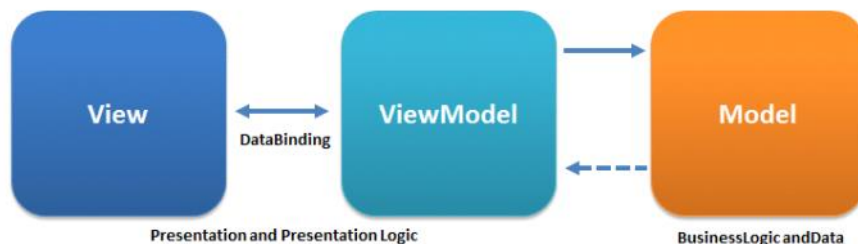


2. 綁定資料

2021年9月28日 下午 02:18

Vue JS 是以資料狀態渲染畫面!

MVVM (Model-view-viewmodel) 是一種軟體架構模式。



1. v-model 綁定元素裡面的值(value)與Vue裡面的資料(data)

2.

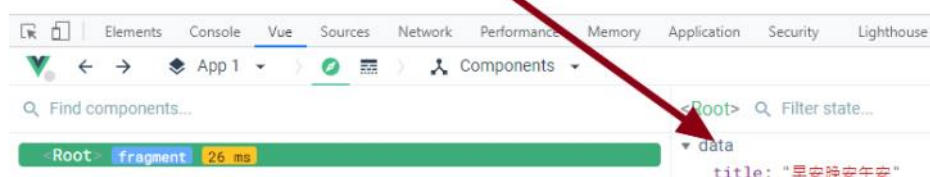
```
<div id="app">
  <h1>{{ title }}</h1>
  <input type="text" name="" id="" v-model="title">
</div>
<script>
  const App = {
    data() {
      return {
        title: '超強技術不囉說!'
      }
    },
  }
  Vue.createApp(App).mount('#app')
</script>
```

輸入的資料會自動綁定到vue的data裡面

← → ↻ ⓘ 127.0.0.1:5500/test02.html

早安晚安午安

早安晚安午安



v-text: 顯示純文字

v-html: 顯示HTML指令

```

<div id="app">
  <h1>{{ title }}</h1>
  <div v-text="text_html"></div>
  <div v-html="text_html"></div>
  <input type="text" name="" id="" v-model="title">
</div>
<script>
  const App = {
    data() {
      return {
        title: '超強技術不囉說!',
        text_html: '<h3>顯示HTML語法</h3>'
      }
    }
  }
  Vue.createApp(App).mount('#app')
</script>

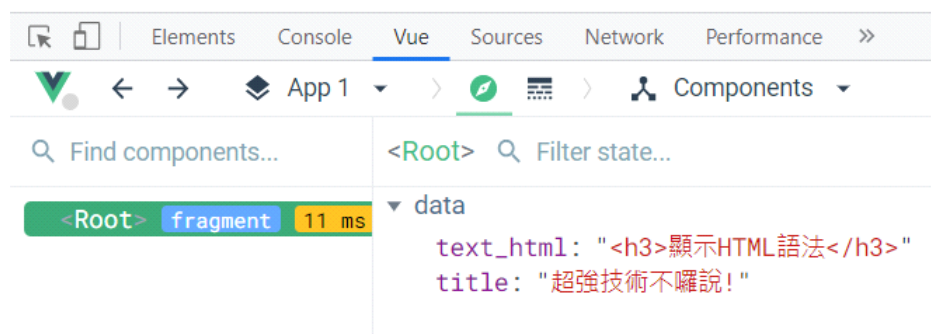
```

超強技術不囉說!

<h3>顯示HTML語法</h3>

顯示HTML語法

超強技術不囉說!



3. v-bind 將vue data渲染至畫面

渲染圖片

```

<div id="app">
  <h1>{{ title }}</h1>
  <div v-text="text_html"></div>
  <div v-html="text_html"></div>
  <input type="text" name="" id="" v-model="title">
  <hr>
  
</div>
<script>
  const App = {
    data() {
      return {
        title: '超強技術不囉說!',
        text_html: '<h3>顯示HTML語法</h3>',
        imgSrc: 'images/clothes04.jpg'
      }
    }
  },
  Vue.createApp(App).mount('#app')

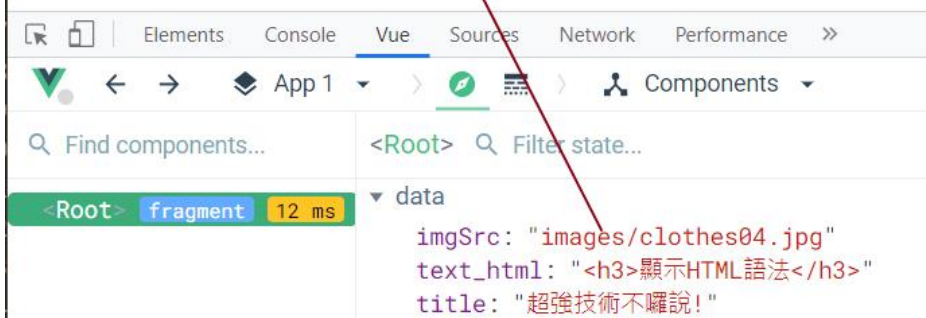
```

超強技術不囉說!

<h3>顯示HTML語法</h3>

顯示HTML語法

超強技術不囉說!



渲染class名稱

```

<div id="app">
  <h1>{{ title }}</h1>
  <div v-text="text_html"></div>
  <div v-html="text_html"></div>
  <input type="text" name="" id="" v-model="title">
  <hr>
  
</div>
<script>
  const App = {
    data() {
      return {
        title: '超強技術不囉說!',
        text_html: '<h3>顯示HTML語法</h3>',
        imgSrc: 'images/clothes04.jpg',
        className: 'w-50'
      }
    }
  },
  Vue.createApp(App).mount('#app')
</script>

```



觀察原始碼是否有渲染成功!


4. v-for 和 v-if

使用v-if 改變class狀態

```

<nav class="nav nav-pills nav-fill justify-content-center">
  <a class="nav-link" href="#" v-on:click="link = '草莓'" v-bind:class="{ 'active' : link === '草莓' }">草莓</a>
  <a class="nav-link" href="#" v-on:click="link = '香蕉'" v-bind:class="{ 'active' : link === '香蕉' }">香蕉</a>
  <a class="nav-link" href="#" v-on:click="link = '櫻桃'" v-bind:class="{ 'active' : link === '櫻桃' }">櫻桃</a>
</nav>

```



<Root> 🔍 Filter state...

- ▼ data
 - className: "w-50"
 - imgSrc: "images/clothes04.jpg"
 - link: "草莓"
 - ▼ list: Array[3]
 - ▶ 0: Object
 - ▶ 1: Object
 - ▶ 2: Object
 - text_html: "<h3>顯示HTML語法</h3>"
 - title: "超強技術不囉說!"

使用v-if v-else-if

```

<hr>
<nav class="nav nav-pills nav-fill justify-content-center">
  <a class="nav-link" href="#" v-on:click="link = '草莓'" v-bind:class="{ 'active' : link === '草莓' }">草莓</a>
  <a class="nav-link" href="#" v-on:click="link = '香蕉'" v-bind:class="{ 'active' : link === '香蕉' }">香蕉</a>
  <a class="nav-link" href="#" v-on:click="link = '櫻桃'" v-bind:class="{ 'active' : link === '櫻桃' }">櫻桃</a>
</nav>
<div>
  <div v-if="link === '草莓'">草莓很甜!</div>
  <div v-else-if="link === '香蕉'">香蕉超級甜!</div>
  <div v-else-if="link === '櫻桃'">櫻桃無敵甜!</div>
</div>

```

草莓 香蕉 櫻桃

香蕉超級甜!

0 - 一斤 草莓 - 100 元
1 - 一斤 香蕉 - 200 元

className: "w-50"

imgSrc: "images/clothes04.jpg"

link: "香蕉"

▼ list: Array[3]

- ▶ 0: Object
- ▶ 1: Object
- ▶ 2: Object

text_html: "<h3>顯示HTML語法</h3>"

title: "超強技術不囉說!"

以json格式的資料


```

<script>
  const App = {
    data() {
      return {
        title: '超強技術不囉說!',
        text_html: '<h3>顯示HTML語法</h3>',
        imgSrc: 'images/clothes04.jpg',
        className: 'w-50',
        list: [
          {
            product: '草莓',
            price: 100
          },
          {
            product: '香蕉',
            price: 200
          },
          {
            product: '櫻桃',
            price: 300
          }
        ]
      }
    }
  }
  Vue.createApp(App).mount('#app')
</script>

```

以v-for 渲染至畫面

```

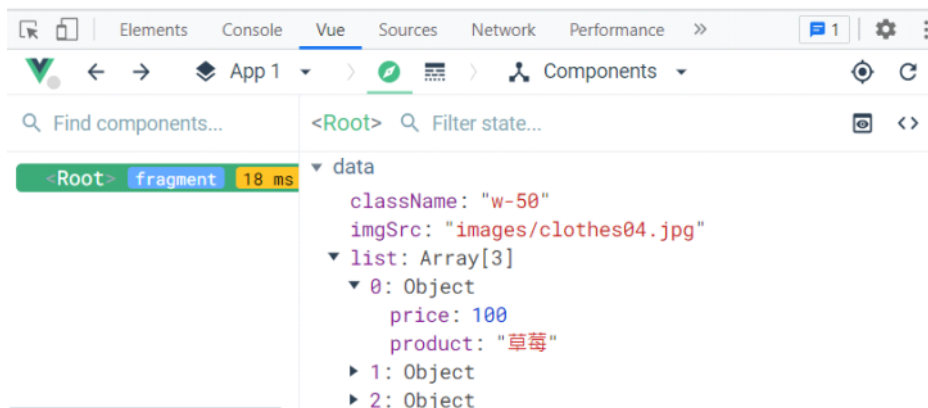
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action" v-for="item in list">
    {{ item.product }}
  </a>
</div>

```

草莓

香蕉

櫻桃



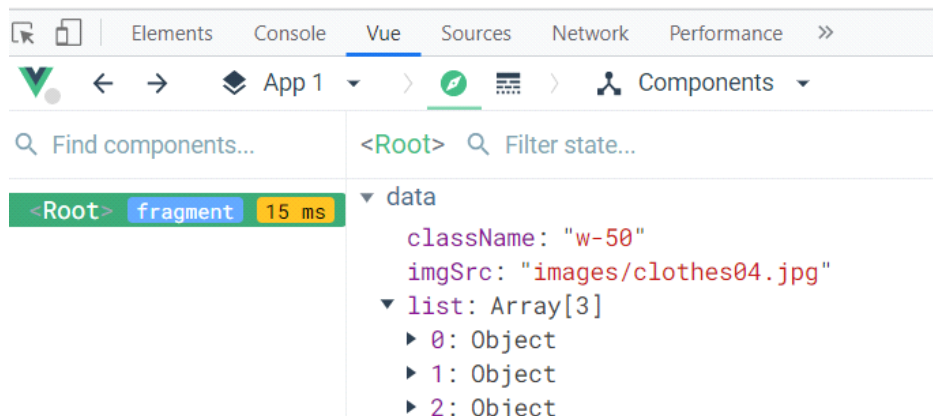
搭配索引值，索引值從0開始

```
<div class="list-group">
  <a href="#" class="list-group-item list-group-item-action" v-for="(item, key) in list">
    {{ key }} - 一斤 {{ item.product }} - {{ item.price }} 元
  </a>
</div>
```

0 - 一斤 草莓 - 100 元

1 - 一斤 香蕉 - 200 元

2 - 一斤 櫻桃 - 300 元

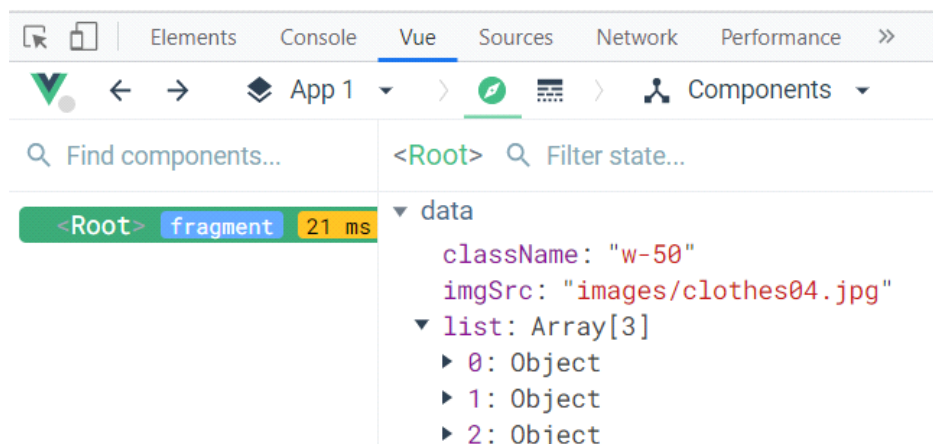


搭配 **v-if** 判斷，再經由**v-if**回傳**true** 或 **false**來決定是否渲染。**v-if** 和 **v-for** 不可寫在同一個標籤！

```
<div class="list-group">
  <template v-for="(item, key) in list">
    <a href="#" class="list-group-item list-group-item-action"
      v-if="item.price < 250">
      {{ key }} - 一斤 {{ item.product }} - {{ item.price }} 元
    </a>
  </template>
</div>
```

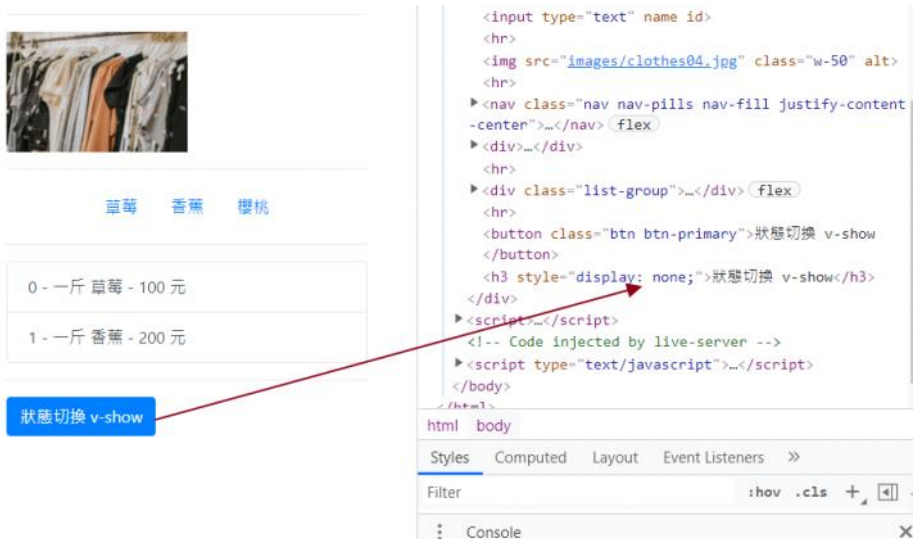
0 - 一斤 草莓 - 100 元

1 - 一斤 香蕉 - 200 元



v-show 使用 display: none

v-if 標籤直接消失



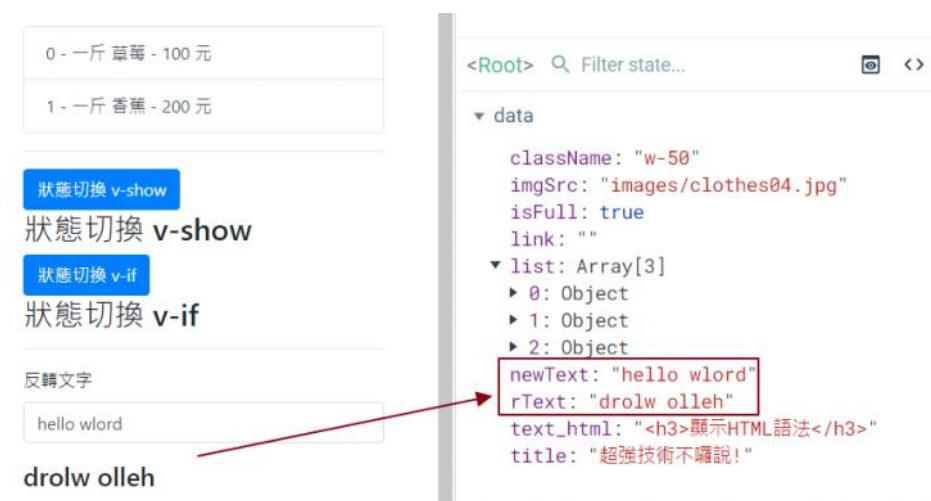
5. v-on 呼叫methods 產生資料互動

自定義reverseText()

```
methods: {
  reverseText() {
    this.rText = this.newText.split('').reverse().join('');
  },
}
```

v-on 監聽 change 事件 觸發 reverseText()

```
<div class="form-group">
  <label for="">反轉文字</label>
  <input type="text" class="form-control" v-model="newText" v-on:change="reverseText">
</div>
<h4>{{ rText }}</h4>
```



6. 修飾符

v-on:keyup.enter="reverseText"
鬆開enter鍵時執行reverseText()

```
<div class="form-group">
  <label for="">反轉文字</label>
  <input type="text" class="form-control" v-model="newText" v-on:keyup.enter="reverseText">
</div>
<h4>{{ rText }}</h4>
```


參考官網資料

<https://cn.vue.js.org/v2/guide/events.html#%E7%B3%BB%E7%BB%9F%E4%BF%AE%E9%A5%B0%E9%94%AE>



縮寫

v-on =====> @

v-bind =====> :

```
<div id="app">
  {{ text }}
  <div v-text="text"></div>
  <div v-html="text01"></div>
  <input type="text" name="" v-model="text" @keyup.enter="reverseText">
  <button class="btn btn-primary" @click="reverseText()">反轉</button>
  <div> {{ newText }} </div>
  
  <ul class="list-group">
    <li class="list-group-item" v-for="(item, key) in list" v-if="item.price < 80">
      {{ key }} - 一斤{{ item.product }} - {{ item.price }} 元
    </li>
  </ul>
</div>
```

7. 動態切換 classname

設定 rotate 旋轉45度

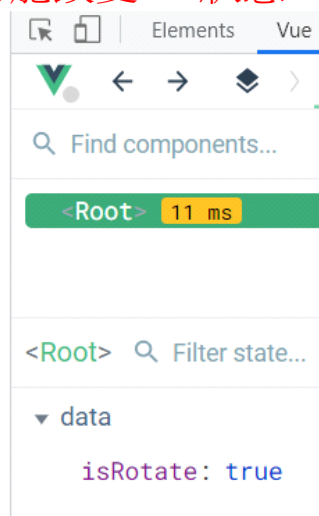
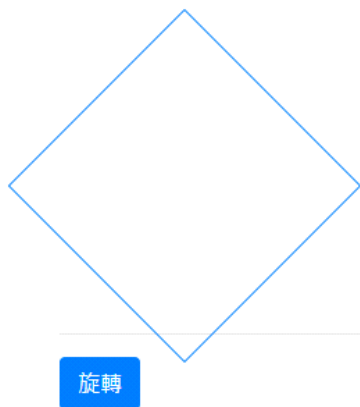
```
<style>
  .box{
    height: 180px;
    width: 180px;
    transition: transform .5s;
  }
  .rotate{
    transform: rotate(45deg);
  }
</style>
```

:class="{ '要加入的ClassName': 判斷式 }"

```
<div id="app" class="m-5">
  <div class="container">
    <div class="row">
      <div class="col-12">
        <div class="box border border-primary" :class="{ 'rotate': isRotate}"></div>
        <hr>
        <button class="btn btn-primary" @click="isRotate = !isRotate">旋轉</button>
      </div>
    </div>
  </div>
</div>
```

```
<script>
  const App = {
    data() {
      return {
        isRotate: false
      }
    },
  }
  Vue.createApp(App).mount('#app')
</script>
```

觀察 isrotated 布林值變化看是否能改變box狀態!



8. computed

使用按鈕監聽觸發反轉字串事件

```

<div class="col-8 offset-2">
  <div class="form-group">
    <label for="">反轉字串</label>
    <input type="text" class="form-control" placeholder="" v-model="newText">
  </div>
  <button class="btn btn-primary btn-block" @click="reverseText">反轉</button>
  <h4>反轉後: {{ reText }}</h4>
</div>

```

```

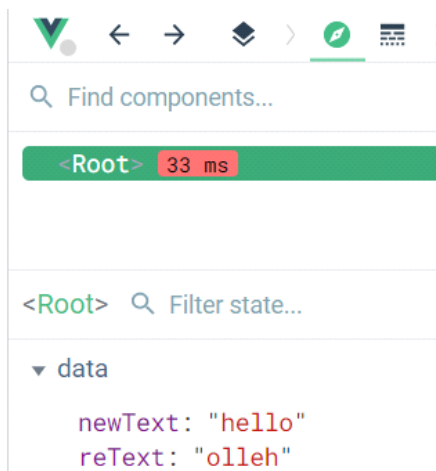
<script>
  const App = {
    data() {
      return {
        newText: '',
        reText: ''
      }
    },
    methods: {
      reverseText() {
        this.reText = this.newText.split('').reverse().join('');
      }
    },
  },
  Vue.createApp(App).mount('#app')
</script>

```

反轉字串

反轉

反轉後: olleh



改用**computed**會直接將結果存到變數，而此變數可以直接使用！

```

<script>
  const App = {
    data() {
      return {
        newText: '',
        reText: ''
      }
    },
    methods: {
      reverseText() {
        this.reText = this.newText.split('').reverse().join('');
      }
    },
    computed: {
      reverseText01() {
        return this.newText.split('').reverse().join('');
      }
    }
  }
  Vue.createApp(App).mount('#app')
</script>

```

```

<div class="col-8 offset-2">
  <div class="form-group">
    <label for="">反轉字串</label>
    <input type="text" class="form-control" placeholder="" v-model="newText">
  </div>
  <button class="btn btn-primary btn-block" @click="reverseText">反轉</button>
  <h4>反轉後: {{ reverseText01 }}</h4>
</div>

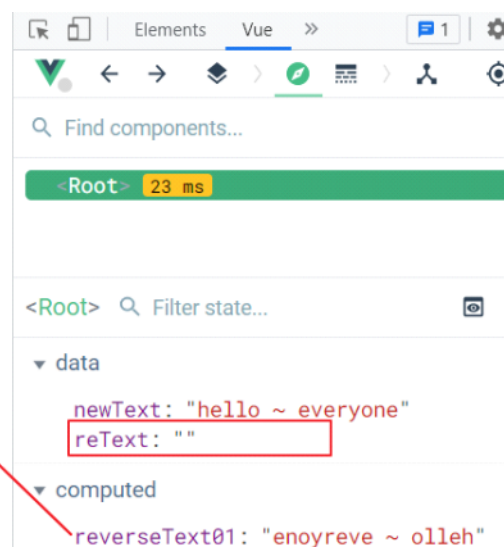
```

反轉字串

hello ~ everyone

反轉

反轉後: enoyreve
~ olleh



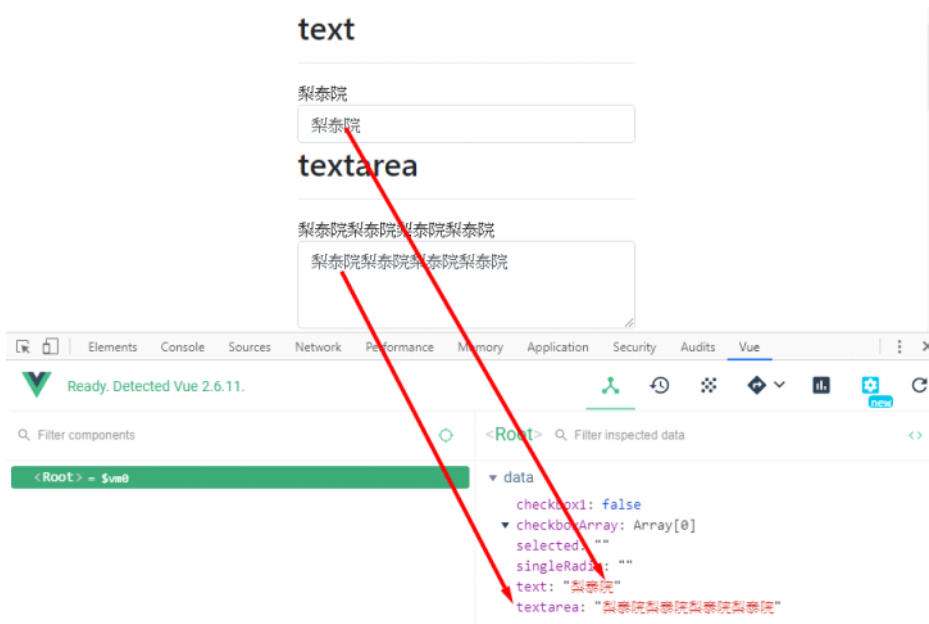
Methods 與 Computed 的使用情境

- (1) computed 是在監控資料更動後，重新運算結果呈現於畫面上一般來說不會修改資料，只會回傳用於畫面呈現的資料。
- (2) methods 就是互動的函式，需要觸發才會運作會用來修改資料內容。
- (3) 關於效能如果資料量大，computed 自然會比較慢只要資料變動就會觸發，無形之中執行次數也會增加，因此在大量資料時，會盡量透過 methods 減少不必要的運算。

9. 表單元素整理

text and textarea 綁定

```
<h2>text</h2>
<hr>
{{ text }}
<input type="text" class="form-control" v-model="text">
<h2>textarea</h2>
<hr>
{{ textarea }}
<textarea name="" id="" cols="30" rows="3" class="form-control"
v-model="textarea"></textarea>
```



Checkbox 單一筆資料綁定

```
<label class="form-check-label">
  <input type="checkbox" class="form-check-input" name="" id=""
  v-model="checkboxM">
  要吃麥當勞嗎?
</label>
```

☒ 要吃麥當勞嗎?

<Root> Filter state...

▼ data

checkboxM: true
newText: ""
reText: ""

▼ computed

reverseText01: ""

Checkbox 多筆資料綁定

```
<div class="form-check">
  <label class="form-check-label">
    <input type="checkbox" class="form-check-input" name="" id="" v-model="checkboxM">
      要吃麥當勞嗎?
    </label>
  </div>
<div class="form-check">
  <label class="form-check-label">
    <input type="checkbox" class="form-check-input" name="" id="" value="可樂" v-model="checkM_array">
      可樂
    </label>
  </div>
<div class="form-check">
  <label class="form-check-label">
    <input type="checkbox" class="form-check-input" name="" id="" value="漢堡" v-model="checkM_array">
      漢堡
    </label>
  </div>
<div class="form-check">
  <label class="form-check-label">
    <input type="checkbox" class="form-check-input" name="" id="" value="薯條" v-model="checkM_array">
      薯條
    </label>
  </div>
```

- ☒ 要吃麥當勞嗎?
- ☒ 可樂
- ☒ 漢堡
- ☒ 薯條

<Root> 🔍 Filter state...

▼ data

▼ checkM_array: Array[3]
0: "可樂"
1: "漢堡"
2: "薯條"
checkboxM: true
newText: ""
reText: ""

▼ computed

reverseText01: ""

v-for 顯示資料Checkbox所選取的資料

```
<ul class="list-group">
  <li class="list-group-item" v-for="(item, key) in checkM_array">
    {{ key+1 }} -- {{ item }}
  </li>
</ul>
```

- ☐ 要吃麥當勞嗎?
- ☒ 可樂
- ☒ 漢堡
- ☒ 薯條

1 -- 可樂

2 -- 漢堡

3 -- 薯條

Radio單筆資料綁定並顯示出

```
<div class="form-check">
  <label class="form-check-label">
    <input type="radio" class="form-check-input" name="" id="" value="現金" v-model="radioPay">
    現金
  </label>
</div>
<div class="form-check">
  <label class="form-check-label">
    <input type="radio" class="form-check-input" name="" id="" value="刷卡" v-model="radioPay">
    刷卡
  </label>
</div>
```

☐ 要吃麥當勞嗎?

☒ 可樂

☐ 漢堡

☒ 薯條

☐ 現金

☒ 刷卡

訂購餐點如下:

1 -- 可樂

2 -- 薯條

付款方式: 刷卡

<Root> 🔍 Filter state...

▼ data

▼ checkM_array: Array[2]

0: "可樂"

1: "薯條"

checkboxM: false

newText: ""

radioPay: "刷卡"

reText: ""

▼ computed

reverseText01: ""

select資料綁定並顯示出

```
data() {
  return {
    newText: '',
    reText: '',
    checkboxM: false, // 要吃麥當勞嗎
    checkM_array: [], // 餐點
    radioPay: '', // 付款方式
    selectSugar: '' // 甜度
  }
},
```

```
<div class="form-group">
  <label for="">飲料甜度</label>
  <select class="form-control" name="" id="" v-model="selectSugar">
    <option value="全糖">全糖</option>
    <option value="半糖">半糖</option>
    <option value="無糖">無糖</option>
  </select>
</div>
```

訂購餐點如下:

1 -- 可樂
2 -- 漢堡
3 -- 薯條

半糖

付款方式:

```
<Root> 🔍 Filter state...
▼ data
  ▼ checkM_array: Array[3]
    0: "可樂"
    1: "漢堡"
    2: "薯條"
  checkboxM: false
  newText: ""
  radioPay: ""
  reText: ""
  selectSugar: "半糖"
```

▼ computed

```
reverseText01: ""
```

利用v-if 來決定是否顯示訂餐選項

```
<div class="col-8 offset-2">
  <div class="form-check">
    <label class="form-check-label">
      <input type="checkbox" class="form-check-input" name="" id="" v-model="checkboxM">
        要吃麥當勞嗎?
    </label>
  </div>
  <div v-if="checkboxM">
    <div class="form-check">...
  </div>
  <div class="form-check">...
</div>
  <div class="form-check">...
</div>
  <div class="form-group">...
</div>
  <div class="form-check">...
</div>
  <div class="form-check">...
</div>
  <hr>
  <ul class="list-group">...
</ul>
</div>
</div>
```

☐ 要吃麥當勞嗎?

```
<Root>  🔍 Filter state...  
  ▼ data  
    ▼ checkM_array: Array[3]  
      0: "可樂"  
      1: "漢堡"  
      2: "薯條"  
    checkboxM: false  
    newText: ""  
    radioPay: "現金"  
    reText: ""  
    selectSugar: "半糖"  
  ▼ computed  
    reverseText01: ""
```

☒ 要吃麥當勞嗎?

☒ 可樂

☒ 漢堡

☒ 薯條

飲料甜度

半糖

☒ 現金

☐ 刷卡

訂購餐點如下:

1 -- 可樂

2 -- 漢堡

3 -- 薯條

半糖

付款方式: 現金

<Root> 🔍 Filter state...

▼ data

▼ checkM_array: Array[3]

0: "可樂"

1: "漢堡"

2: "薯條"

checkboxM: true

newText: ""

radioPay: "現金"

reText: ""

selectSugar: "半糖"

▼ computed

reverseText01: ""