FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

THỰC HỌC – THỰC NGHIỆP

LẬP TRÌNH FRONT-END FRAMEWORK

COMPONENT TRONG VUEJS

http://www.poly.edu.vn

Mục tiêu

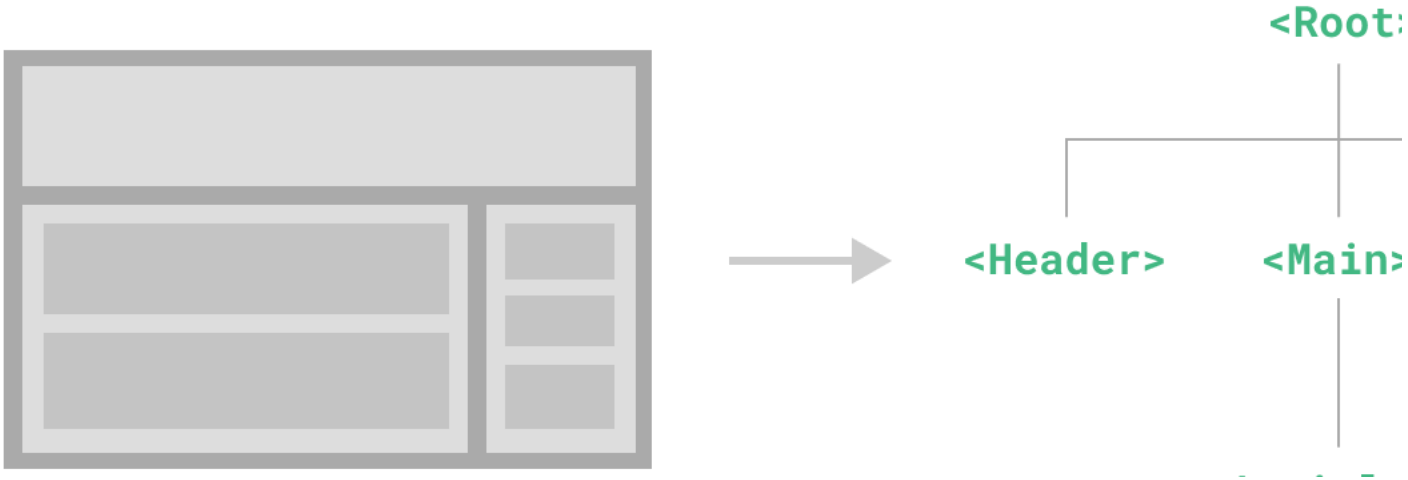
- Hiểu về component cơ bản, biết cách khai báo, sử dụng và nắm được kiến trúc component trong Vue
- Nắm được cách sử dụng Props để gửi dữ liệu từ component cha đến component con
- Nắm được cách sử dụng Emit() để gửi dữ liệu từ component con lên componet cha
- Hiểu về Slot, Provide và Inject để gửi dữ liệu không cần sử dụng Props

NỘI DUNG

- Phần 1: Component cơ bản
  - Tổng quan về component
  - Khai báo và sử dụng component
  - Props
  - Emit()
- Phần 2: Component nâng cao
  - Slots
  - Slot Named và Scoped Slots
  - Dynamic Slot Names
  - Project/Inject

PHẦN 1
TỔNG QUAN COMPONENT

COMPONENT LÀ GÌ?



- **Component** là một khối xây dựng cơ bản cho phép bạn chia nhỏ giao diện người dùng (UI) thành các phần nhỏ, độc lập và có thể tái sử dụng.
- Mỗi component có thể chứa HTML, CSS, và JavaScript riêng, giúp bạn dễ dàng quản lý và phát triển các phần khác nhau của ứng dụng.

KIẾN TRÚC CỦA COMPONENT

Một component trong Vue thường được chia thành 3 phần chính:

1. Template (HTML)

```
<template>
        <div>
        <h1>{{title}}</h1>
        <p>{{message}}</p>
        </div>
</template>
```

2. Script (JavaScript)

```
<script setup>
import { ref } from"vue";
const title = ref("Welcome to Vue Component");
const message = ref("This is a reusable component.");
function greet() {
        console.log("Hello from the component!");
}
</script>
```

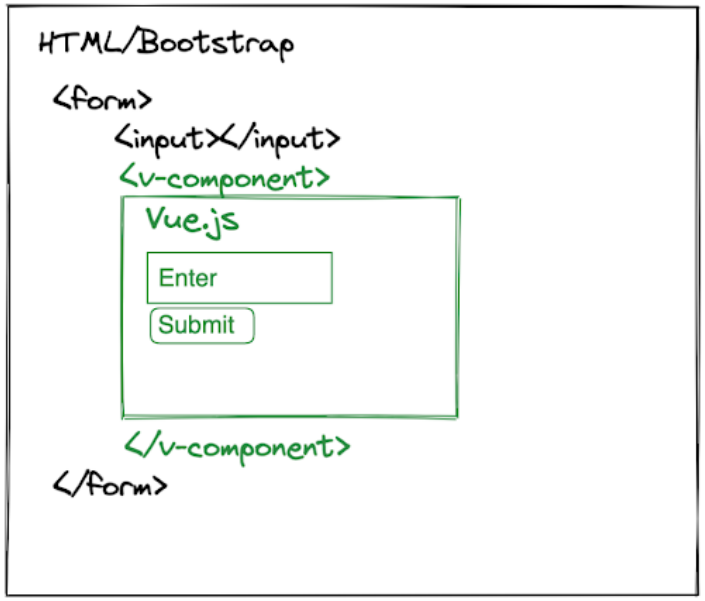
3.Style (CSS)

```
<style scoped>
h1 { color: blue; }
</style>
```

FPT Education
FPT POLYTECHNIC

TẠI SAO PHẢI SỬ DỤNG COMPONENT?

- **Tái sử dụng**: có thể dùng lại trong nhiều phần của ứng dụng, giảm thiểu lặp lại mã.
- **Độc lập**: Hoạt động độc lập, tương tác dữ liệu qua props và events.
- **Đóng gói**: Chứa HTML, CSS, và JS trong một khối duy nhất, dễ quản lý và bảo trì.
- **Tổ chức cây**: tổ chức thành một cây component lồng nhau, với component gốc ở đỉnh.



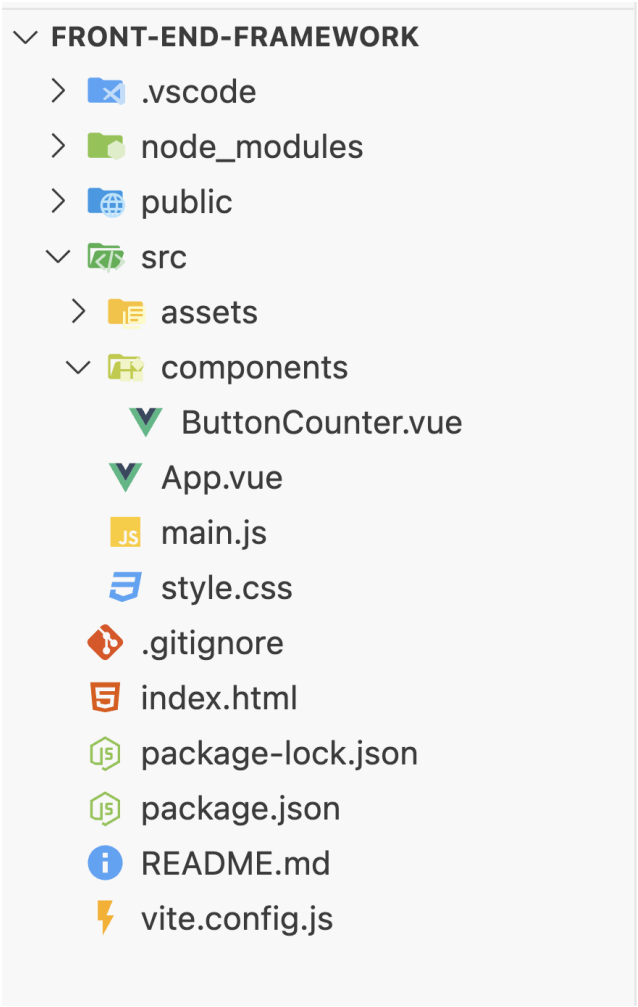
KHAI BÁO COMPONENT

- Chúng ta thường định nghĩa mỗi component Vue trong một tệp riêng biệt với đuôi **.vue** - được gọi là Single-File Component (viết tắt SFC).

```
<script setup>
import { ref } from"vue";
const count = ref(0);
</script>
<template>
<button @click="count++">You clicked me {{count}} times.</button>
</template>
```

- **Single-File Component** (SFC) là cách định nghĩa một component trong một tệp duy nhất với ba phần chính:
  1. **Template**: HTML xác định cấu trúc giao diện.
  2. **Script**: JavaScript chứa logic và trạng thái của component.
  3. **Style**: CSS để định dạng giao diện, **scoped** để chỉ áp dụng cho component đó.

SỬ DỤNG COMPONENT



ButtonCounter.vue

```
<script setup>
        import { ref } from "vue";
        const count = ref(0);
</script>
<template>
<button @click="count++">You clicked {{count}} times.</button>
</template>
```

App.vue

```
<script setup>
import ButtonCounter from"./components/ButtonCounter.vue";
</script>
<template>
        <div id="app">
                <ButtonCounter />
        </div>
</template>
```

TÁI SỬ DỤNG COMPONENT

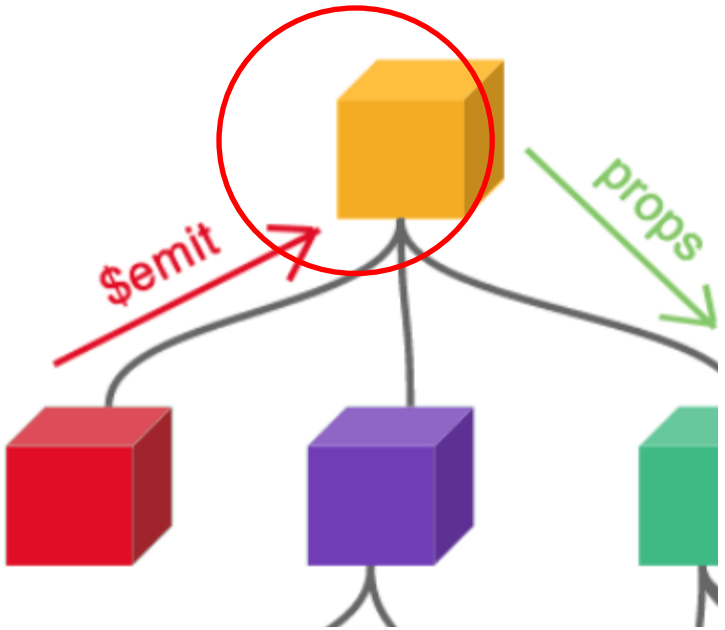
- Các component có thể được tái sử dụng nhiều lần theo ý muốn:

```
<script setup>
import ButtonCounter from"./components/ButtonCounter.vue";
</script>
<template>
<div id="app">
        <!-- Cú pháp PascalCase -->
        <ButtonCounter />
        <!-- Bạn cũng có thể sử dụng cú pháp kebab-case -->
        <button-counter></button-counter>
</div>
</template>
```

- Lưu ý rằng khi nhấp vào các nút, mỗi nút sẽ duy trì một bộ đếm riêng biệt. Đó là vì mỗi khi bạn sử dụng một component, một instance mới của nó sẽ được tạo ra.

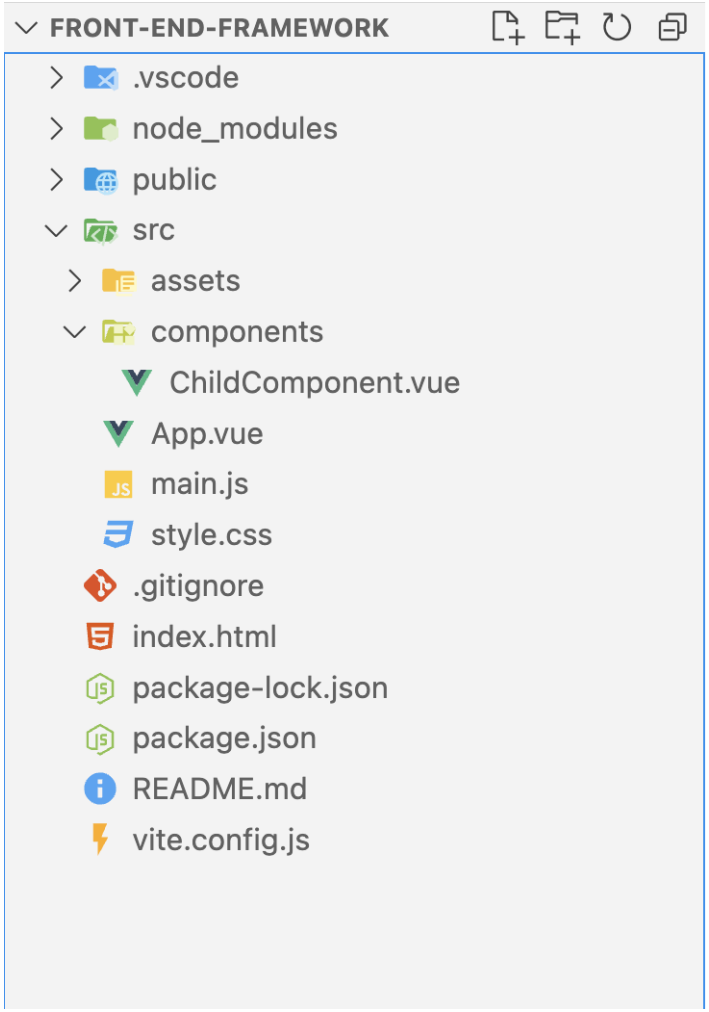
TRUYỀN PROPS

- **Props** là các giá trị được truyền từ component cha xuống component con hoặc gửi dữ liệu từ component con lên component cha
- Tương tự như các tham số truyền vào một hàm, props cho phép component tái sử dụng với dữ liệu khác nhau.
- Component con sử dụng **defineProps** để khai báo các props nhận được



FPT Education
FPT POLYTECHNIC

CÁCH SỬ DỤNG PROPS



ChildComponent.vue

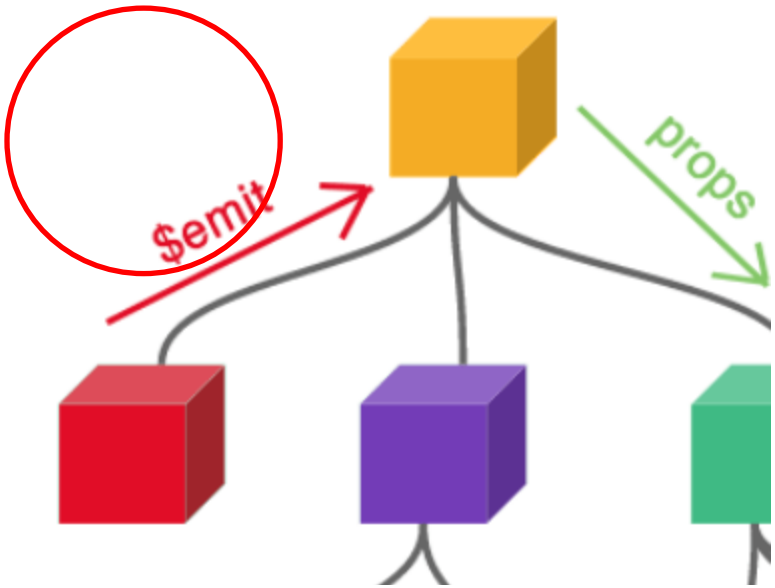
```
<template>
        <h2>Child Component</h2>
<!-- Hiển thị các giá trị được truyền từ component cha -->
        <p>{{message}}</p>
        <p>Count: {{count}}</p>
</template>
<script setup>
// Sử dụng defineProps để khai báo props trong Composition API
const props = defineProps({
message: String,
count: Number,
});
</script>
```

App.vue

```
<script setup>
// Import component con
import ChildComponent from"./components/ChildComponent.vue";
</script>
<template>
<div>
<h1>Parent Component</h1>
<!-- Truyền props 'message' và 'count' đến ChildComponent -->
<ChildComponent message="Hello from parent!" :count="5" />
</div>
</template>
```

EMIT LÀ GÌ?

- **emit** là cơ chế để component con phát ra sự kiện và component cha có thể lắng nghe các sự kiện đó. Điều này giúp tương tác giữa các component mà không cần chia sẻ dữ liệu trực tiếp.



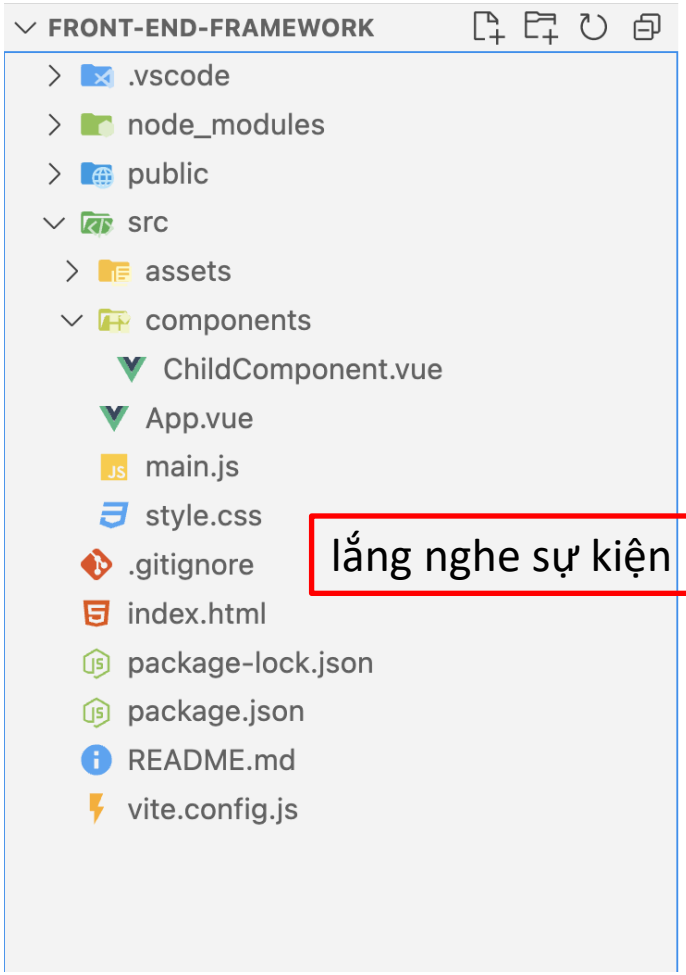
QUY TRÌNH CỦA EMIT TRONG VUE 3

- **Khai báo sự kiện**: Component con khai báo sự kiện với **defineEmits()**.
- **Phát sự kiện**: Component con phát sự kiện bằng cách gọi emit.
- **Lắng nghe sự kiện**: Component cha lắng nghe sự kiện từ component con bằng v-on hoặc @.
- **Xử lý sự kiện**: Component cha xử lý sự kiện khi nó được phát ra từ component con.

```
v-on:emitEvent='parentEventHandler'>
</ChildComponent>
```

Emit out

SỬ DỤNG EMIT



lắng nghe sự kiện

ChildComponent.vue

```
<template>
<div>
<h2>Child Component</h2>
<button @click="notifyParent">Click Me</button>
</div>
</template>
<scriptsetup>
// Sử dụng `emit` trong Composition API
const emit = defineEmits(["childEvent"]);
// Hàm phát sự kiện 'childEvent' khi người dùng nhấn nút
const notifyParent= () => {
emit("childEvent", "Hello from child!");
};
</script>
```

App.vue

```
<script setup>
import { ref } from"vue";
import ChildComponent from"./components/ChildComponent.vue";
// Khai báo ref để lưu trữ dữ liệu sự kiện
const messageFromChild = ref("");
// Hàm xử lý sự kiện từ component con
const handleChildEvent = (message) => {
messageFromChild.value = message; // Cập nhật giá trị của ref
};
</script>
<template>
<h1>Parent Component</h1>
<!-- Lắng nghe sự kiện 'childEvent' từ ChildComponent -->
<ChildComponent @childEvent="handleChildEvent" />
<!-- Hiển thị giá trị từ ref -->
<p>Message from child: {{messageFromChild}}</p>
</template>
```

Xây dựng một ứng dụng đơn giản với hai component: **ParentComponent** và **ChildComponent**. Component cha hiển thị một nút để gọi component con, component con sẽ phát sự kiện ngược lên để thay đổi thông báo trong component cha.

THỰC HIỆN DEMO

Giải pháp: Ứng dụng sẽ gồm hai file component:

1. ChildComponent (component con): Bao gồm một nút, khi nhấn vào nút này sẽ phát sự kiện gửi thông báo mới lên component cha.
2. ParentComponent (component cha): Chịu trách nhiệm hiển thị thông báo và lắng nghe sự kiện từ component con.

Bước 1: Xây dựng component con **ChildComponent.vue**

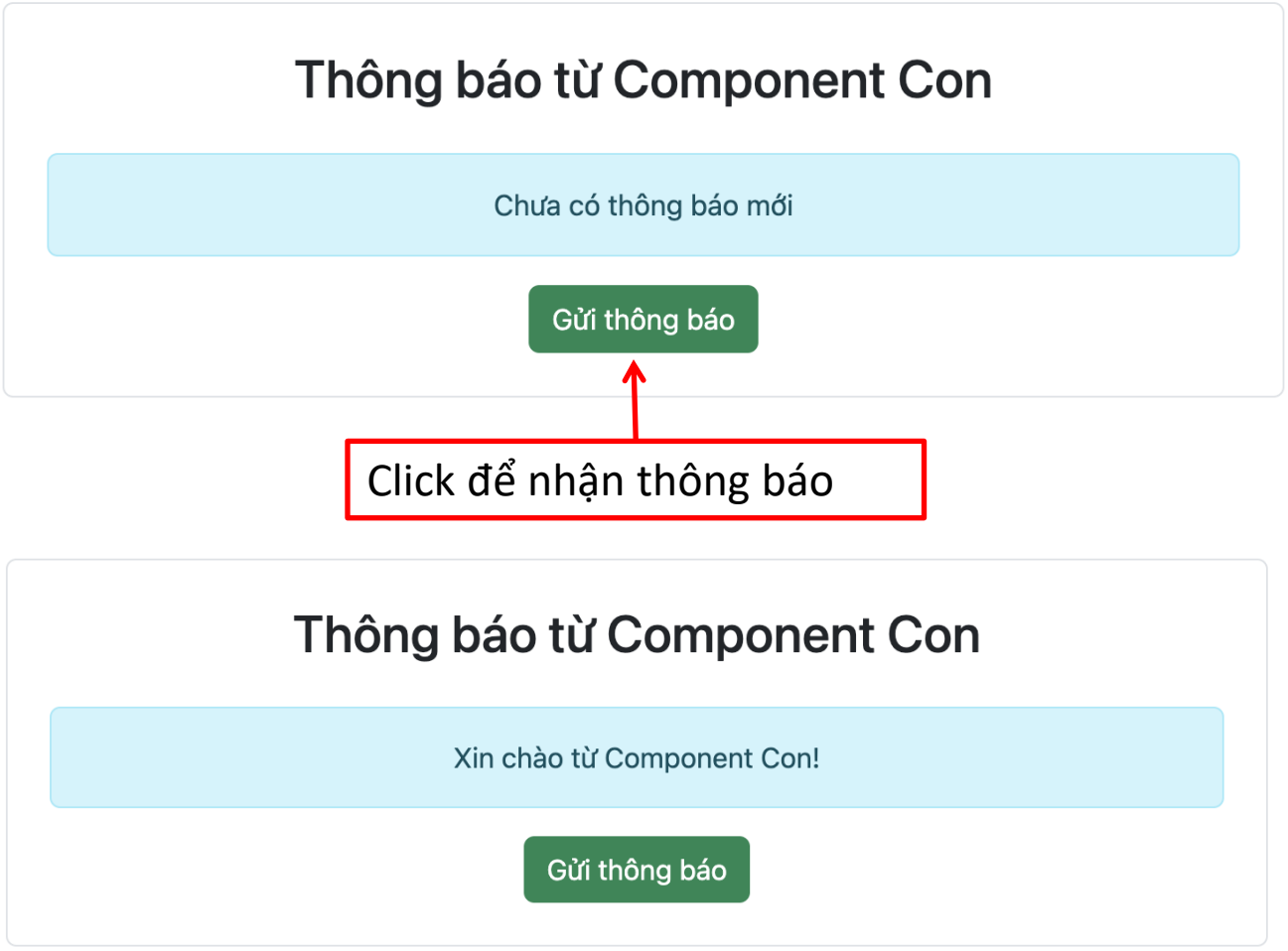
```
<template>
<div class="d-flex justify-content-center">
<button @click="sendMessage" class="btn btn-success">Gửi thông báo</button>
</div>
</template>
<script setup>
// Phát sự kiện 'update-message' khi nhấn nút
const emit = defineEmits(["update-message"]);
const sendMessage= () => {
        emit("update-message", "Xin chào từ Component Con!");
};
</script>
```

Bước 2: Xây dựng component con **ParentComponent.vue**

```
<template>
        <div class="container mt-5 p-4 border rounded">
                <h2 class="text-center mb-4">Thông báo từ Component Con</h2>
                <div class="alert alert-info text-center">
                 {{message}}
                </div>
                <!-- Gọi component con và lắng nghe sự kiện 'update-message' -->
                <ChildComponent @update-message="updateMessage" />
        </div>
</template>
<script setup>
import { ref } from'vue';
import ChildComponent from'./ChildComponent.vue';
// Khai báo biến lưu trữ thông báo
const message = ref('Chưa có thông báo mới');
// Hàm xử lý sự kiện từ component con
const updateMessage= (newMessage) => {
        message.value = newMessage;
};
</script>
```

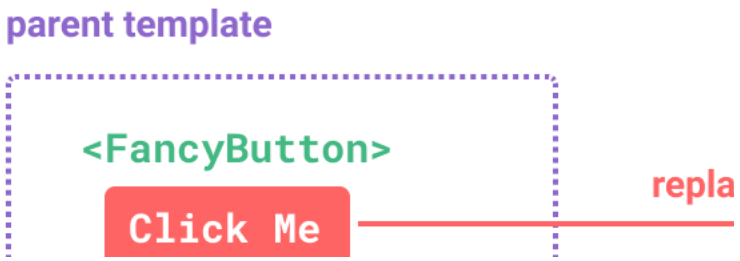
THỰC HIỆN DEMO

Kết quả:



PHẦN 2: COMPONENT NÂNG CAO

Slots là một cơ chế trong Vue cho phép chúng ta chèn nội dung từ component cha vào component con.



Ví dụ:

ChildComponent.vue

```
<template>
<div>
<h2>Tiêu đề từ component cha:</h2>
<slot></slot>
</div>
</template>
```

ParentComponent.vue

```
<script setup>
import ChildComponent from"./components/ChildComponent.vue";
</script>
<template>
<ChildComponent>
<p>Đây là nội dung được truyền từ component cha vào component con.</p>
</ChildComponent>
</template>
```

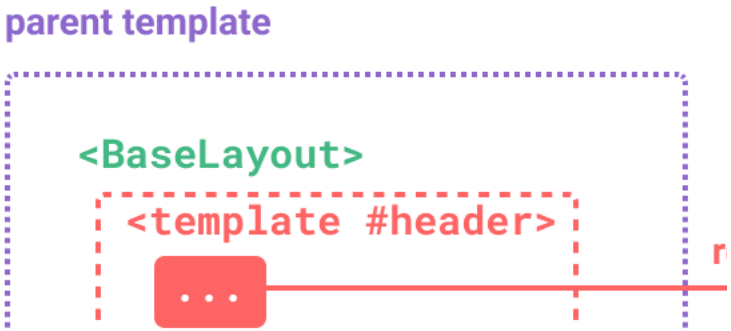
Kết quả

Tiêu đề từ component cha:

Đây là nội dung được truyền từ component cha vào component con.

SLOTS NAMED VÀ SCOPED SLOTS

Named slots cho phép bạn chỉ định vị trí cụ thể trong component con mà nội dung được chèn vào.



Ví dụ

ChildComponent.vue

```
<template>
<header><slot name="header"></slot></header>
<footer><slot name="footer"></slot></footer>
</template>
```

Kết quả

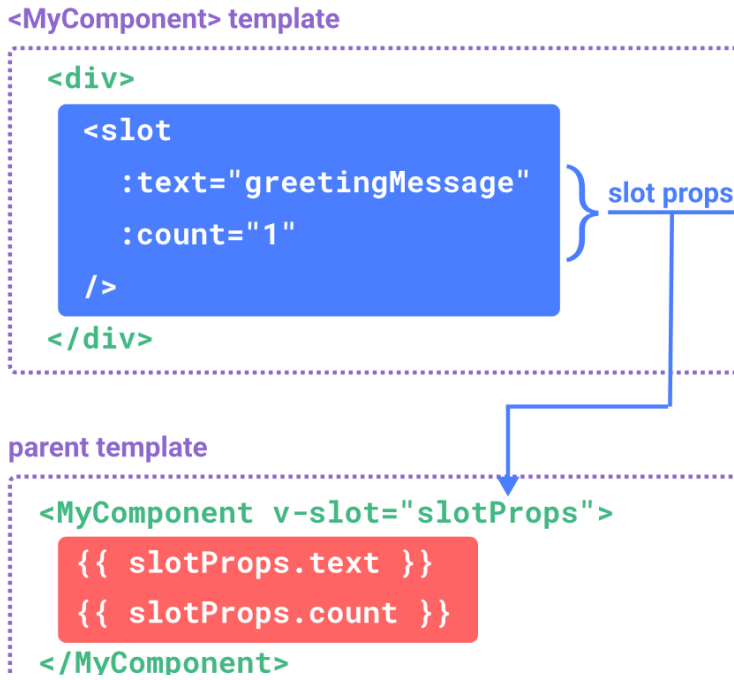
Đây là header được truyền từ component cha
Đây là footer được truyền từ component cha

ParentComponent.vue

```
<script setup>
import ChildComponent from "./components/ChildComponent.vue";
</script>
<template>
<ChildComponent>
        <template v-slot:header>Đây là header được truyền từ component cha</template>
        <template v-slot:footer>Đây là footer được truyền từ component cha</template>
</ChildComponent>
</template>
```

SLOTS NAMED VÀ SCOPED SLOTS

Scoped slots cho phép truyền dữ liệu từ component con lên component cha qua slot.



Ví dụ:

ChildComponent.vue

```
<template>
<slot :message="greeting"></slot>
</template>
<script setup>
const greeting = "Đây là message từ ChildComponent";
</script>
```

ParentComponent.vue

```
<script setup>
import ChildComponent from"./components/ChildComponent.vue";
</script>
<template>
<ChildComponent v-slot="{ message }">
        <p>{{message}}</p>
</ChildComponent>
</template>
```

Kết quả

Đây là message từ ChildComponent

Dynamic Slot Names

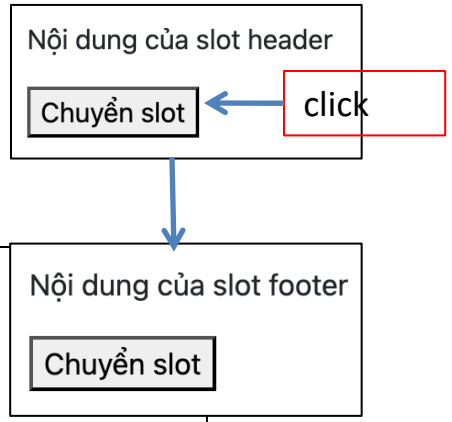
Dynamic Slot Names cho phép sử dụng tên slot thay đổi theo giá trị động, thay vì tên tĩnh như header hay footer.

ChildComponent.vue

```
<template>
        <div>
        <slotname="header"></slot>
        <slotname="footer"></slot>
        </div>
</template>
```

ParentComponent.vue

```
<template>
        <ChildComponent>
                <template #[dynamicSlotName]>
                        <p>Nội dung của slot {{dynamicSlotName}}</p>
                </template>
        </ChildComponent>
        <button @click="switchSlot">Chuyển slot</button>
</template>
<script setup>
        import ChildComponent from"./components/ChildComponent.vue";
        import { ref } from"vue";
        const dynamicSlotName = ref("header"); // Tên slot động
        const switchSlot = () => {
        dynamicSlotName.value = dynamicSlotName.value === "header" ? "footer" : "header";
};
</script>
```



Dynamic Slot Names

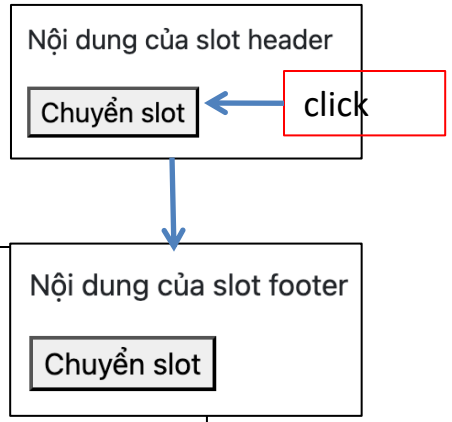
Dynamic Slot Names cho phép sử dụng tên slot thay đổi theo giá trị động, thay vì tên tĩnh như header hay footer.

ChildComponent.vue

```
<template>
        <div>
        <slot name="header"></slot>
        <slot name="footer"></slot>
        </div>
</template>
```

ParentComponent.vue

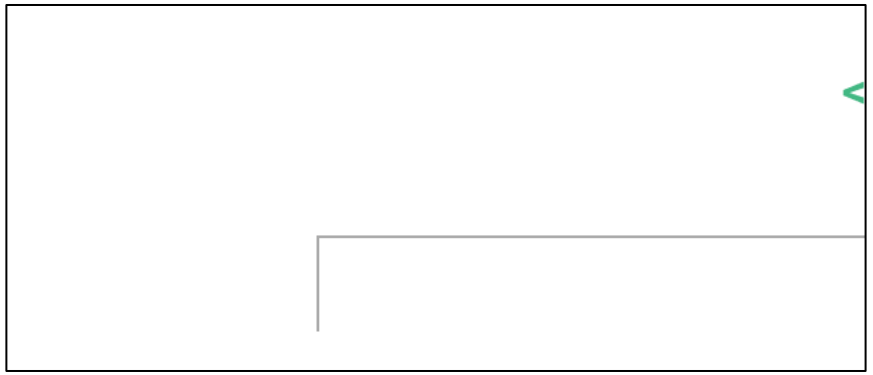
```
<template>
        <ChildComponent>
                <template #[dynamicSlotName]>
                        <p>Nội dung của slot {{dynamicSlotName}}</p>
                </template>
        </ChildComponent>
        <button @click="switchSlot">Chuyển slot</button>
</template>
<script setup>
        import ChildComponent from"./components/ChildComponent.vue";
        import { ref } from "vue";
        const dynamicSlotName = ref("header"); // Tên slot động
        const switchSlot = () => {
        dynamicSlotName.value = dynamicSlotName.value === "header" ? "footer" : "header";
};
</script>
```



PROVIDE / INJECT

- Theo cách thông thường, muốn truyền dữ liệu được từ component cha đến component con, chúng ta cần sử dụng props

- Tuy nhiên có 1 cách khác nhanh hơn đó là sử dụng **Provide / Inject**



- **Provide / Inject** trong Vue là cơ chế chia sẻ dữ liệu giữa các component mà **không cần truyền props** qua nhiều cấp trung gian. Component cha "cung cấp" dữ liệu qua provide, và component con "nhận" dữ liệu qua inject.

PROVIDE

- **Provide** trong Vue là cách để cung cấp dữ liệu từ component cha cho các component con. Để sử dụng, bạn dùng hàm provide().
- Cú pháp sử dụng provide:

```
<!-- Component cha -->
<script setup>
import { provide, ref } from"vue";
const message = ref("Hello from parent!");
provide("message", message);
</script>
```

Tham số của provide:

1. Injection Key: Chuỗi hoặc Symbol, được dùng để tra cứu giá trị khi các component con muốn nhận dữ liệu.
2. Provided Value: Giá trị có thể là bất kỳ loại dữ liệu nào, bao gồm cả các giá trị reactive như ref.

INJECT

- **inject** trong Vue cho phép component con nhận dữ liệu từ component cha mà không cần truyền qua từng cấp qua props. Dữ liệu được cha cung cấp qua provide và con nhận qua inject.
- Cú pháp sử dụng inject:

```
<!-- Component con -->
<template>
<p>Tin nhắn từ component cha: {{message}}</p>
</template>
<script setup>
import { inject } from"vue";
const message = inject("message", "No message provided");
</script>
```

- Nếu không có giá trị nào được cung cấp từ component cha, bạn có thể chỉ định một giá trị mặc định cho inject (**tham số thứ 2 của inject()**)

Tạo một ví dụ về việc chia sẻ dữ liệu giữa các component trong Vue bằng cách sử dụng provide và inject. Trong đó, component cha sẽ cung cấp một thông điệp và component con sẽ nhận và hiển thị thông điệp này mà không cần truyền qua props.

THỰC HIỆN DEMO

❑ Bước 1: Tạo component cha - ParentComponent.vue

```
<template>
        <h2>Component Cha</h2>
        <p>Thông điệp từ cha: {{message}}</p>
        <ChildComponent />
</template>
<script setup>
import { ref, provide } from"vue";
import ChildComponent from"./components/ChildComponent.vue";
const message = ref("Hello from Parent Component!");
provide("message", message);
</script>
```

❑ Bước 2: Tạo component con- ChildComponent.vue

```
<template>
        <h2>Component Con</h2>
        <p>Nhận thông điệp: {{message}}</p>
</template>
<script setup>
import { inject } from'vue';
const message = inject('message');
</script>
```

kết quả

Component Cha

Thông điệp từ cha: Hello from Parent Component!

Component Con

Nhận thông điệp: Hello from Parent Component!

SUMARRY

- ☑ Phần 1: Component cơ bản
  - ❖ Tổng quan về component
  - ❖ Khai báo và sử dụng component
  - ❖ Props
  - ❖ Emit()
- ☑ Phần 2: Component nâng cao
  - ❖ Slots
  - ❖ Slot Named và Scoped Slots
  - ❖ Dynamic Slot Names
  - ❖ Project/Inject


thank
you!