FPT Education

FPT POLYTECHNIC

XÂY DỰNG GIAO DIỆN TƯƠNG TÁC BACKEND

BÀI 6:

CONDITIONAL RENDERING VÀ LIST RENDERING

www.poly.edu.vn

MỤC TIÊU

- Kết thúc bài học này bạn có khả năng
  - Nắm vững kiến thức Conditional rendering
  - Nắm vững kiến thức List rendering
  - Thực hành được v-if, v-else, v-else-if, v-show
  - Phân biệt trường hợp sử dụng v-if và v-show
  - Tìm hiểu v-for và các thành phần mở rộng

NỘI DUNG

- Phần I: Conditional Rendering trong VueJS
  - v-if
  - v-else, v-else-if
  - v-show
- Phần II: List Rendering trong VueJS
  - v-for và các thành phần mở rộng

FPT Education

FPT POLYTECHNIC

BÀI 6:

CONDITIONAL RENDERING VÀ LIST RENDERING

PHẦN I: CONDITIONAL RENDERING

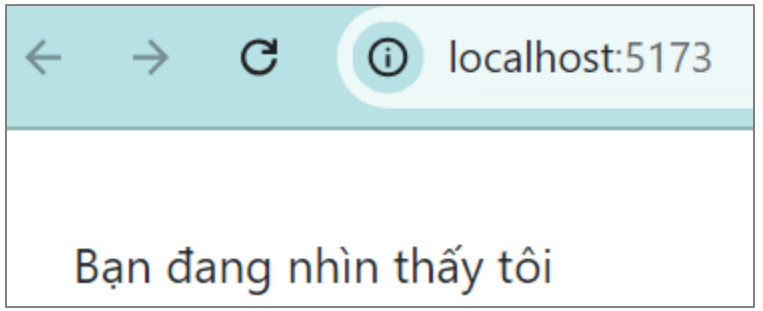
CONDITIONAL RENDERING

- **Conditional rendering** (Hiển thị có điều kiện)

Mục đích giúp hiển thị hoặc ẩn các phần tử trong DOM dựa trên các điều kiện cụ thể. VueJS cung cấp các phương thức sau để thực hiện điều này: **v-if**, **v-else-if**, **v-else** và **v-show**.

1. **v-if**: Kiểm tra điều kiện trước khi hiển thị ra. Ví dụ:

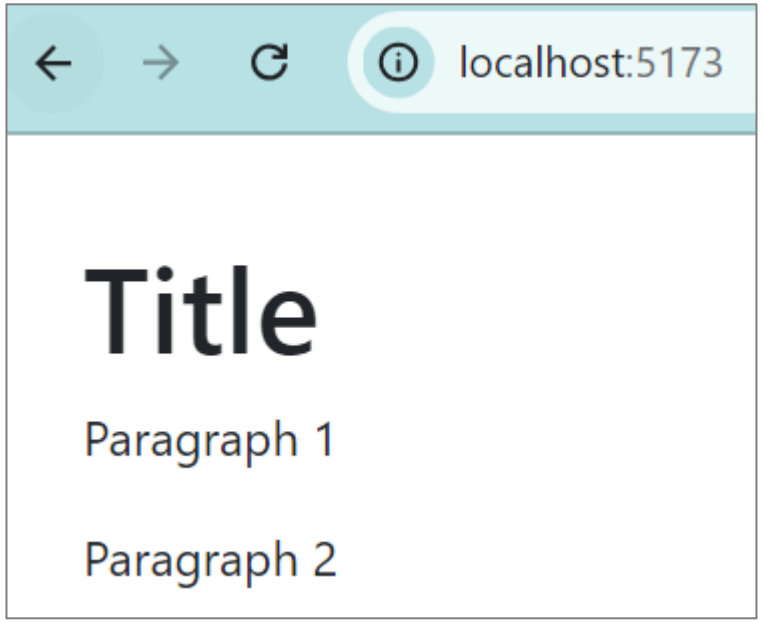
```
<template>
  <span v-if="seen">Bạn đang nhìn thấy tôi</span>
</template>
<script setup>
  import { ref } from 'vue';
  const seen = ref(true);
</script>
```



*Lúc này, nếu **seen** là **true** thì tag span sẽ được hiển thị và ngược lại, nếu là **false** thì tag span được ẩn đi.*

1. **v-if trên <template>:** trường hợp muốn áp dụng v-if với một nhóm các phần tử thì có thể làm như sau:

```
<template v-if="ok">
  <h1>Title</h1>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
</template>

<script setup>
import { ref } from 'vue';
const ok = ref(true);
</script>
```



- Với ok là true thì giao diện trong template được hiển thị, và ngược lại là false thì sẽ không hiển thị.

2. **v-else:** Sử dụng khi cần kiểm soát điều kiện ngược lại của **v-if**. Ví dụ:

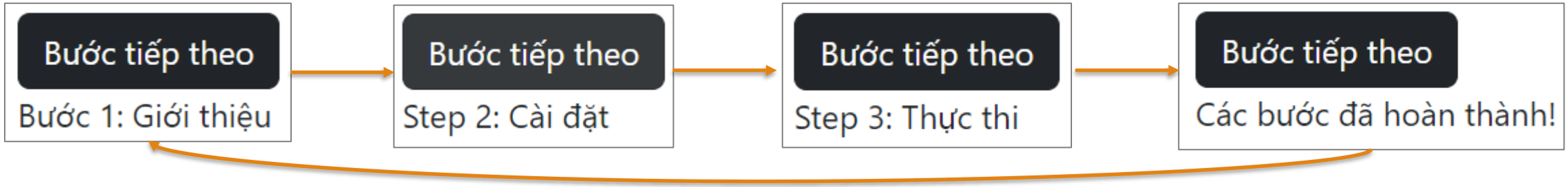
```
<template>
  <p>Bấm vào OK nhiều lần để chuyển đổi thông điệp</p>
  <button @click="message = !message">OK</button>
  <h1 v-if="message">Xin chúc mừng bạn!</h1>
  <h1 v-else>Rất tiếc, hẹn gặp lại!</h1>
</template>

<script setup>
  import { ref } from 'vue';
  const message = ref(true);
</script>
```

V-ELSE-IF

3. v-else-if: Sử dụng khi cần kiểm soát nhiều điều kiện hơn. Ví dụ:

```
<template>
    <button class="btn btn-dark" @click="nextStep">
    Bước tiếp theo
    </button>

    <p v-if="step === 1">Bước 1: Giới thiệu</p>
    <p v-else-if="step === 2">Step 2: Cài đặt</p>
    <p v-else-if="step === 3">Step 3: Thực thi</p>
    <p v-else>Các bước đã hoàn thành!</p>
</template>
```

```
<script setup>
import { ref } from 'vue';
const step = ref(1);
const nextStep = () => {
  if (step.value < 4) {
    step.value++;
  } else {
    step.value = 1;
  }
}
</script>
```



V-SHOW

4. **v-show**: tương tự như v-if, nhưng thay vì thỏa mãn điều kiện mới render ra thì v-show sẽ render ra hết, nhưng chỉ hiện thị phần thỏa mãn điều kiện, những phần còn lại sẽ được đặt thuộc tính display: none.

- Cú pháp tương tự v-if, ví dụ:

```
<h1 v-show="ok">Hello!</h1>
```

- Sự khác biệt là phần tử có **v-show** sẽ luôn được render, nó chỉ ẩn đi bằng css và luôn tồn tại trong tag chứa nó trên DOM
- **v-show** không hỗ trợ gom nhóm trên template và cũng không hoạt động với v-else.

V-SHOW

```
<template>
  <div>
    <button class="btn btn-dark"
        @click="toggleVisibility">Hiển thị/Ẩn
    </button>
    <p v-show="isVisible">
      Thông điệp này được chuyển đổi
       bởi v-show
    </p>
  </div>
</template>
```

```
<script setup>
import { ref } from 'vue';

const isVisible = ref(true);
const toggleVisibility = () => {
  isVisible.value = !isVisible.value;
}
</script>
```

Khi isVisible là false thì đây là component được render ra:

```
<div>
    <button class="btn btn-dark">Hiển thị/Ẩn</button>
    <p style="display: none;">Thông điệp này được chuyển đổi bởi v-show</p>
</div>
```

TRƯỜNG HỢP SỬ DỤNG V-SHOW VÀ V-IF

Chỉ thị	Trường hợp sử dụng
v-if	Dùng khi cần ẩn hiện các phần tử/component lớn, phức tạp, v-if sẽ thêm hoặc xóa phần tử khỏi DOM, giúp tiết kiệm tài nguyên hơn
v-show	Dùng khi cần hiển thị hoặc ẩn phần tử thường xuyên mà không cần thêm/xóa khỏi DOM. v-show chỉ thay đổi thuộc tính CSS display, giúp thao tác nhanh hơn. Thường áp dụng với các component nhỏ nhẹ mà có thể được bật tắt liên tục (ví dụ tooltip)

Ví dụ trường hợp áp dụng **v-show** với các component nhỏ nhẹ mà có thể được bật tắt liên tục:

Tạo giao diện <template>

```
<template>
  <div class="container mt-5 col-sm-5">
    <h2 class="mb-4">Top sách bán chạy</h2>
    <ul class="list-group">
      <li class="list-group-item" v-for="(product, index) in products">
        <h4>{{ product.name }} - {{ product.price }} VNĐ</h4>
        <button @click="toggleDetails(index)" class="btn btn-info">
          {{ showDetails[index] ? "Ẩn chi tiết" : "Xem chi tiết" }}
        </button>
        <p v-show="showDetails[index]" class="mt-2">{{ product.description }}</p>
      </li>
    </ul>
  </div>
</template>
```

Xử lý trong js

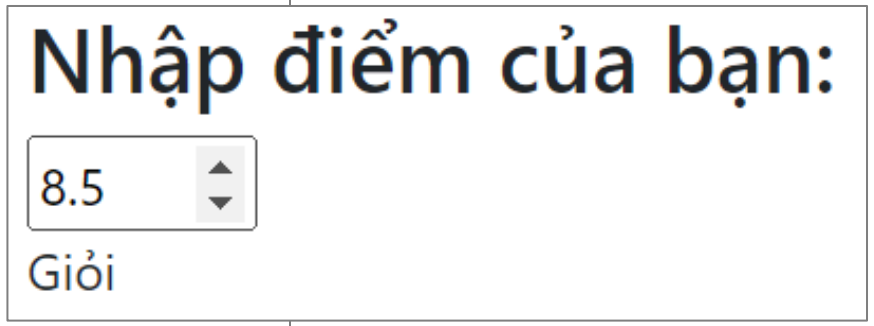
```
<script setup>
import { ref } from 'vue';
const products = ref([
 {
      name: "Vợ Nhặt",
      description: "Tác phẩm 'Vợ Nhặt' của tác giả Kim Lân (1921-2007)...",
      price: 55000,
  },
]);
// Trạng thái hiển thị chi tiết cho từng cuốn sách
const showDetails = ref(products.value.map(() => false));
// Phương thức để ẩn/hiển thị chi tiết sách
const toggleDetails = (index) => {
      showDetails.value[index] = !showDetails.value[index];
};
</script>
```

Nhập vào điểm của học sinh và in ra xếp loại tương ứng:

+ Nếu ĐTB < 5.0 thì xếp loại yếu
+ Nếu 5.0 ≤ ĐTB < 6.5 thì xếp loại trung bình
+ Nếu 6.5 ≤ ĐTB < 8.0 thì xếp loại khá
+ Nếu ĐTB 8.0 ≤ ĐTB < 9 thì xếp loại giỏi
+ Nếu ĐTB ≥ 9.0 thì xếp loại xuất sắc

HƯỚNG DẪN THỰC HIỆN DEMO

```
<template>
  <div>
    <h3>Nhập điểm của bạn:</h3>
    <input type="number" v-model="score" min="0" max="10" />

    <p v-if="score >= 9">Xuất sắc</p>
    <p v-else-if="score >= 8">Giỏi</p>
    <p v-else-if="score >= 6.5">Khá</p>
    <p v-else-if="score >= 5">Trung bình</p>
    <p v-else>Yếu</p>
  </div>
</template>

<script setup>
import { ref } from 'vue';

const score = ref(0);
</script>
```



FPT Education

FPT POLYTECHNIC

BÀI 6:

CONDITIONAL RENDERING VÀ LIST RENDERING

PHẦN II: LIST RENDERING

www.poly.edu.vn

LIST RENDERING

- **List rendering**: cho phép lặp qua một mảng dữ liệu và hiển thị mỗi phần tử của mảng đó dưới dạng một danh sách các phần tử HTML, được thực hiện bằng cách sử dụng chỉ thị v-for.
- Cú pháp:

```
v-for="item in list"
```

- Trong đó:
  - list là mảng dữ liệu cần duyệt.
  - item là biến được gán cho các phần tử có trong mảng.

V-FOR

Ví dụ: Hiển thị danh sách các học sinh theo tên.

```
<template>
    <ul>
        <li v-for="st in students">{{ st }}</li>
    </ul>
</template>

<script>
import { ref } from 'vue';
const students = ref([
    'Hoàng Văn Nam',
    'Vũ Văn Tài',
    'Lê Đức Trí',
    'Lê Thị Hà'
])
</script>
```

- Hoàng Văn Nam
- Vũ Văn Tài
- Lê Đức Trí
- Lê Thị Hà

V-FOR

- Trường hợp muốn lấy ra **index** của phần tử thì sử dụng cú pháp sau:

```
v-for="(item, index) in list"
```

- Khi đó tham số index sẽ chứa chỉ số index của phần tử trong mảng. Ví dụ:

```
<template>
    <ul>
        <li v-for="(st, index) in students">{{ st }} - Vị trí thứ {{ index }}</li>
    </ul>
</template>
```

- Hoàng Văn Nam - Vị trí thứ 0
- Vũ Văn Tài - Vị trí thứ 1
- Lê Đức Trí - Vị trí thứ 2
- Lê Thị Hà - Vị trí thứ 3

V-FOR VỚI OBJECT

Sử dụng **v-for** để lặp qua một mảng chứa các **đối tượng** và hiển thị thuộc tính của từng đối tượng.

- 1. Táo - Giá: $10
- 2. Chuối - Giá: $15
- 3. Dưa hấu - Giá: $18

```
<template>
    <ul>
      <li v-for="item in items" :key="item.id">
        {{ item.id }}. {{ item.name }} - Giá: {{ item.price }}
      </li>
    </ul>
</template>

<script setup>
import { ref } from 'vue';
const items = ref([
  { id: 1, name: 'Táo', price: '$10' },
  { id: 2, name: 'Chuối', price: '$15' },
  { id: 3, name: 'Dưa hấu', price: '$18' },
]);
</script>
```

V-FOR VỚI RANGE

- Vuejs cũng hỗ trợ trường hợp muốn hiển phần tử với số lượng xác định:

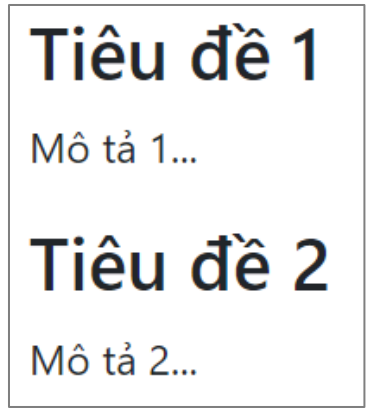
```
<template>
    <ul>
        <li v-for="i in 10">Số {{ i }}</li>
    </ul>
</template>
```

- Số 1
- Số 2
- Số 3
- Số 4
- Số 5
- Số 6
- Số 7
- Số 8
- Số 9
- Số 10

TEMPLATE VỚI V-FOR

- Trong trường hợp muốn hiển thị một khối các phần tử thì Vue.js hỗ trợ sử dụng **<template>** để gom nhóm các tag đó giống như v-if.

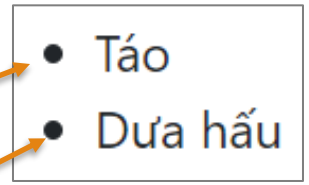
```
<template>
  <div>
    <template v-for="item in items" :key="item.id">
      <h3>{{ item.title }}</h3>
      <p>{{ item.description }}</p>
    </template>
  </div>
</template>

<script setup>
import { ref } from 'vue';
const items = ref([
  { id: 1, title: 'Tiêu đề 1', description: 'Mô tả 1...' },
  { id: 2, title: 'Tiêu đề 2', description: 'Mô tả 2...' },
]);
</script>
```



KẾT HỢP V-FOR VỚI V-IF

Khi sử dụng **v-for** cùng với **v-if**, thì **v-for** có độ ưu tiên cao hơn **v-if**. Điều này rất có lợi nếu như chúng ta muốn kiểm tra sự tồn tại của một nhánh nào đó trong vòng lặp.

```
<template>
   <ul>
      <template v-for="item in items" :key="item.id">
        <li v-if="item.visible">{{ item.name }}</li>
      </template>
    </ul>
</template>
<script setup>
import { ref } from 'vue';
const items = ref([
  { id: 1, name: 'Táo', visible: true },
  { id: 2, name: 'Chuối', visible: false },
  { id: 3, name: 'Dưa hấu', visible: true },
]);
</script>
```



Tạo form thêm mới sinh viên và hiển thị danh sách sinh viên như giao diện dưới đây:

Quản lý Sinh viên

Thêm sinh viên

Tên sinh viên

Tuổi

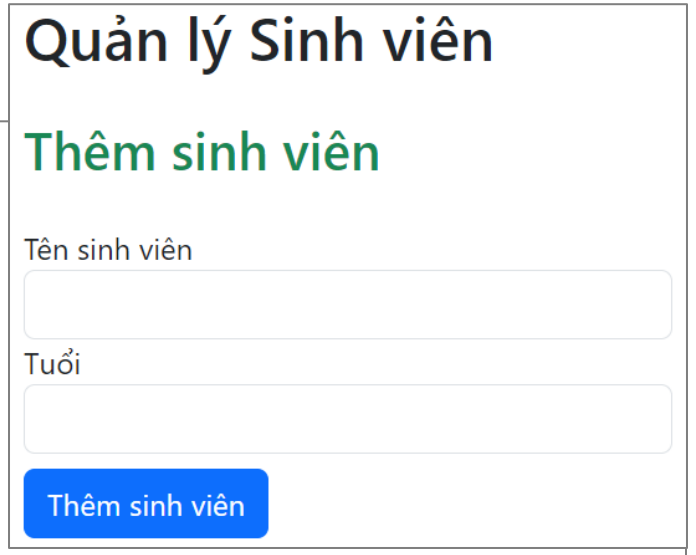
Thêm sinh viên

Danh sách sinh viên

- Trần Quang Anh - 18 tuổi
- Lê Thị Lan - 19 tuổi
- Phạm Văn Bảo - 18 tuổi

HƯỚNG DẪN THỰC HIỆN DEMO

B1: Tạo giao diện form Thêm sinh viên trong <template>




```
<template>
  <div class="container mt-5 row">
    <h2 class="mb-4">Quản lý Sinh viên</h2>

    <!-- Form thêm sinh viên mới -->
    <form class="col-sm-4" @submit.prevent="addStudent">
      <h3 class="mb-4 text-success">Thêm sinh viên</h3>
      <div class="form-group">
        <label for="name">Tên sinh viên</label>
        <input type="text" v-model="newStudent.name" class="form-control" required>
      </div>
      <div class="form-group">
        <label for="age">Tuổi</label>
        <input type="number" v-model.number="newStudent.age" class="form-control" required>
      </div>
      <button type="submit" class="btn btn-primary mt-2">Thêm sinh viên</button>
    </form>
  </div>
</template>
```

HƯỚNG DẪN THỰC HIỆN DEMO

B2: Thêm giao diện hiển thị danh sách sinh viên trong <template>

```
<!-- Danh sách sinh viên -->
    <div class="col-sm-4">
        <h3 class="mb-4 text-danger">Danh sách sinh viên</h3>
        <ul class="list-group">
        <li class="list-group-item" v-for="(student, index) in students" >
            {{ student.name }} - {{ student.age }} tuổi
        </li>
        </ul>
    </div>
```

HƯỚNG DẪN THỰC HIỆN DEMO

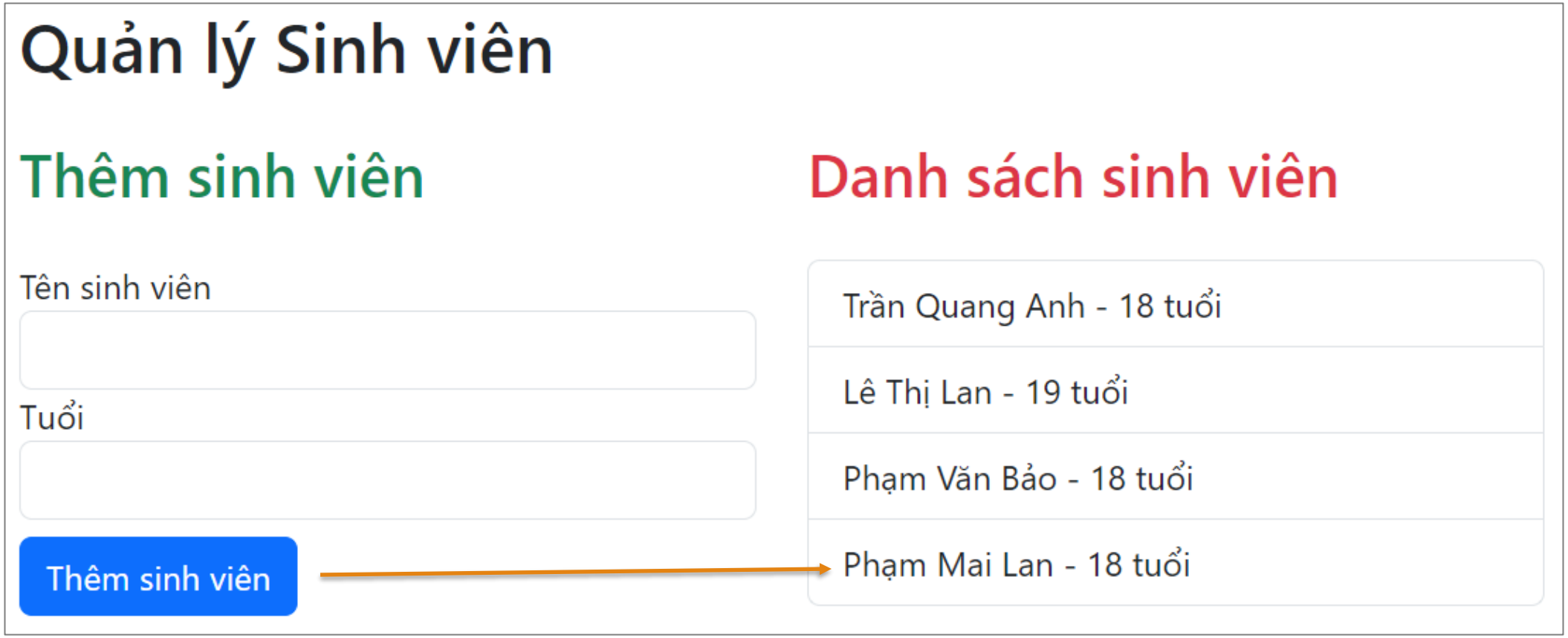
B3: Viết mã script

```
<script setup>
import { ref } from 'vue';
// Danh sách sinh viên
const students = ref([
    { name: 'Trần Quang Anh', age: 18 },
    { name: 'Lê Thị Lan', age: 19 },
    { name: 'Phạm Văn Bảo', age: 18 }
]);

// Dữ liệu sinh viên mới
const newStudent = ref({
    name: '',
    age: null
});
```

```
// Phương thức thêm sinh viên
const addStudent = () => {
if (newStudent.value.name &&
newStudent.value.age > 0){
 // Thêm sinh viên vào danh sách
    students.value.push({
        name: newStudent.value.name,
        age: newStudent.value.age
    });

 // Reset form sau khi thêm
    newStudent.value.name = '';
    newStudent.value.age = null;
 }
};
</script>
```

HƯỚNG DẪN THỰC HIỆN DEMO

Kết quả:



TỔNG KẾT NỘI DUNG BÀI HỌC

- ☑ Conditional rendering giúp hiển thị hoặc ẩn các phần tử trong DOM dựa trên các điều kiện cụ thể.
  - ☑ Sử dụng: v-if, v-else, v-else-if, v-show
- ☑ List rendering: cho phép lặp qua một mảng dữ liệu và hiển thị mỗi phần tử của mảng đó dưới dạng một danh sách các phần tử HTML.
  - ☑ Sử dụng: v-for


Thank
You