FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

THỰC HỌC – THỰC NGHIỆP

FRONT-END FRAMEWORK

EVENT HANDLING VÀ FORM BINDING

http://www.poly.edu.vn

Mục tiêu

- Kết thúc bài học này bạn có khả năng
  - Hiểu về Listen to Events (Lắng nghe sự kiện)
  - Tìm hiểu Trình xử lý phương thức
  - Các sự kiện về chuột, bàn phím…
  - Nắm được cách thức liên kết dữ liệu trong Form binding
  - Sử dụng v-model kết hợp các thuộc tính, giá trị

NỘI DUNG

- Phần I: Event Handling
  - Listen to Events (Lắng nghe sự kiện)
  - Method Handlers (Trình xử lý phương thức)
  - Event modifiers, Key modifiers, Mouse Button Modifiers
- Phần II: Form Binding
  - Binding đơn giản
  - Ràng buộc giá trị

PHẦN 1
EVENT HANDLING

Event Handling (Xử lý sự kiện) trong Vue.js

LẮNG NGHE SỰ KIỆN

Listen to Events (Lắng nghe sự kiện)

- Trong vue.js để lắng nghe sự kiện, bạn sử dụng thuộc tính **v-on** hoặc cú pháp rút gọn @ trong các phần tử HTML:

```
<!-- Cú pháp v-on -->
<button v-on:click="handler">Click me</button>
<!-- Cú pháp rút gọn -->
<button @click="handler()">Click me</button>
```

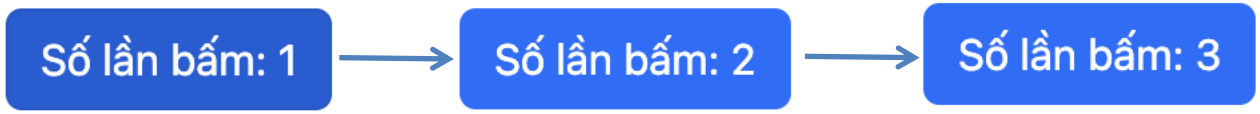
- Giá trị của **handler** có các loại sau:
  - **inline handler**: một cách để xử lý sự kiện trực tiếp trong phần tử HTML
  - **method handler**: là cách xử lý sự kiện bằng cách gọi một phương thức được định nghĩa trong component của Vue.js.

LẮNG NGHE SỰ KIỆN

inline handler: Cú pháp này cho phép bạn định nghĩa các hàm xử lý sự kiện trực tiếp trong html

Ví dụ 1: Xử lý sự kiện click

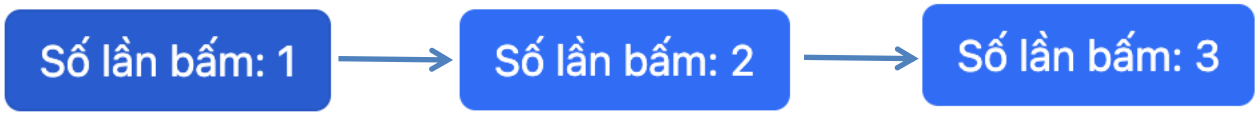
```
<template>
<button class="btn btn-primary" @click="counter++">Số lần bấm: {{counter}}</button>
</template>
<scriptsetup>
import { ref } from"vue";
const counter = ref(0);
</script>
```



METHOD HANDLERS

Method handlers (Trình xử lý phương thức)

- Trong các bài toán thực tế, mã xử lý sự kiện không chỉ dừng lại ở một dòng lệnh đơn giản. Khi logic phức tạp hơn, Vue.js cho phép tách phần xử lý sự kiện vào các hàm (methods), giúp mã dễ đọc và bảo trì, tương tự như trong JavaScript thuần.
- Ví dụ: Chuyển đoạn code xử lý sự kiện ở trên vào hàm



METHOD HANDLERS

Thực hiện ví dụ:

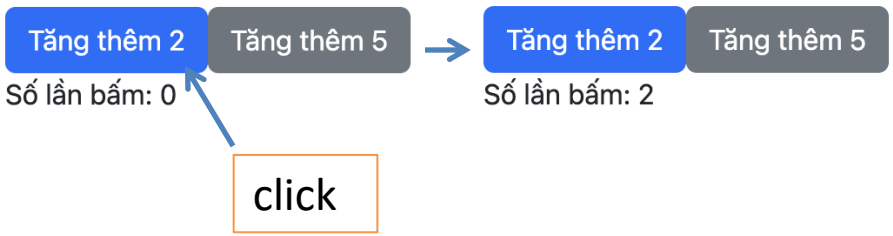
```
<template>
<button class="btn btn-primary" @click="increment">Số lần bấm: {{counter}}</button>
</template>
<script setup>
import { ref } from"vue";
const counter = ref(0);
const increment = () => {
counter.value++;
};
</script>
```

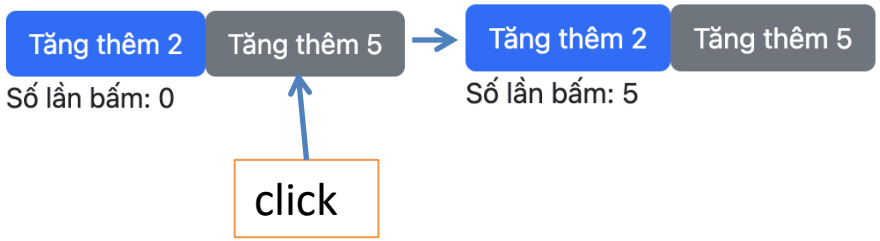
- @click="increment": Sự kiện click được gắn với hàm increment. Mỗi lần người dùng nhấn vào nút, hàm này sẽ được gọi.
- Hàm **increment** có nhiệm vụ tăng giá trị của **counter** mỗi lần nút được nhấn. Vì counter là một đối tượng **ref**, ta cần truy cập và thay đổi giá trị thông qua **.value**.

Truyền Tham Số trong Event Handlers

Bạn có thể truyền tham số vào các hàm (methods) để xử lý sự kiện, giúp bạn tùy chỉnh logic theo từng trường hợp cụ thể.

```
<template>
<button class="btn btn-primary" @click="incrementBy(2)">Tăng thêm 2</button>
<button class="btn btn-secondary" @click="incrementBy(5)">Tăng thêm 5</button>
<p>Số lần bấm: {{counter}}</p>
</template>
<script setup>
import { ref } from"vue";
const counter = ref(0);
// Hàm nhận tham số
const incrementBy = (amount) => {
counter.value += amount;
};
</script>
```





FPT Education
FPT POLYTECHNIC

SO SÁNH INLINE VÀ METHOD HANDLERS

Tiêu chí	Inline handlers	Method handlers
Định nghĩa	Viết trực tiếp trong template.	Tách riêng trong <script>.
Cú pháp	@click="counter++"	@click="increment"
Tính đọc hiểu	Khó đọc với logic phức tạp.	Dễ đọc và bảo trì hơn.
Tái sử dụng	Khó tái sử dụng.	Dễ tái sử dụng.
Phức tạp	Phù hợp với xử lý đơn giản.	Tốt hơn cho logic phức tạp.
Gỡ lỗi	Khó khăn hơn nếu có lỗi.	Dễ gỡ lỗi hơn.

Kết luận:

Inline Handlers: Nhanh gọn cho xử lý đơn giản.

Method Handlers: Tốt hơn cho logic phức tạp và dễ bảo trì.

xây dựng một ứng dụng đơn giản cho việc đếm số. Ứng dụng cho phép người dùng tăng, giảm và đặt lại giá trị số đếm.

THỰC HIỆN DEMO

Hướng dẫn thực hiện:

```
<template>
        <div>
        <h1>Đếm Số: {{count}}</h1>
        <button @click="increment">Tăng</button>
        <button @click="decrement">Giảm</button>
        <button @click="reset">Đặt lại</button>
        </div>
</template>
<script setup>
import { ref } from'vue';
const count = ref(0); // Khai báo biến count
const increment = () => {
        count.value++; // Tăng giá trị count
};
const decrement = () => {
        count.value--; // Giảm giá trị count
};
const reset = () => {
        count.value = 0; // Đặt lại giá trị count
};
</script>
```

EVENT MODIFIER

- Khi làm việc với JavaScript thuần, xử lý sự kiện thường yêu cầu nhiều mã để quản lý các tình huống khác nhau. Ví dụ, bạn có thể cần ngăn chặn hành vi mặc định của một form khi người dùng gửi lên khi nhấn một nút.
- Điều này có thể được thực hiện như sau:

```
functionformHandler(event) {
        event.preventDefault(); // Ngăn gửi form
        // Logic xử lý form ở đây
        console.log("Form được gửi");
}
document.querySelector("form").addEventListener("submit", formHandler);
```

- Vue 3 cung cấp **event modifiers** để đơn giản hóa các tình huống phổ biến này ngay trong template mà không cần viết nhiều mã JavaScript.

```
<form @submit.prevent="handleSubmit">Gửi</form>
```

EVENT MODIFIER

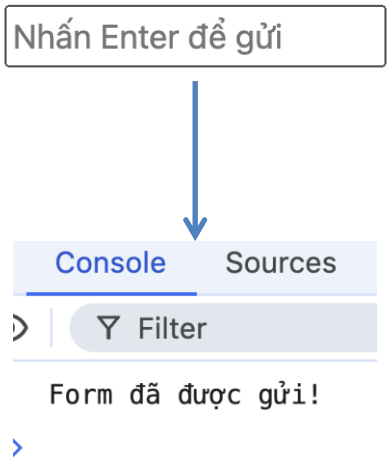
Vue cung cấp một số event modifiers (bộ điều chỉnh sự kiện) để giúp xử lý các tình huống phổ biến một cách dễ dàng và ngắn gọn.

Event Modifier	Mô tả
.stop	Ngăn chặn sự kiện không tiếp tục truyền lên các phần tử cha trong DOM
.prevent	Gọi event.preventDefault(), sử dụng để ngăn chặn hành vi mặc định của sự kiện
.capture	Lắng nghe sự kiện ở giai đoạn capturing
.self	Chỉ kích hoạt sự kiện với chính phần tử đang gán sự kiện đó
.once	Chỉ kích hoạt sự kiện một lần

KEY MODIFIERS

- Vue cho phép bạn sử dụng các **modifier** để lắng nghe các phím cụ thể, ví dụ như Enter, Tab, Esc, và nhiều phím khác.
- Điều này giúp việc xử lý sự kiện bàn phím trở nên đơn giản và dễ bảo trì hơn, đặc biệt trong các ứng dụng cần nhiều tương tác với bàn phím.
- Cú pháp: `<element @event.keyModifier="method"/>`
- Ví dụ

```
<template>
        <input type="text"
                placeholder="Nhấn Enter để gửi"
                @keyup.enter="submitForm"/>
</template>
<script setup>
const submitForm = () => {
        console.log("Form đã được gửi!");
};
</script>
```



KEY MODIFIERS

Vue đã cung cấp alias (tên phím mà con người hiểu được) cho một số phím thông dụng:

- .enter
- .tab
- .delete
- .esc
- .space
- .up, .down, .left, .right

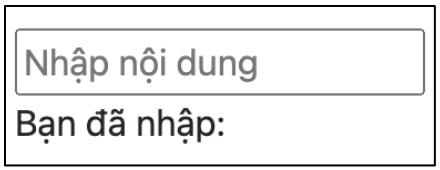
PHẦN 2: FORM BINDING

TỔNG QUAN VỀ FORM BINDING

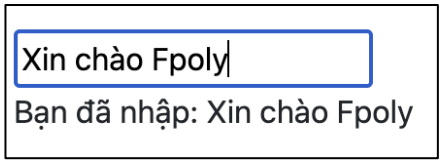
- **Form binding** trong Vue.js liên kết dữ liệu giữa các phần tử HTML trong form và component.
- Directive **v-model** giúp việc này trở nên dễ dàng và hiệu quả bằng cách tự động đồng bộ hóa dữ liệu.

FORM BINDING CƠ BẢN

❑ **Text input**: Binding đơn giản với input type text sử dụng v-model

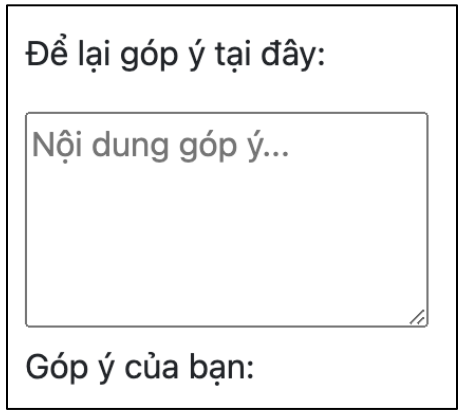


Thay đổi nội dung input




```
<template>
        <input v-model="message" placeholder="Nhập nội dung"/>
        <p>Bạn đã nhập: {{message}}</p>
</template>
<script setup>
import { ref } from"vue";
const message = ref("");
</script>
```

❑ **Multiline text**: Văn bản nhiều dòng

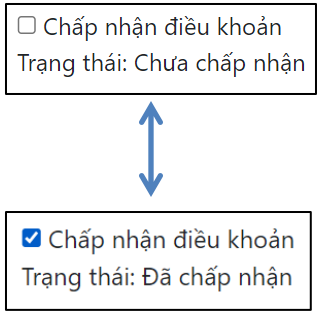



```
<script setup>
import { ref } from"vue";
const feedback = ref("");
</script>
<template>
<span>Để lại góp ý tại đây:</span>
<textarea v-model="feedback" placeholder="Nội dung góp ý..."rows="4"> </textarea>
<p>Góp ý của bạn: {{feedback}}</p>
</template>
```

FORM BINDING CƠ BẢN

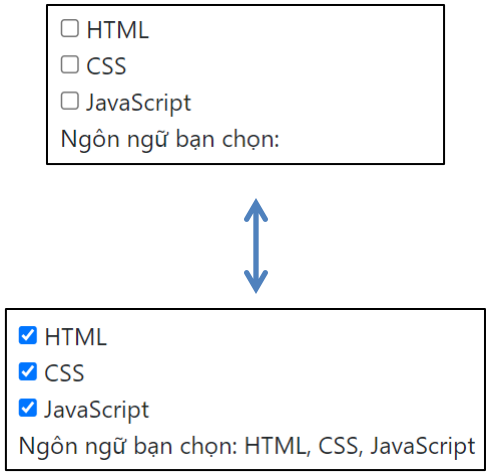
Checkbox

```
<template>
        <label>
                <input type="checkbox"v-model="isChecked"/>
                Chấp nhận điều khoản
        </label>
        <p>Trạng thái: {{isChecked ? "Đã chấp nhận" : "Chưa chấp nhận"}}</p>
</template>
<script setup>
        import { ref } from"vue";
        const isChecked = ref(false);
</script>
```



Multiple checkbox

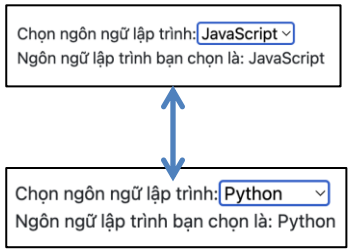
```
<template>
<input type="checkbox"value="HTML"v-model="coursesName"/>HTML <br/>
<input type="checkbox"value="CSS"v-model="coursesName"/>CSS <br/>
<input type="checkbox"value="JavaScript"v-model="coursesName"/>JavaScript <br/>
<p>Ngôn ngữ bạn chọn: {{coursesName.join(", ") }}</p>
</template>
<script setup>
import { ref } from"vue";
const coursesName = ref([]);
</script>
```



FORM BINDING CƠ BẢN

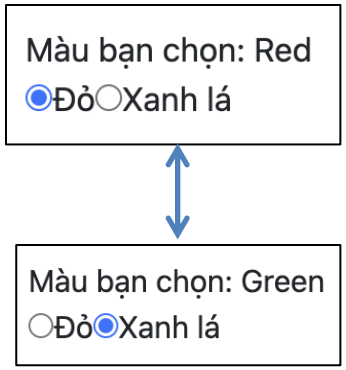
Select/Option:

```
<template>
        <label for="language">Chọn ngôn ngữ lập trình: </label>
        <select id="language"v-model="selectedLanguage">
                <option value="JavaScript">JavaScript</option>
                <option value="Java">Java</option>
                <option value="C++">C++</option>
        </select>
        <p>Ngôn ngữ lập trình bạn chọn là: {{selectedLanguage}}</p>
</template>
<script setup>
import { ref } from"vue";
const selectedLanguage=ref("JavaScript");
</script>
```



Radio

```
<script setup>
        import { ref } from"vue";
        const favoriteColor = ref("Red");
</script>
<template>
        <div>Màu bạn chọn: {{favoriteColor}}</div>
        <input type="radio"id="red" value="Red" v-model="favoriteColor"/>
        <label for="red">Đỏ</label>
        <input type="radio"id="green" value="Green" v-model="favoriteColor"/>
        <label for="green">Xanh lá</label>
</template>
```



VALUE BINDINGS

Value Bindings: (Ràng buộc giá trị)

Trong Vue 3, v-model tạo liên kết hai chiều giữa dữ liệu trong component và các input elements (như input, select, textarea). Khi giá trị input thay đổi, dữ liệu được cập nhật tự động và ngược lại, khi dữ liệu thay đổi, giá trị trên input cũng được cập nhật.

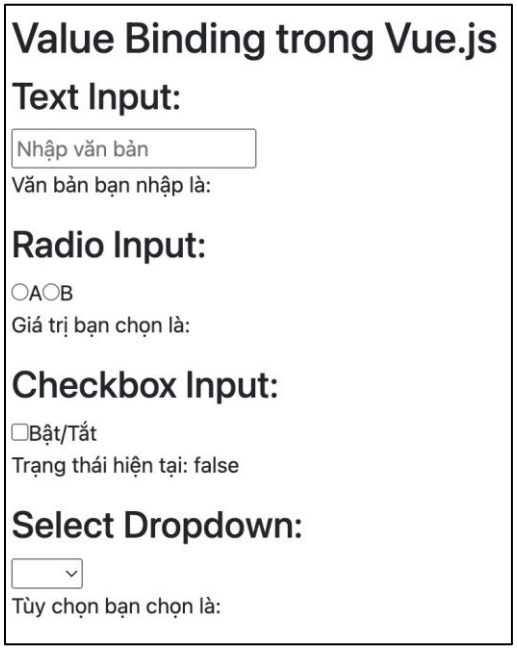
```
<template>
<!-- `picked` là một chuỗi "a" khi được chọn -->
<input type="radio"v-model="picked"value="a"/>
<!-- `toggle` là true hoặc false -->
<input type="checkbox"v-model="toggle"/>
<!-- `selected` là một chuỗi "abc" khi tùy chọn đầu tiên được chọn -->
<select v-model="selected">
<option value="abc">ABC</option>
</select>
</template>
```

VALUE BINDINGS

Thực hiện ví dụ

```
<template>
        <h2>Value Binding trong Vue.js</h2>
        <h3>Text Input:</h3>
        <input type="text" v-model="text" placeholder="Nhập văn bản"/>
        <p>Văn bản bạn nhập là: {{text}}</p>
        <h3>Radio Input:</h3>
        <input type="radio" id="a" value="a" v-model="picked"/>
        <label for="a">A</label>
        <input type="radio" id="b" value="b" v-model="picked"/>
        <label for="b">B</label>
        <p>Giá trị bạn chọn là: {{picked}}</p>
        <h3>Checkbox Input:</h3>
        <input type="checkbox" id="toggle" v-model="toggle"/>
        <label for="toggle">Bật/Tắt</label>
        <p>Trạng thái hiện tại: {{toggle}}</p>
        <h3>Select Dropdown:</h3>
        <select v-model="selected">
        <option value="abc">ABC</option>
        <option value="def">DEF</option>
        <option value="ghi">GHI</option>
        </select>
        <p>Tùy chọn bạn chọn là: {{selected}}</p>
</template>
```

```
<script setup>
import { ref } from"vue";
const text=ref("");
const picked=ref("");
const toggle=ref(false);
const selected=ref("");
</script>
```



FPT Education
FPT POLYTECHNIC

V-MODEL VỚI CÁC THUỘC TÍNH KHÁC

Sử dụng **.lazy**, **.number**, và **.trim modifiers**

- **.lazy**: Chỉ cập nhật giá trị sau khi sự kiện change được kích hoạt thay vì input.
- **.number**: Chuyển đổi giá trị input thành kiểu số.
- **.trim**: Tự động loại bỏ khoảng trắng ở đầu và cuối của chuỗi.

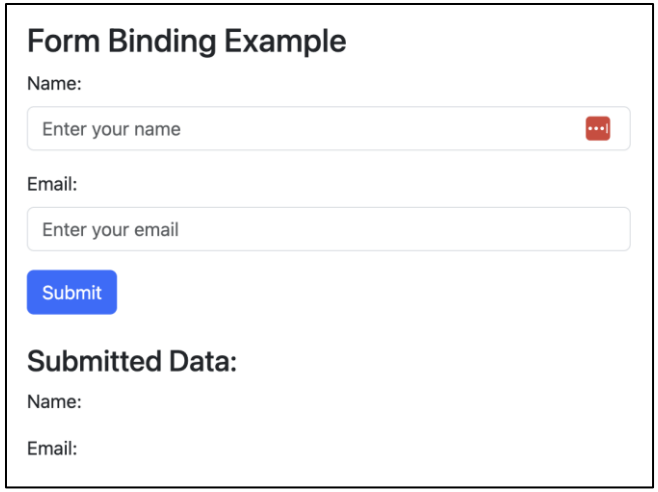
```
<template>
<!-- đồng bộ sau sự kiện "change" thay vì "input" -->
<input v-model.lazy="msg"/>
<!-- chuyển đổi giá trị thành kiểu số -->
<input v-model.number="age"/>
<!-- loại bỏ khoảng trắng đầu và cuối chuỗi -->
<input v-model.trim="msg"/>
</template>
```

Tạo form trong Vue 3 để thu thập tên và email người dùng, sử dụng v-model để liên kết dữ liệu và hiển thị kết quả sau khi gửi.

THỰC HIỆN DEMO

Bước 1: Xây dựng giao diện

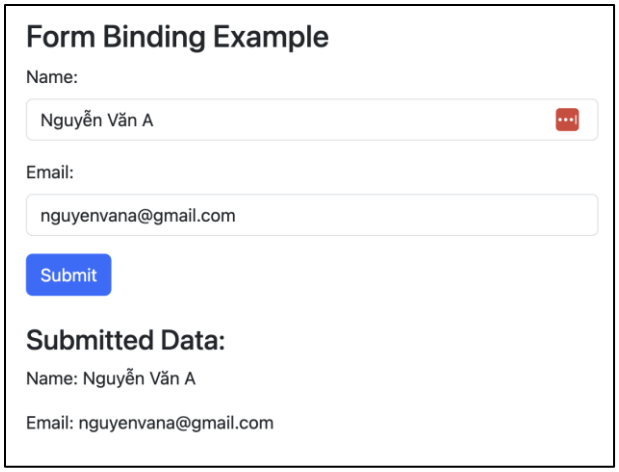
```
<template>
<div class="container mt-5">
<h2>Form Binding Example</h2>
<form @submit.prevent="handleSubmit">
        <div class="mb-3">
                <label for="name"class="form-label">Name:</label>
                <input v-model.trim="name"
                        id="name"
                        type="text"
                        class="form-control"
                        placeholder="Enter your name" required/>
        </div>
        <div class="mb-3">
                <label for="email" class="form-label">Email:</label>
                <input v-model.trim="email"
                        id="email"
                        type="email"
                        class="form-control"
                        placeholder="Enter your email" required/>
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
</form>
<h3class="mt-4">Submitted Data:</h3>
<p>Name: {{name}}</p>
<p>Email: {{email}}</p>
</div>
</template>
```



THỰC HIỆN DEMO

Bước 2: Code javascript

```
<scriptsetup>
import { ref } from'vue';
const name = ref('');
const email = ref('');
const handleSubmit = () => {
console.log('Form submitted:', { name:name.value, email:email.value })
};
</script>
```



submit

SUMARRY

- ☑ Phần I: Event Handling
  - ❖ Listen to Events (Lắng nghe sự kiện)
  - ❖ Method Handlers (Trình xử lý phương thức)
  - ❖ Event modifiers, Key modifiers, Mouse Button Modifiers
- ☑ Phần II: Form Binding
  - ❖ Binding đơn giản
  - ❖ Ràng buộc giá trị


thank
you!