



FPT POLYTECHNIC



THỰC HỌC – THỰC NGHIỆP



Conceive Design Implement Operate

FRONT-END FRAMEWORK

CONDITIONAL RENDERING

VÀ LIST RENDERING

- ⊙ Nắm vững kiến thức Conditional rendering
- ⊙ Nắm vững kiến thức List rendering
- ⊙ Phân biệt trường hợp sử dụng v-if và v-show
- ⊙ Tìm hiểu v-for và các thành phần mở rộng



Phần I: Conditional Rendering

- ❖ v-if, v-else, v-else-if, v-show
- ❖ Sử dụng v-if và v-show
- ❖ So sánh v-if và v-show
- ❖ Thực hành

Phần II: List Rendering

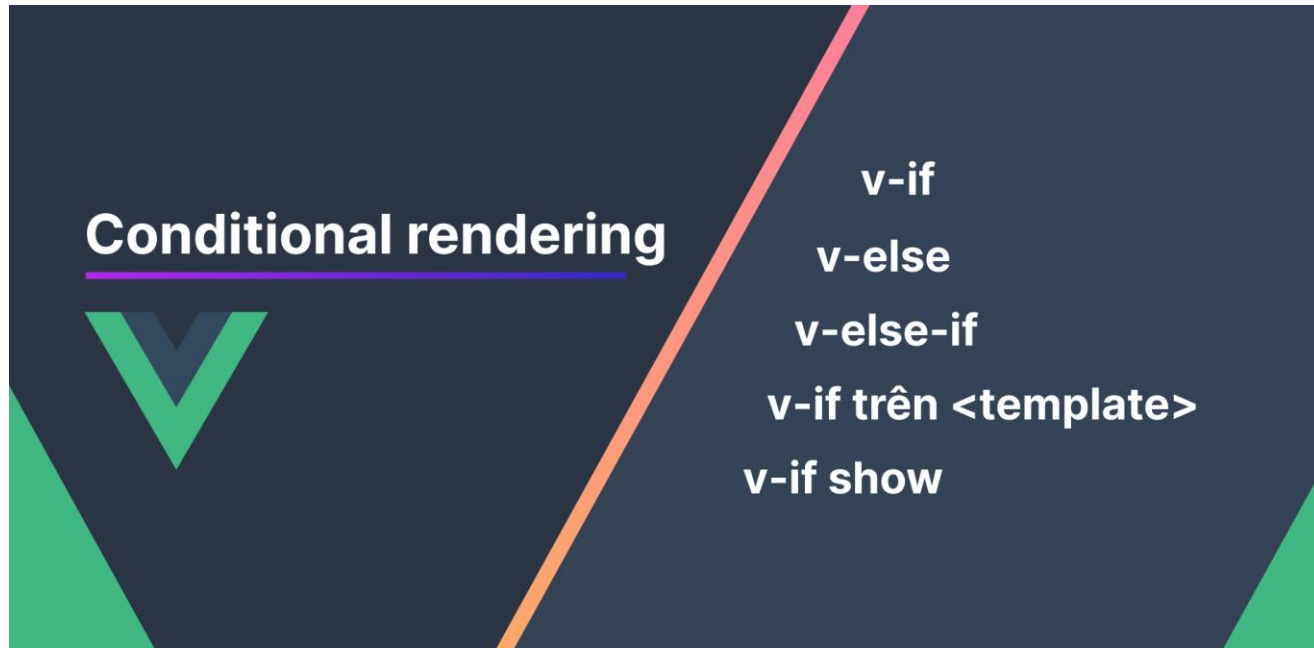
- ❖ v-for
- ❖ v-for với object
- ❖ v-for destructuring
- ❖ v-for với template
- ❖ Thực hành





PHẦN 1

CONDITIONAL RENDERING

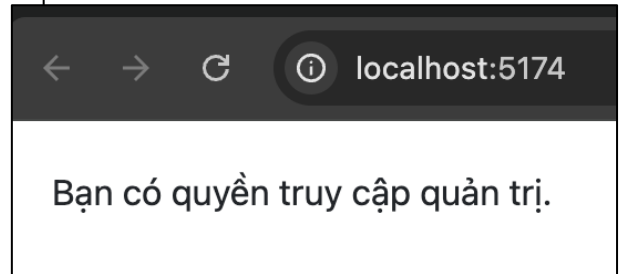


- ❑ Conditional rendering (Hiển thị có điều kiện): là một kỹ thuật trong phát triển web giúp hiển thị hoặc ẩn một phần của giao diện người dùng (UI) dựa trên một điều kiện cụ thể. Trong Vue.js, bạn có thể sử dụng các directive như **v-if**, **v-else-if**, **v-else**, và **v-show** để thực hiện việc này.

- **v-if** là một directive trong Vue.js được sử dụng để điều kiện hiển thị hoặc ẩn một phần tử trong DOM.
- Ví dụ

```
<template>
<div>
<span v-if="isAdmin">Bạn có quyền truy cập quản trị.</span>
</div>
</template>
<script setup>
import { ref } from "vue";
// Biến để xác định quyền truy cập
const isAdmin = ref(false); // Thay đổi thành true để kiểm tra quyền quản trị
</script>
```

➤ [Xem ví dụ](#)



Trong ví dụ này:

- Nếu isAdmin là true, thông báo "Bạn có quyền truy cập quản trị." sẽ được hiển thị.
- Nếu isAdmin là false, không có nội dung nào được hiển thị vì không có phần tử v-else.

- Bạn có thể sử dụng directive **v-else** để chỉ định một “khối lệnh else” cho **v-if**.
- Cú pháp:

```
<v-if="condition">Nội dung nếu điều kiện đúng</v-if>  
<v-else>Nội dung nếu điều kiện sai</v-else>
```

```
<template>  
  <div>  
    <h3>Kiểm tra tình trạng học viên:</h3>  
    <p v-if="score >= 5">Học viên đã qua môn</p>  
    <p v-else>Học viên chưa qua môn</p>  
  </div>  
</template>  
<script setup>  
  import { ref } from 'vue';  
  // Khai báo biến score và đặt giá trị ban đầu  
  const score = ref(4.5);  
</script>
```

Kiểm tra tình trạng học viên:
Học viên chưa qua môn

Trong đó:

- `v-if="score >= 5"`: Hiển thị đoạn văn “Học viên đã qua môn” nếu score lớn hơn hoặc bằng 5.
- `v-else`: Hiển thị đoạn văn “Học viên chưa qua môn” nếu score nhỏ hơn 5.

Yêu cầu:

1. Hiển thị form đăng nhập khi người dùng chưa đăng nhập (sử dụng `v-if="!isLoggedIn"`).
2. Hiển thị thông báo thành công và tên người dùng khi người dùng đã đăng nhập (sử dụng `v-if="isLoggedIn"`).
3. Quản lý trạng thái đăng nhập thông qua biến `isLoggedIn`:
4. Khi `isLoggedIn` là `false`, form đăng nhập được hiển thị.
5. Khi `isLoggedIn` là `true`, form đăng nhập ẩn đi và thông báo chào mừng xuất hiện.

Hướng dẫn thực hiện

```
<template>
  <div class="container mt-5">
    <!-- Form đăng nhập -->
    <div v-if="!isLoggedIn" class="login-form">
      <h2>Đăng nhập</h2>
      <input type="text" class="form-control mb-2" placeholder="Tên đăng nhập"/>
      <input type="password" class="form-control mb-2" placeholder="Mật khẩu"/>
      <button class="btn btn-primary">Đăng nhập</button>
    </div>
    <!-- Thông báo đăng nhập thành công và hiển thị tên người dùng -->
    <div v-if="isLoggedIn" class="alert alert-success mt-4" role="alert">
      Chào mừng, {{username}}! Bạn đã đăng nhập thành công.
    </div>
  </div>
</template>
<script setup>
import { ref } from "vue";
// Khai báo biến để theo dõi trạng thái đăng nhập và thông tin người dùng
// Nếu const isLoggedIn = ref(true); thì sẽ hiển thị thông báo và ẩn form
const isLoggedIn = ref(false);
const username = ref("Nguyễn Văn A");
</script>
```

- **v-else-if** là một directive dùng để kiểm tra nhiều điều kiện khác nhau sau v-if và trước v-else.
- Cú pháp:

```
<p v-if="condition1">Nội dung nếu điều kiện 1 đúng</p>
<p v-else-if="condition2">Nội dung nếu điều kiện 2 đúng</p>
<p v-else-if="condition3">Nội dung nếu điều kiện 3 đúng</p>
<p v-else>Nội dung nếu tất cả các điều kiện trên đều không đúng</p>
```

- Ví dụ

```
<template>
  <div>
    <h1>Thông báo điểm số:</h1>
    <p v-if="score">=90">Xuất sắc</p>
    <p v-else-if="score">=80">Giỏi</p>
    <p v-else-if="score">=70">Khá</p>
    <p v-else-if="score">=60">Trung bình</p>
    <p v-else>Yếu</p>
  </div>
</template>
<script setup>
import { ref } from "vue";
// Biến lưu điểm số của học sinh
const score = ref(75); // Thay đổi giá trị để kiểm tra các điều kiện
</script>
```

Thông báo điểm số:

Khá

- **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 hiển thị phần thỏa mãn điều kiện, những phần còn lại sẽ được đặt thuộc tính **display: none**.
- **Cú pháp**:

```
<h1 v-show="ok">Xin chào</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.

Ví dụ với yêu cầu:

- Tạo hai thông báo và một nút để chuyển đổi trạng thái hiển thị giữa hai thông báo.
- Một thông báo sẽ được hiển thị khi biến `isVisible` là `true`, và ngược lại khi nó là `false`, thông báo thứ hai sẽ xuất hiện.
- Nút bấm sẽ hiển thị “Ẩn Thông báo” khi **`isVisible`** là `true` và “Hiện Thông báo” khi **`isVisible`** là `false`.

Bước 1: Xây dựng template

```
<template>
<div class="container mt-5">
<!-- Nút để thay đổi trạng thái hiển thị -->
<button @click="toggleVisibility" class="btn btn-primary mb-3">
{{isVisible ? "Ẩn" : "Hiện"}} Thông báo
</button>
<!-- Thông báo chỉ hiển thị nếu isVisible là true -->
<div v-show="isVisible" class="alert alert-success" role="alert">
  Đây là thông báo thành công!
</div>
<!-- Thông báo ẩn khi isVisible là false -->
<div v-show="!isVisible" class="alert alert-warning" role="alert">
  Thông báo này sẽ hiển thị khi thông báo chính bị ẩn.
</div>
</div>
</template>
```

Bước 2: Viết script

```
<script setup>
import { ref } from "vue";
// Khai báo biến để theo dõi trạng thái hiển thị
const isVisible = ref(true);
// Hàm để chuyển đổi trạng thái hiển thị
const toggleVisibility = () => {
  isVisible.value = !isVisible.value;
};
</script>
```

SO SÁNH V-SHOW VÀ V-IF

v-if	v-show
<ul style="list-style-type: none"> • Hiển thị: Tạo và hủy phần tử trong DOM dựa trên điều kiện. • Hiệu suất: Tốt cho các điều kiện ít thay đổi vì tốn kém hơn khi tạo và phá hủy phần tử. • Sử dụng: Khi điều kiện hiển thị ít thay đổi hoặc khi cần giảm bớt tài nguyên. 	<ul style="list-style-type: none"> • Hiển thị: Thay đổi thuộc tính display của phần tử, không tạo hoặc hủy phần tử trong DOM. • Hiệu suất: Tốt cho các điều kiện thay đổi thường xuyên vì không tạo lại phần tử. • Sử dụng: Khi phần tử cần hiển thị hoặc ẩn thường xuyên mà không cần tái tạo



HIỂN THỊ FORM ĐĂNG NHẬP KHI TRẠNG THÁI LÀ LOGGEDOUT.

KHI NHẤN “ĐĂNG NHẬP”, CHUYỂN TRẠNG THÁI SANG LOGGINGIN (GIẢ LẬP QUÁ TRÌNH ĐĂNG NHẬP).

SAU 2 GIÂY, NẾU NGƯỜI DÙNG NHẬP ĐÚNG TÊN ĐĂNG NHẬP VÀ MẬT KHẨU, TRẠNG THÁI CHUYỂN SANG LOGGEDIN VÀ HIỂN THỊ THÔNG BÁO CHÀO MỪNG.

NẾU THÔNG TIN KHÔNG HỢP LỆ, HIỂN THỊ LẠI FORM VÀ THÔNG BÁO LỖI.

Bước 1: Xây dựng giao diện

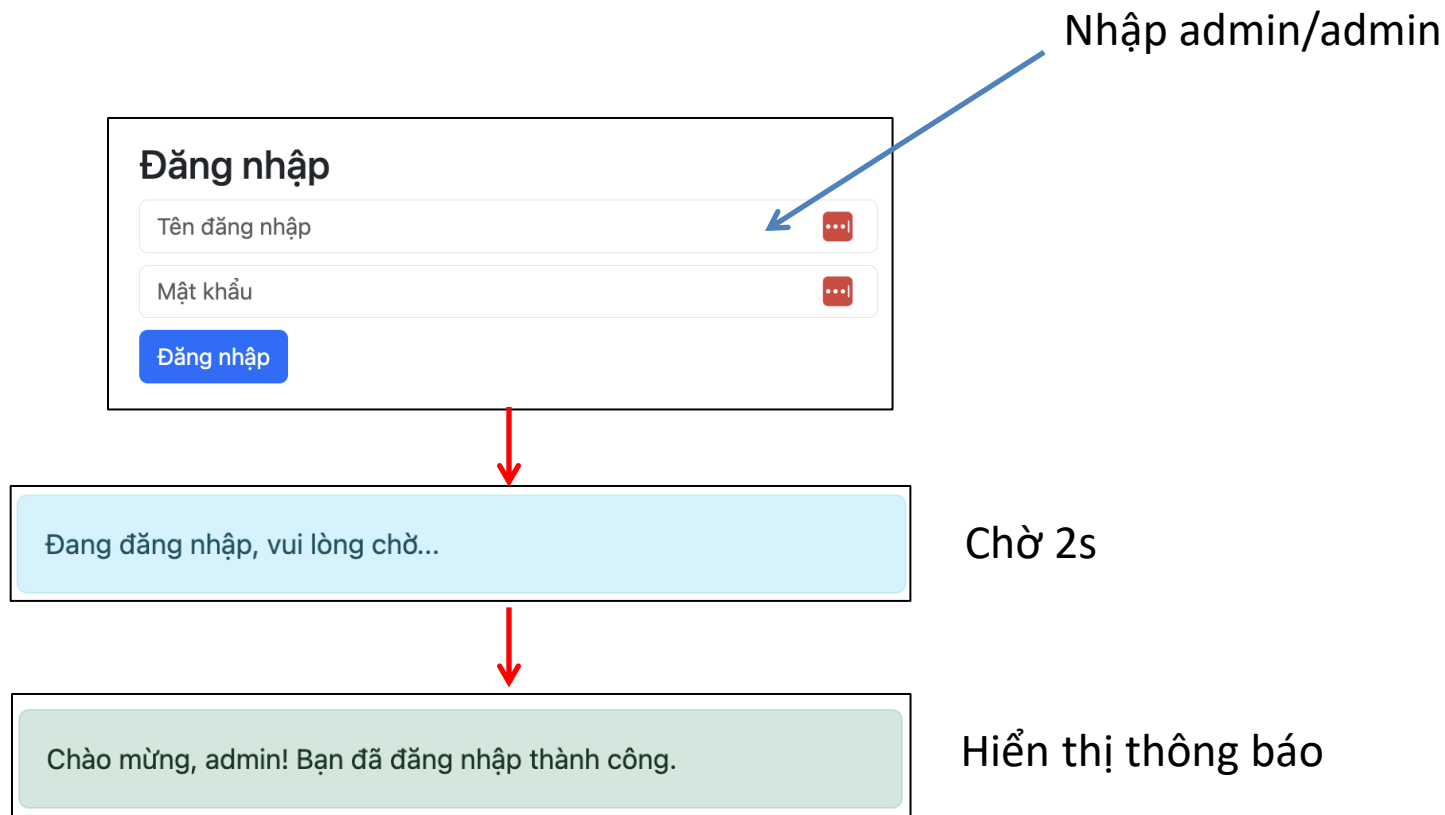
```
<template>
<div class="container mt-5">
  <!-- Trạng thái chưa đăng nhập -->
  <div v-if="status === 'loggedOut'" class="login-form">
    <h2>Đăng nhập</h2>
    <input type="text"
      v-model="username"
      class="form-control mb-2"
      placeholder="Tên đăng nhập"
    />
    <input type="password"
      v-model="password"
      class="form-control mb-2"
      placeholder="Mật khẩu"
    />
    <button @click="login"
      class="btn btn-primary">Đăng nhập</button>
  </div>
  <!-- Trạng thái đang đăng nhập -->
  <div v-else-if="status === 'loggingIn'" class="alert alert-info mt-4">
    Đang đăng nhập, vui lòng chờ...
  </div>
  <!-- Trạng thái đã đăng nhập thành công -->
  <div v-else class="alert alert-success mt-4">
    Chào mừng, {{username}}! Bạn đã đăng nhập thành công.
  </div>
</div>
</template>
```

 [Xem ví dụ](#)

Bước 2: Viết script

```
<script setup>
import { ref } from "vue";
// Khai báo biến phản ứng để theo dõi trạng thái và thông tin người dùng
// Các trạng thái: 'loggedOut', 'loggingIn', 'loggedIn'
const status = ref("loggedOut");
const username = ref("");
const password = ref("");
// Hàm giả lập quá trình đăng nhập
const login = async () => {
    status.value = "loggingIn";
    // Chuyển trạng thái sang đang đăng nhập
    // Giả lập quá trình đăng nhập
    // Giả lập delay 2 giây
    await new Promise((resolve) => setTimeout(resolve, 2000));
    if (username.value && password.value) {
        status.value = "loggedIn"; // Đăng nhập thành công
    } else {
        status.value = "loggedOut"; // Đăng nhập thất bại
        alert("Vui lòng nhập tên đăng nhập và mật khẩu");
    }
};
</script>
```

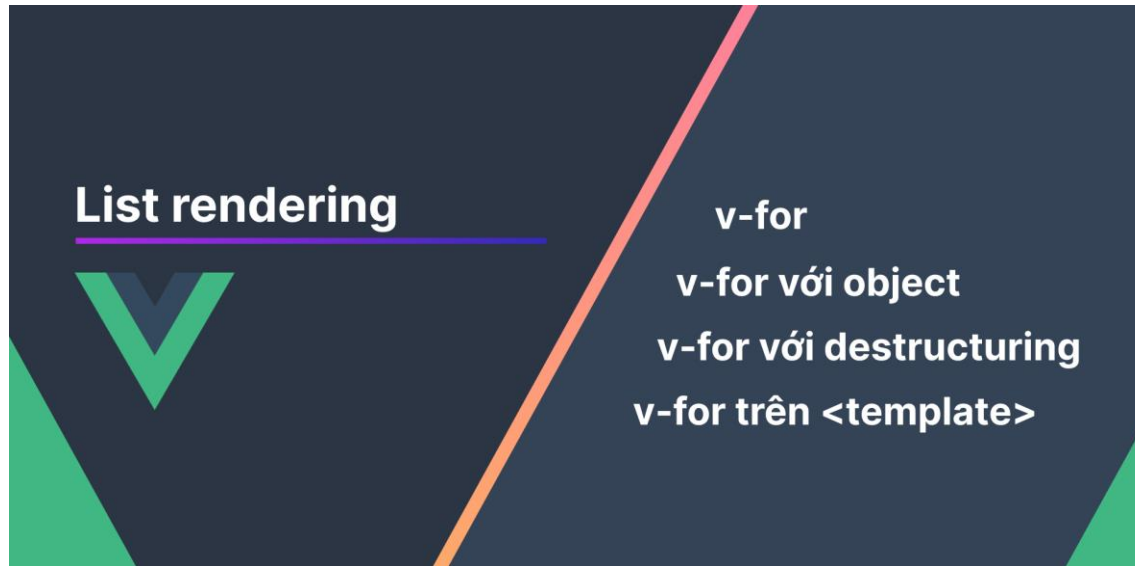
Kết quả



➤ [Xem ví dụ](#)



PHẦN 2: LIST RENDERING



- ❑ **List rendering** là cơ chế lặp qua các phần tử trong một mảng hoặc đối tượng và hiển thị chúng trên giao diện. Vue cung cấp directive **v-for** để thực hiện việc lặp này.
- ❑ 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.

❑ Ví dụ : hiển thị danh sách các món ăn

```
<template>
  <div class="container mt-4">
    <h3 class="mb-3">Danh sách các món ăn:</h3>
    <ul class="list-group">
      <li v-for="food in foods"
        :key="food"
        class="list-group-item">
        {{food}}
      </li>
    </ul>
  </div>
</template>
<script setup>
  import { reactive } from 'vue';
  const foods = reactive(["Pizza", "Burger", "Sushi", "Pasta"]);
</script>
```

Danh sách các món ăn:

Pizza

Burger

Sushi

Pasta

- ❑ 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" :key="index">
      {{st}} - Vị trí thứ {{index + 1}}
    </li>
  </ul>
</template>
<script setup>
import { reactive } from "vue";
const students = reactive(["Đạt", "Kiên", "Sơn", "Quỳnh"]);
</script>
```

- John - Vị trí thứ 1
- Anna - Vị trí thứ 2
- Peter - Vị trí thứ 3
- Maria - Vị trí thứ 4

➤ [Xem ví dụ](#)

- ❑ **v-for** có thể sử dụng cú pháp phân rã (destructuring) trên item

```
<template>
  <div class="container mt-4">
    <h3 class="mb-3">Danh sách món ăn và giá:</h3>
    <ul class="list-group">
      <li v-for="(price, food) in foodPrices" :key="food" class="list-group-item">
        {{food}}: {{price}} VND
      </li>
    </ul>
  </div>
</template>
<script setup>
import { reactive } from "vue";
// Object chứa thông tin món ăn và giá
const foodPrices = reactive({ pizza:200000, burger:80000, sushi:300000, pasta:150000 });
</script>
```

Trong đó:

- **v-for="(price, food) in foodPrices"**: Đây là cú pháp destructuring. price đại diện cho giá trị, và food đại diện cho khóa (tên món ăn).
- **foodPrices**: Đây là một object trong Vue, với mỗi khóa (tên món ăn) đi kèm một giá trị (giá tiền).

➤ [Xem ví dụ](#)

Khi sử dụng v-for với object trong Vue.js, bạn có thể lặp qua các thuộc tính của đối tượng. v-for trả về cặp **[key, value]**, giúp bạn truy cập cả khóa và giá trị của object.

```
<template>
  <div class="container mt-5">
    <ul class="list-group">
      <!-- Lặp qua các thuộc tính của đối tượng itemDetails -->
      <li v-for="(value, key) in itemDetails" :key="key" class="list-group-item">
        <strong>{{key}}:</strong> {{value}}
      </li>
    </ul>
  </div>
</template>
<script setup>
  import { reactive } from "vue"; // Dữ liệu đối tượng với nhiều thuộc tính
  const itemDetails = reactive({ name: "Sản phẩm A", code: "A001", price: 29.99, inStock: true,
});
</script>
```

name: Sản phẩm A

code: A001

price: 29.99

inStock: true

- ❑ 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
- ❑ Ví dụ:

```
<template>
<div>
  <template v-for="product in products" :key="product.id">
    <h3>{{product.name}}</h3>
    <p>Giá: {{product.price}} VND</p>
    <p>Kho: {{product.stock>0 ? "Còn hàng" : "Hết hàng"}}</p>
  </template>
</div>
</template>
<script setup>
  import { reactive } from "vue";
  const products = reactive([
    { id:1, name:"Điện thoại", price:5000000, stock:10 },
    { id:2, name:"Laptop", price:15000000, stock:0 },
  ]);
</script>
```

Điện thoại

Giá: 5000000 VND

Kho: Còn hàng

Laptop

Giá: 15000000 VND

Kho: Hết hàng



HIỂN THỊ DANH SÁCH SẢN PHẨM: MỖI SẢN PHẨM BAO GỒM HÌNH ẢNH, TÊN, MÃ SẢN PHẨM VÀ GIÁ.

Bước 1: Xây dựng giao diện template

```
<template>
<div class="container mt-5">
  <h1>Quản lý sản phẩm</h1>
  <!-- Bootstrap table để hiển thị danh sách items -->
  <table class="table table-striped">
    <thead>
      <tr>
        <th>Hình ảnh</th>
        <th>Tên</th>
        <th>Mã</th>
        <th>Giá</th>
        <th>Tình trạng</th>
        <th>Thao tác</th>
      </tr>
    </thead>
    <tbody>
      <!-- Lặp qua mảng items và render một hàng cho mỗi item -->
      <tr v-for="item in state.items" :key="item.id">
        <td>
          
        </td>
        <td>{{item.name}}</td>
        <td>{{item.code}}</td>
        <td>{{item.price | currency}}</td>
        <td>
          <!-- Hiển thị tình trạng sản phẩm -->
          <span :class="{ badge:true, 'bg-success':item.available, 'bg-secondary': !item.available}">
            {{item.available ? 'Có sẵn' : 'Hết hàng'}}
          </span>
        </td>
        <td width="200">
          <!-- Thêm nút xóa -->
          <button class="btn btn-danger" @click="deleteItem(item.id)">
            Xóa
          </button>
        </td>
      </tr>
    </tbody>
  </table>
</div>
</template>
```


Bước 2: viết script chứa dữ liệu và logic xử lý

```
<script setup>
import { reactive } from 'vue';
// Dữ liệu danh sách
const state = reactive({
  items: [
    {
      id:1,
      image:'https://picsum.photos/id/1/50/50',
      name:'Item 1',
      code:'001',
      price:10.0,
      available:true,
    },
    {
      id:2,
      image:'https://picsum.photos/id/2/50/50',
      name:'Item 2',
      code:'002',
      price:20.0,
      available:false,
    },
    {
      id:3,
      image:'https://picsum.photos/id/3/50/50',
      name:'Item 3',
      code:'003',
      price:30.0,
      available:true,
    },
  ],
});
```

```
// Hàm để xóa sản phẩm khỏi danh sách
const deleteItem = (id) => {
  state.items = state.items.filter((item) => item.id !== id);
};
</script>
```

Kết quả

Quản lý sản phẩm

Hình ảnh	Tên	Mã	Giá	Tình trạng	Thao tác
	Item 1	001	10	Có sẵn	Xóa
	Item 2	002	20	Hết hàng	Xóa
	Item 3	003	30	Có sẵn	Xóa

Click Xóa

Hình ảnh	Tên	Mã	Giá	Tình trạng	Thao tác
	Item 1	001	10	Có sẵn	Xóa
	Item 2	002	20	Hết hàng	Xóa

- ☑️ Nắm vững kiến thức Conditional rendering
- ☑️ Nắm vững kiến thức List rendering
- ☑️ Phân biệt trường hợp sử dụng v-if và v-show
- ☑️ Tìm hiểu v-for và các thành phần mở rộng



thank
you!