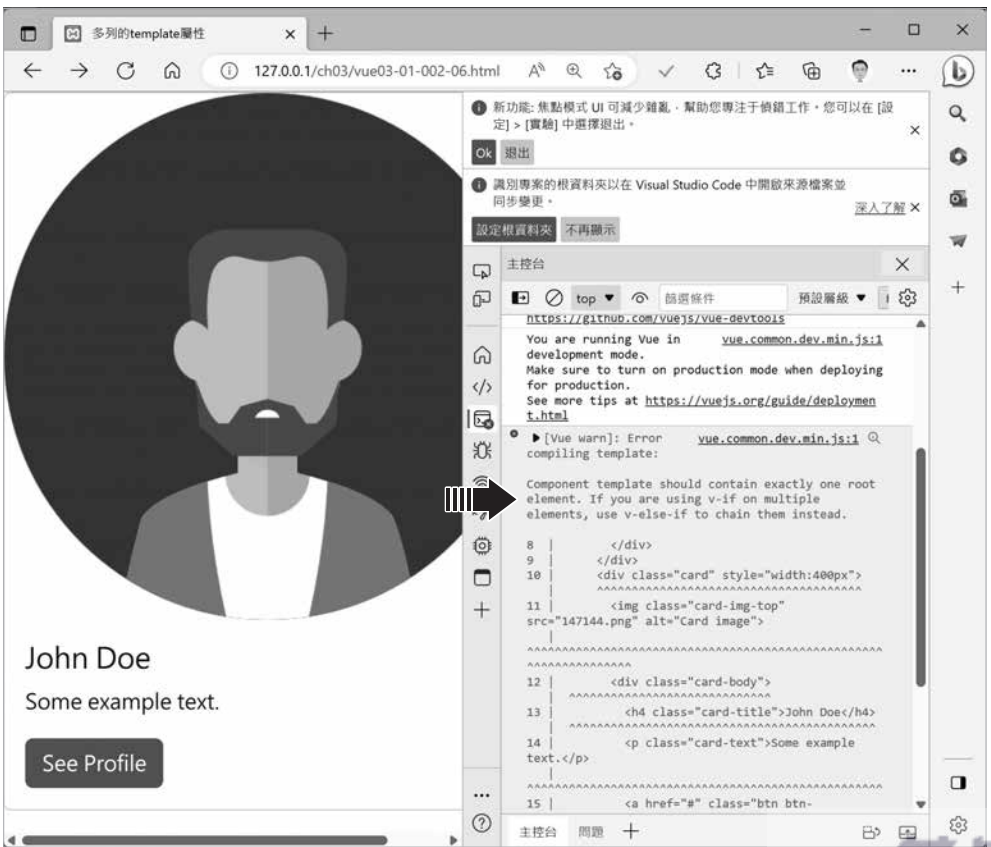
```
<!-- Vue 實例的程式碼 -->
<script>
  const app = Vue.createApp({
    template: `
  <div class="card" style="width:400px">
    <img class="card-img-top" src="147144.png" alt="Card image">
    <div class="card-body">
      <h4 class="card-title">John Doe</h4>
      <p class="card-text">Some example text.</p>
      <a href="#" class="btn btn-primary">See Profile</a>
    </div>
  </div>
  <div class="card" style="width:400px">
    <img class="card-img-top" src="147144.png" alt="Card image">
    <div class="card-body">
      <h4 class="card-title">John Doe</h4>
      <p class="card-text">Some example text.</p>
      <a href="#" class="btn btn-primary">See Profile</a>
    </div>
  </div>
```

```
    `
  })

  app.mount("#app")
</script>
```

此時若在瀏覽器中開啟 vue03-01-002-05 網頁，即可看到二個完全相同的 Card 元件。特別注意：在 Vue 2 的時候，同時有超過一個 HTML 標籤的 template 是會出錯的！（詳 vue03-01-002-06.html）

為什麼？

這是因為在使用 template 屬性時，所有內容要被包在一個單一的標籤之中才可以，此即下圖我們在 Console 頁籤中看到的錯誤訊息：template 中僅能有一個「根元素」，但本例卻有「二個元素」。



因此，修正的方式通常就是在原先的內容的最外圍包上一對的 與 標籤（詳 vue03-01-002-07.html）：

```
<!-- Vue 實例的程式碼 -->
<script>
  new Vue({
    el: '#app',
    template: `
<div>
  <div class="card" style="width:400px">
    <img class="card-img-top" src="147144.png" alt="Card image">
    <div class="card-body">
      <h4 class="card-title">John Doe</h4>
      <p class="card-text">Some example text.</p>
      <a href="#" class="btn btn-primary">See Profile</a>
    </div>
  </div>
  <div class="card" style="width:400px">
    <img class="card-img-top" src="147144.png" alt="Card image">
    <div class="card-body">
      <h4 class="card-title">John Doe</h4>
      <p class="card-text">Some example text.</p>
      <a href="#" class="btn btn-primary">See Profile</a>
    </div>
  </div>
</div>
    `
  })
</script>
```

雖然這樣就完成了將 HTML 使用者介面寫在選項物件 template 屬性的需求。

不過，這樣的設計在 Visual Studio Code 中會有點「小困擾」。困擾的原因是原先寫在 id 為 app 的 Vue 實例掛載點的 標籤中的程式碼，可以透過 Visual Studio Code 編輯器的協助（同時按下 Alt + Shift + F）下進行格式的編排，但是寫到 template 屬性後，這個格式編排功能就無法使用，開發人員必需自行排版。為了能同時使用 template 屬性也同時能使用格式編排功能，我們可以有二種替代的方式。

第一種方式，在 <body> 的 Vue 實例掛載點外的其他位置寫入一個 <template> 的標籤，然後將要呈現的內容寫入。

第二種方式，在 <body> 的 Vue 實例掛載點外的其他位置，寫入一個屬性為 type='x-template' 的 <script> 標籤，然後將要呈現的內容寫入。

使用 <template> 標籤

```
<body>
  <!-- Vue實例的掛載點 -->
  <div id='app'>

  </div>
  <!-- 供Vue實例的template屬性使用 -->
  <template id='...'>

  </template>
</body>
```

使用 <script> 標籤

```
<body>
  <!-- Vue實例的掛載點 -->
  <div id='app'>

  </div>
  <!-- 供Vue實例的template屬性使用 -->
  <script id='...' type='x-template'>

  </script>
</body>
```

不管那一種，都要為 <template> 或是 <script> 設定一個 id，這個 id 會寫到原先 Vue 實例的 template 屬性中，這樣就能為 Vue 實例的 template 屬性與 HTML 標籤的位置連繫起來了。這跟指定「掛載點」的做法是一樣的！

```
<!-- Vue實例的掛載點 -->
<div id='app'  >
</div>
<!-- 供Vue實例的template屬性使用 -->
<template id='ui'  ></template>
```

```
<!-- Vue實例的程式碼 -->
const app = Vue.createApp({
    template :  '#ui'
})

app.mount("#app")
```

接下來範例先使用第一種方式來實作。此範例來自 https://bbbootstrap.com/snippets/bootstrap-5-jobs-card-listing-59188500# 網站，共有三個 Card 元件：

STEP 1 將 vue01-03-001-01 此一 Bootstrap 5 的範本複製為 vue03-01-003-01.html。

STEP 2 加入 Vue 的基本結構（詳 vue03-01-003-02.html）。

```
<body>
  <!-- Vue 實例的掛載點 -->
  <div id="app" class="container">
    <!-- Content here -->
  </div>

  <script src=……（省略）></script>

  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

  <!-- Vue 實例的程式碼 -->
  <script>
    const app = Vue.createApp({

    })

    app.mount("#app")
  </script>
</body>
```

STEP 3 在 Vue 實例中加入 template 屬性，並指定其值為 '#ui'，注意，是使用字串的方式括起「#ui」，而不是用一對「``」喔，這裡的「#ui」是指一個 id 為 ui 的 HTML 的 <template> 標籤，因此一併設置 id 為 ui 的 <template> 標籤，而且 <template> 標籤「不要」放在「掛載點」之內（詳 vue03-01-003-03.html）：

```
<div id="app" class="container">
  <!-- Content here -->
</div>
<template id="ui">


</template>

<script src="……（省略）"></script>
```

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<!-- Vue 實例的程式碼 -->
<script>
  const app = Vue.createApp({
    template: '#ui'
  })

  app.mount("#app")
</script>
```

STEP 4 到範例的網頁後，切換到 HTML 頁籤並點選右上角的 copy 進行複製，然後再將剪下的內容「貼到」<template></template> 標籤中，這些 HTML 標籤是個利用 Bootstrap 5 每列以每三欄寬的方式放置 Card 元件，因此共有三個 Card 元件（詳 vue03-01-003-04.html）：

```
<template id="ui">
  <div class="container mt-5 mb-3">
    <div class="row">
      <div class="col-md-4">
        <div class="card p-3 mb-2">
        </div>
      </div>
      <div class="col-md-4">
        <div class="card p-3 mb-2">
        </div>
      </div>
      <div class="col-md-4">
        <div class="card p-3 mb-2">
        </div>
      </div>
    </div>
  </div>
</template>
```

STEP 5 一樣在到範例的官網，但切換到 CSS 頁籤並點選右上角的 copy 進行複製，然後到本例的 .html 檔案中的 </head> 前加入 <style></style> 標籤後，再將剪下的內容「貼到」該標籤中（詳 vue03-01-003-05.html）。

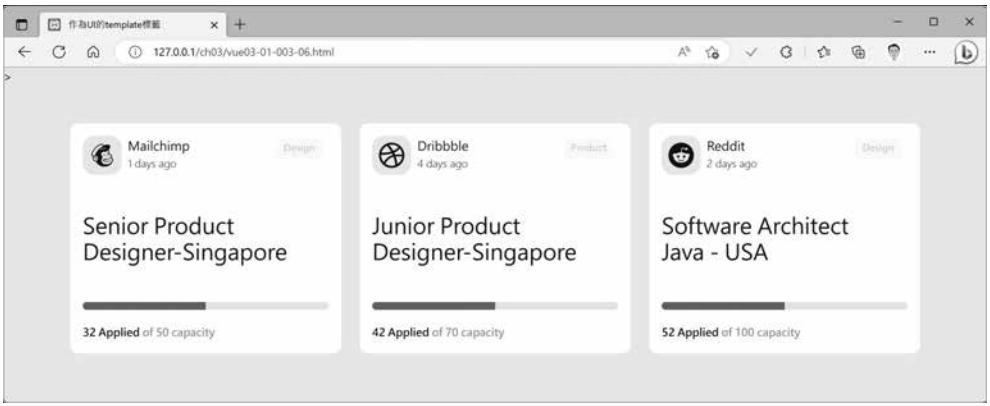
STEP 6 本例係使用 Bootstrap 5，因此，相關的 CDN 在 Step 01 中業已完成。不過，本例還使用了其他的 CDN，因此，一樣在到範例的網頁，但切換到 RESOURCES 頁籤，然後將最後二個關於 jQuery 及 boxicon 的 CDN 加入本例的 .html 檔案中（詳 vue03-01-003-06.html）。

① 先到 <style> 前分別加入下列二個 HTML 標籤：

```
<link rel="" />
<script src=""></script>
```

② 將 boxicon 的 CDN 複製到 <link> 的 rel 屬性。最後，將 jQuery 的 CDN 複製後貼到 <script> 的 src 屬性。

完成後開啟 vue03-01-003-06.html，其執行結果如下：



第二種方式只差在將第一種方式的 <template> 標籤換掉而已，所以，請先複製上一個範例 vue03-01-003-06.html 為 vue03-01-003-07.html，然後用下面內含 type 屬性的 <script> 標籤「換掉」原本的 <template> 標籤即可：

```
<script id='ui' type='x-template'>
  <div class="container mt-5 mb-3">
    <div class="row">
      <div class="col-md-4">
        <div class="card p-3 mb-2">
        </div>
      </div>
```

5

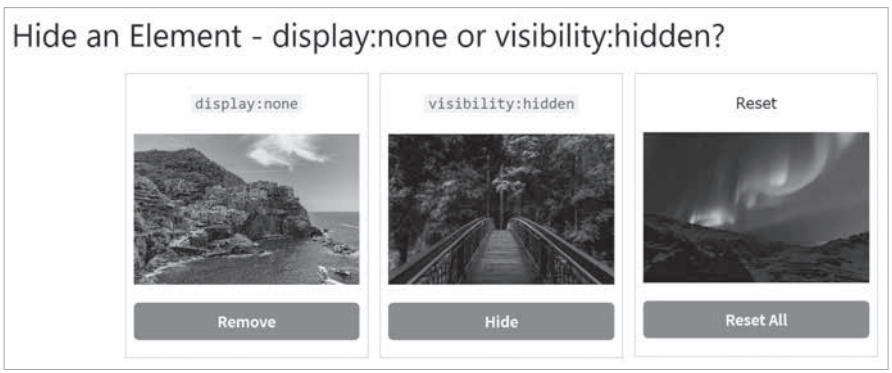
選擇性資料的呈現

關於資料與 CSS 樣式綁定後的呈現，在前面章節已有完整的說明，本章延續資料呈現的主題，不過其內容會是關於如何控制控制 HTML 標籤做「條件式」的選擇性資料呈現所會用到的二個指令：v-if 指令與 v-show 指令。

5-1 v-show 指令

v-show 指令的作用就是 show 秀出來的意思，至於秀與不秀則依賴某個特定條件而定。

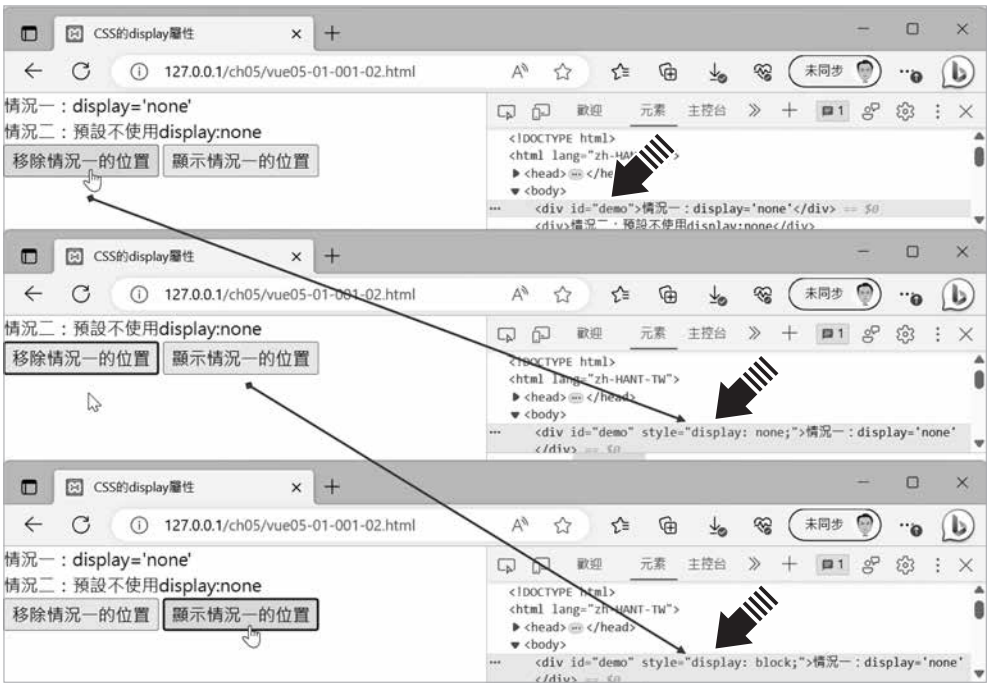
在 HTML 中，可以利用 CSS 中的 display 的值來控制是否顯示，例如，W3Schools 的網頁 [1] 有一個範例正好展現這樣的效果：如果 display 的值是 none 的話，該 HTML 的標籤完全不會出現在網頁中，類似的功能是使用 visibility 的值為 hidden，則該 HTML 標籤會佔用網頁中的位置，但不會顯示出來。



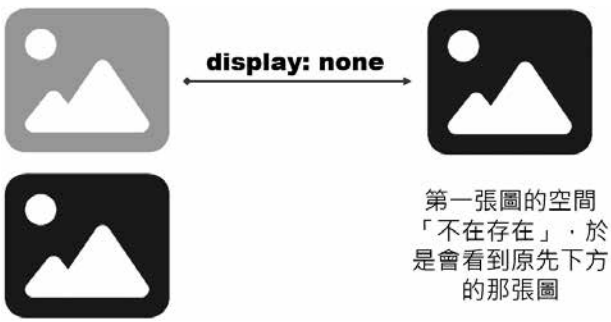
[1] 這個範例的網址是：https://www.w3schools.com/css/css_display_visibility.asp。

v-show 指令的作用是透過操弄 CSS 的 display 屬性值是否為 none 來控制已存在的內容（也就是說在 DOM 結構中已有的內容）是否要呈現出來（詳 vue05-01-001-01.html）。

如果模擬上述 W3Schools 範例並以 JavaScript 操作時，請參考 vue05-01-001-02.html 檔案的執行結果：



由上述執行的結果，描述其行為的示意圖如下：



接下來正式來說明 Vue 提供的 v-show 指令。v-show 指令的語法如下：

```
v-show=' 真假值 '
```

實際運用時，約莫有三種情形：

一、直接指定真假值為 true 或是 false，例如，v-show='true'，不過，通常不會直接指定固定的 true 或是 false，因為這樣就失去動態決定是否顯示特定內容的意義。

二、真假值與 Vue 實例的資料綁定在一起。例如，v-show='isShow'，以 isShow 的值做為決定顯示與否，而 isShow 的值只有二種：不是 true 就是 false，其邏輯很像流程控制的 if 敘述。

三、使用能產生真假的運算式，例如，v-show='item==1'，判斷 item 的值是否為某特定值，此時 item 的值可能有很多種，而我們想要判斷定它是屬於哪一種，此時能控制的情況就不會只限於二種而已，其邏輯很像流程控制的 switch 敘述。

下面範例針對上述四種情況簡單示範如下：

STEP 1 複製 vue01-template-03.html 為 vue05-01-002-01.html。

STEP 2 在 Vue 實例中設計如下，其中有二個 data 會在使用者介面中做資料綁定。其中 isShow 的值只有真或假二種，而 item 的值則可以有多種，至於有幾種則視情況而定（詳 vue05-01-002-01.html）：

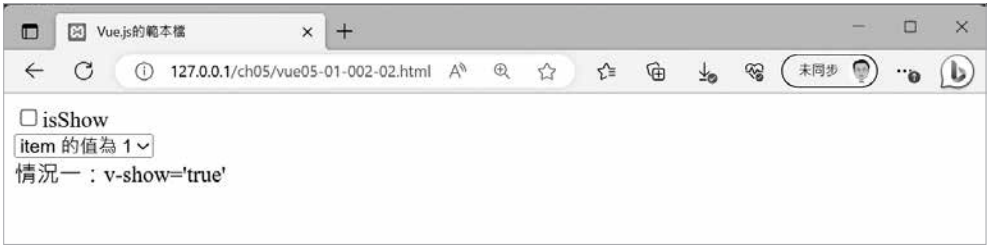
```
const app = Vue.createApp({
  data() {
    return {
      isShow: false,
      item: 1,
    };
  },
```

```
});
app.mount("#app");
```

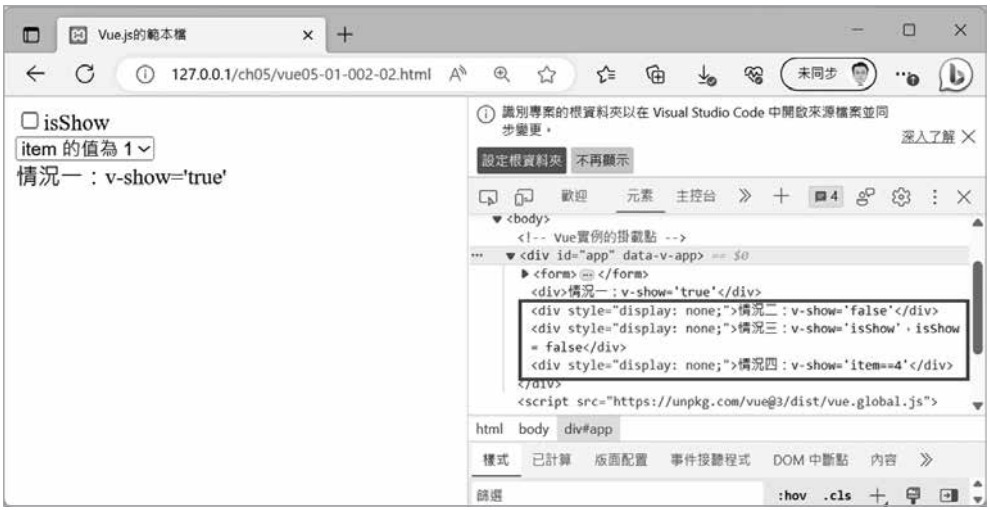
STEP 3 在 id 為 app 的 Vue 實例掛載點中，配合 isShow 的結果只有是與不是的二值結果，設計 checkbox，而配合 item 的值有多種可能，因此設計 <select> 標籤。另外，針對條件而決定呈現與否的部份，設計了四個使用 v-show 的 標籤（詳 vue05-01-002-02.html）：

```
<!-- Vue 實例的掛載點 -->
<div id='app'>
  <form>
    <input type='checkbox' v-model='isShow'>isShow
    <br />
    <select v-model='item'>
      <option value=1>item 的值為 1</option>
      <option value=2>item 的值為 2</option>
      <option value=3>item 的值為 3</option>
      <option value=4>item 的值為 4</option>
    </select>
  </form>

  <div v-show='true'>
    情況一：v-show='true'
  </div>
  <div v-show='false'>
    情況二：v-show='false'
  </div>
  <div v-show='isShow'>
    情況三：v-show='isShow'，isShow = {{ isShow }}
  </div>
  <div v-show='item==4'>
    情況四：v-show='item==4'
  </div>
</div>
```

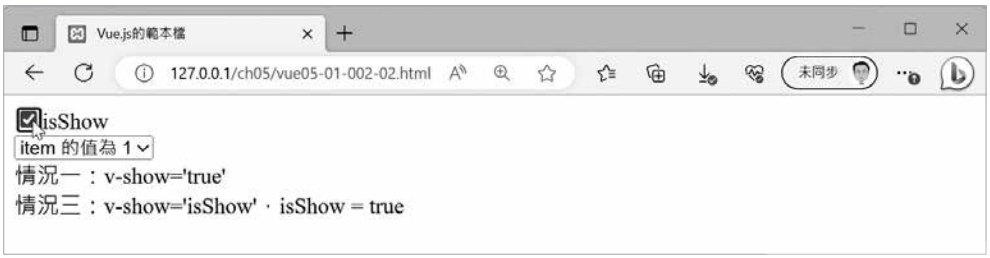
開啟 vue05-01-002-02.html 檔案，一開始，因為情況一的「v-show='true'」永遠為 true，所以，對應的 就會被秀出來，而情況二的「v-show='false'」則永遠為 false，因此對應的 標籤就不會被秀出來；與 Vue 實例進行 isShow 綁定的情況三與 item 資料綁定的情況四，因為 isShow 為 false，而 item 的初始值為 1，因此對應的 標籤一開始都不會被顯示出來：



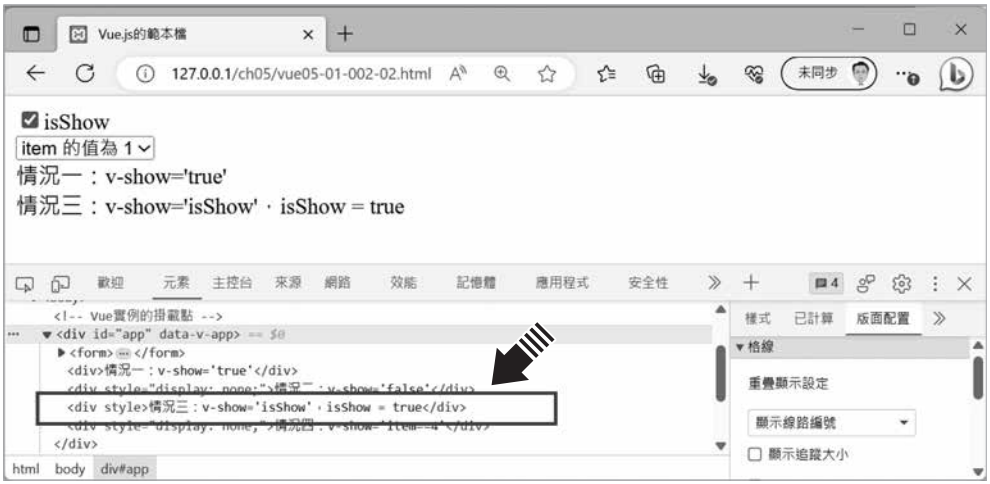
開啟瀏覽器的開發人員工具時，不難發現只要 v-show 的值為 false 的情況下，該 標籤 dsiplay 的值皆是 none，也就是因為這個值，所以對應的 標籤內容雖然已經存在於 DOM 中，但是卻不會被秀出來！



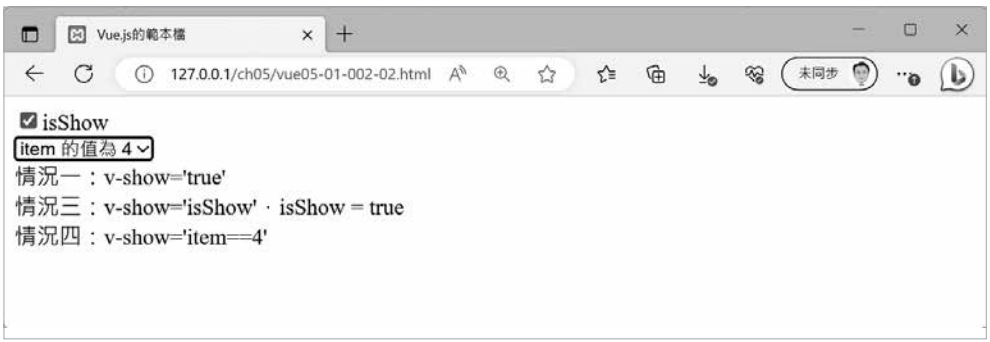
如果點選 isShow 的 check box，則會切換 isShow 的值，因此會由預設的 false 切換為 true，因此情況三的 isShow 的值為 true，則對應到情況三的 就會被秀出來，故會產生下面的結果：



此時再觀察元素頁籤，我們會發現原先情況三的 的 display:none 的值已經不存在了，僅餘 style！



如果點選 select 的值為 4 時，則會切換 isShow 的值，情況四的 item==4 的運算結果為 true，故會產生下面的結果：



從 v-show 指令這樣的條件式控制來決定現有的 HTML 使用者介面的是否呈現的角度，我們來分析一下常見的導覽列或是選單這二種使用者介面。

這二種使用者介面，它們的行為模式就是選項被點選之後執行該選項想要執行的功能，該選項被點選就相當於其值為 true，而沒有被點選的選項其值為 false。依此行為模式，接下來我們利用使用 Bootstrap 5 的導覽列的範例加入 v-show 指令來簡單地實作一下導覽列的實際應用。