



Conceive Design Implement Operate



THỰC HỌC – THỰC NGHIỆP

FRONT-END FRAMEWORK

TỔNG QUAN VỀ VUE.JS

- Nắm được các khái niệm cơ bản về Vue.js
- Cài đặt môi trường phát triển Vue.js với thư viện Vitejs
- Giới thiệu về các cú pháp trong Vue.js
- Hiểu được cách sử dụng Bootstrap để tạo giao diện trong Vue



 Framework là gì?

 Giới thiệu về Framework VueJS

 Cài đặt môi trường phát triển

 Tạo và thực thi Project với Vue.Js

 Cài đặt và sử Bootstrap trong Vuejs

 Giới thiệu một vài cú pháp trong VueJS





PHẦN 1: TỔNG QUAN VỀ VUEJS

- Framework là bộ công cụ lập trình.
- Giúp tổ chức và quản lý mã nguồn tốt hơn
- Tiết kiệm thời gian và công sức khi phát triển website
- Giúp cải thiện tốc độ tải và phản hồi của ứng dụng.



- Vue** (phát âm /vju:/, như “view”) là một framework JavaScript
- Dùng để xây dựng các giao diện người dùng (UI).
- Cú pháp đơn giản và thân thiện với người mới bắt đầu.
- Sử dụng Virtual DOM để tối ưu hiệu năng ứng dụng



TẠI SAO DÙNG VUEJS?



HTML/CSS/JS
tách biệt

Reactive Data
Binding

Single-File
Components

Làm việc tốt với
Back-end

- ❑ **VueJS** được tạo bởi Evan You,
cựu nhân viên Google
- ❑ **VueJS** bắt đầu phát triển vào
năm 2013
- ❑ Ra mắt phiên bản đầu tiên chính
thức năm 2014
- ❑ Phiên bản mới nhất tính đến thời
điểm hiện tại (7/2024) là v3.4.33



VUE HOẠT ĐỘNG NHƯ NÀO?

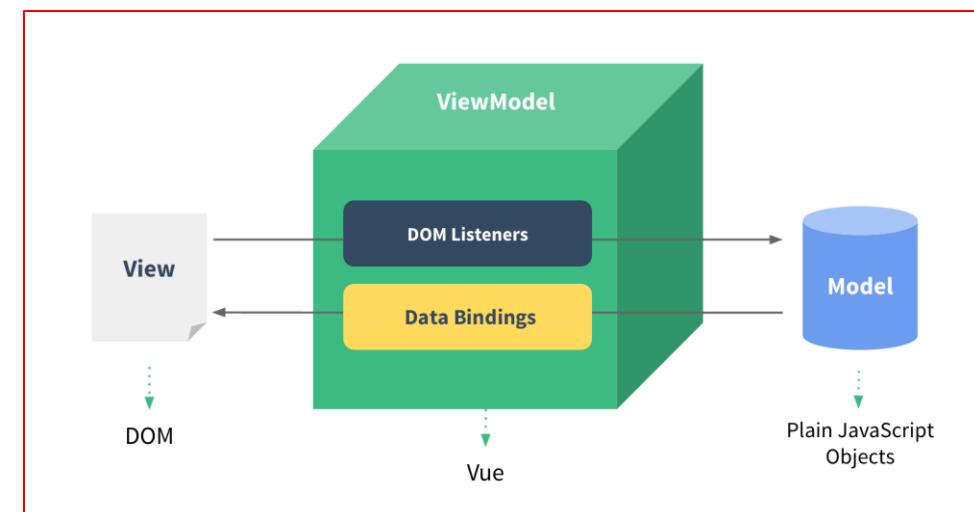
❑ Vue.js tự động đồng bộ dữ liệu giữa Model và View thông qua ViewModel, giúp mã nguồn dễ bảo trì và phát triển hơn. Mô hình này được gọi là **MVVM**

❑ **View:** Giao diện người dùng

❑ **ViewModel:**

- Data Bindings: Kết nối dữ liệu từ Model với View.
- DOM Listeners: Theo dõi sự kiện trên DOM và cập nhật Model.

❑ **Model:** Chứa dữ liệu.





Visual Studio Code

```
4. vue create hello-world (node)
Vue CLI v3.4.0
? Please pick a preset: Manually select features
? Check the features needed for your project:
> Babel
  o TypeScript
  o Progressive Web App (PWA) Support
  o Router
  o Vuex
  o CSS Pre-processors
  ● Linter / Formatter
  o Unit Testing
  o E2E Testing
```



- Truy cập: <https://nodejs.org>
- Tải và cài đặt

Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

[Download Node.js \(LTS\)](#)

Downloads Node.js v20.16.0¹ with long-term support.
Node.js can also be installed via package managers.

Want new features sooner? Get [Node.js v22.5.1¹](#) instead.



The screenshot shows the Node.js website's landing page. On the left, there's a green hexagonal background with the text "Run JavaScript Everywhere". Below it is a green button labeled "Download Node.js (LTS)". A red arrow points from this button towards the code editor on the right. On the right, there's a dark-themed code editor window with the following Node.js code:

```
1 // server.mjs
2 import { createServer } from 'node:http';
3
4 const server = createServer((req, res) => {
5   res.writeHead(200, { 'Content-Type': 'text/plain' });
6   res.end('Hello World!\n');
7 });
8
9 // starts a simple http server locally on port 3000
10 server.listen(3000, '127.0.0.1', () => {
11   console.log('Listening on 127.0.0.1:3000');
12 });
13
14 // run with `node server.mjs`
```

Below the code editor, there's a "JavaScript" tab and a "Copy to clipboard" button.

Learn more what Node.js is able to offer with our Learning materials.

- ❑ Cách 1: CDN (Content delivery Network)
- ❑ Tạo file index.html và nhúng link sau ở đầu file

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<div id="app">{{ message }}</div>

<script>
    const { createApp, ref } = Vue;

    createApp({
        setup() {
            const message = ref("Hello vue!");
            return {
                message,
            };
        },
    }).mount("#app");
</script>
```

➤ Xem ví dụ

Cách 2: Sử dụng NPM cài đặt Vitejs

- Vitejs** là công cụ để tạo 1 project
Vuejs cơ bản và có hiệu suất cao
- Do chính tác giả Vue tạo ra
- Sử dụng **npm** để tải **Vitejs**
- Sử dụng Terminal (hoặc Command Prompt):



```
npm create vue@latest front-end-framework
```

CÀI ĐẶT MÔI TRƯỜNG VUEJS

- Lệnh này sẽ cài đặt và thực thi **create-vue**, công cụ tạo dự án chính thức của **Vue**.
- Bạn sẽ được yêu cầu chọn các tính năng tùy chọn như hình sau.

- ✓ Add TypeScript? ... No / Yes
- ✓ Add JSX Support? ... No / Yes
- ✓ Add Vue Router for Single Page Application development? ... No / Yes
- ✓ Add Pinia for state management? ... No / Yes
- ✓ Add Vitest for Unit testing? ... No / Yes
- ✓ Add an End-to-End Testing Solution? ... No / Cypress / Nightwatch / Playwright
- ✓ Add ESLint for code quality? ... No / Yes
- ✓ Add Prettier for code formatting? ... No / Yes
- ✓ Add Vue DevTools 7 extension for debugging? (experimental) ... No / Yes

Scaffolding project in ./<front-end-framework>...
Done.

CÀI ĐẶT MÔI TRƯỜNG VUEJS

☐ Sử dụng Terminal VSC:

```
cd front-end-framework  
npm install  
npm run dev
```

Truy cập thư mục

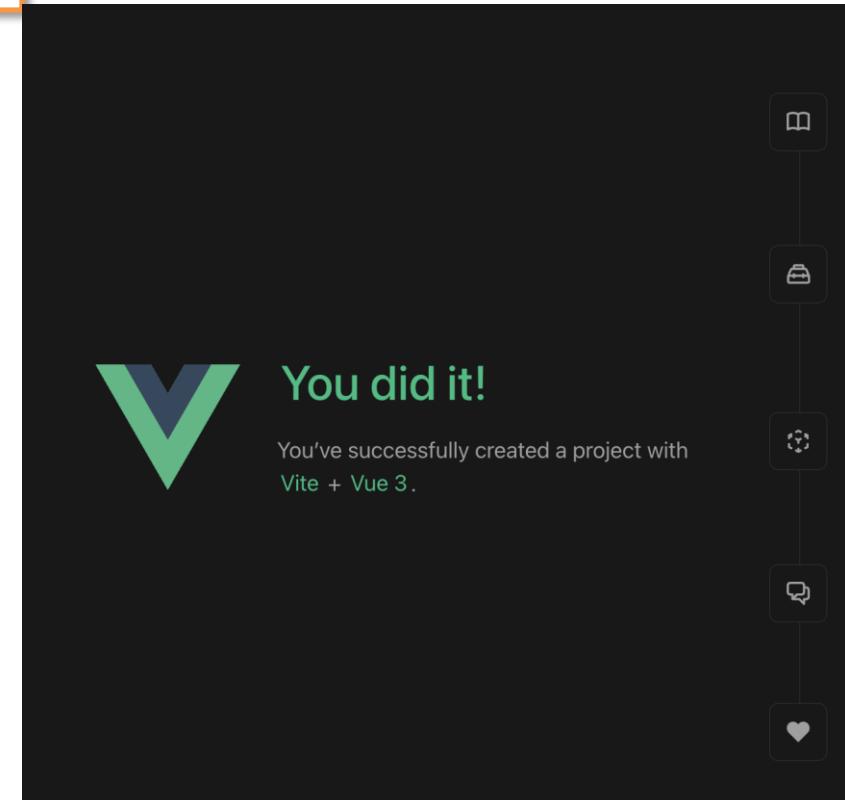
Cài đặt các thư viện

Chạy ứng dụng Vue

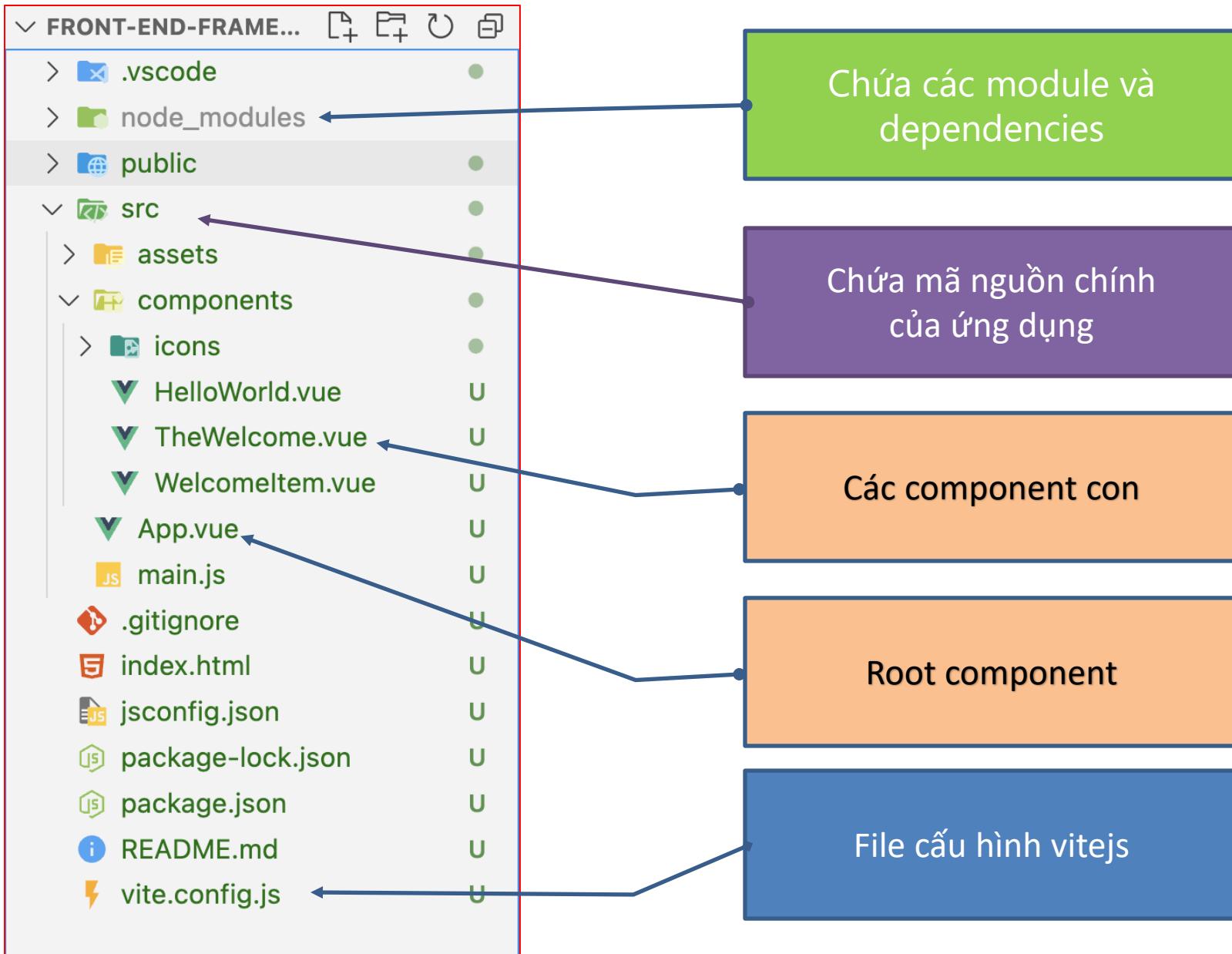
☐ Bật trình duyệt

VITE v5.3.5 ready in 659 ms

- Local: <http://localhost:5173/>
- Network: use **--host** to expose
- press **h + enter** to show help



TỔ CHỨC CODE TRONG VUEJS



- ❑ Mọi ứng dụng Vue đều bắt đầu bằng việc tạo một application instance mới với hàm **createApp**:

```
import { createApp } from "vue";
const app = createApp({
    /* Các tùy chọn của thành phần gốc */
});
```

- ❑ Thành phần gốc(root component) là thành phần đầu tiên trong một ứng dụng Vue.
- ❑ Chứa các thành phần con và là điểm khởi đầu của ứng dụng.
- ❑ Được truyền vào hàm **createApp** để tạo ra ứng dụng.
- ❑ Ví dụ: App.vue thường là thành phần gốc.

```
// main.js là file đầu tiên được chạy khi ứng dụng Vue.js được khởi tạo.
import { createApp } from "vue";
// Nhập thành phần gốc App từ một single-file component
import App from "./App.vue";
createApp(App).mount("#app");
```

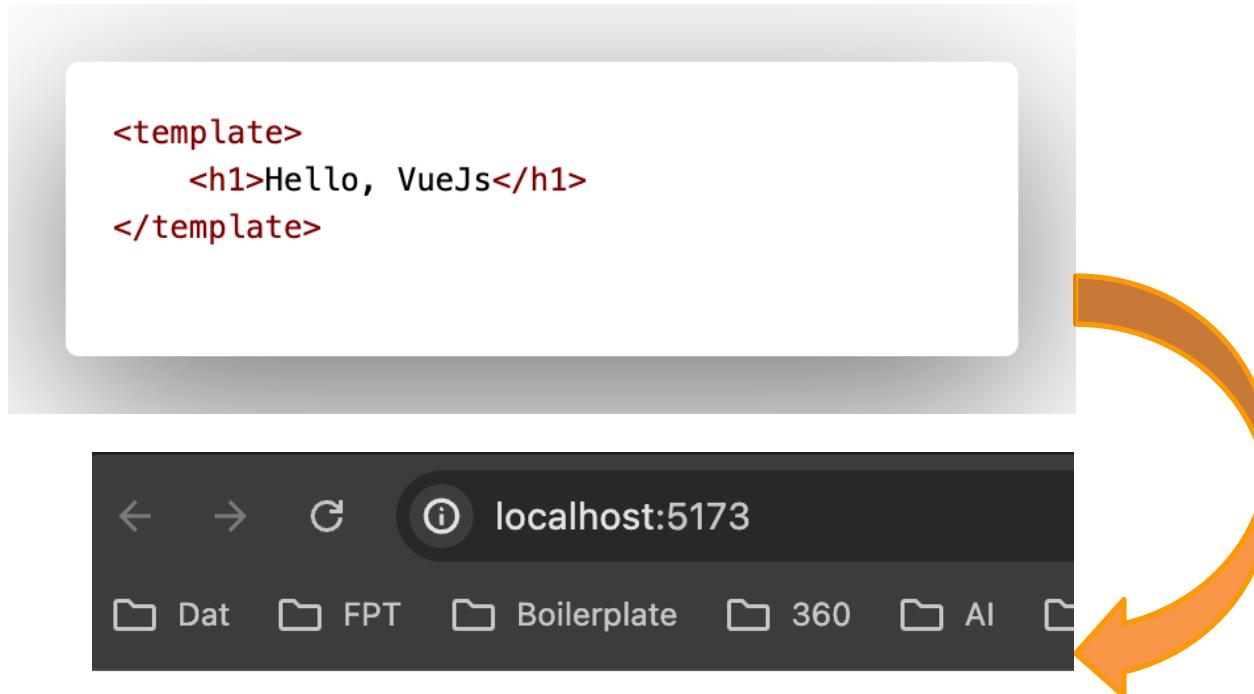
- **<script setup>**: Đơn giản hóa khai báo và sử dụng dữ liệu trong Vue 3.
- **<template>**: Định nghĩa cấu trúc giao diện của thành phần bằng HTML.
- **<style scoped>**: Áp dụng CSS chỉ cho thành phần hiện tại.

```
<script setup>
// Code js của bạn ở đây
</script>

<template>
    <!-- Code html của bạn ở đây -->
</template>

<style scoped>
/* Code css của bạn ở đây */
</style>
```

- ☐ Truy cập file **App.vue** cập nhật code sau và lưu lại.



Hello, VueJS!

➤ [Xem ví dụ](#)



demo



PHẦN 2: STYLE VÀ TEMPLATE SYNTAX

Sử dụng framework css **Bootstrap** ta thực hiện 2 bước

- Cài đặt Bootstrap: **npm i bootstrap**
- Để nhúng thư viện bootstrapp vào dự án vue:
Mở tệp **main.js** và thêm các dòng sau:

```
import { createApp } from "vue";
import App from "./App.vue";

import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.bundle.min.js";

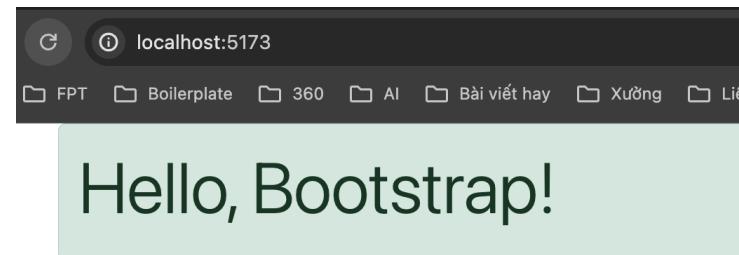
createApp(App).mount("#app");
```

- [Xem ví dụ](#)

Kiểm tra bootstrap có hoạt động chưa?

Mở file **App.vue** trong thư mục app và cập nhật code như sau:

```
<template>
  <div id="app">
    <div class="container">
      <div class="alert alert-success">
        <h1 class="display-4">Hello, Bootstrap!</h1>
      </div>
    </div>
  </div>
</template>
```



➤ Xem ví dụ

CÚ PHÁP TEMPLATE TRONG VUE

- ❑ **Vue** sử dụng cú pháp mẫu dựa trên HTML cho phép bạn liên kết một cách khai báo giữa DOM được render với dữ liệu của của đối tượng Vue bên dưới
- ❑ Hình thức ràng buộc dữ liệu cơ bản nhất là text interpolation
- ❑ Cú pháp “mustache” với hai dấu ngoặc. :
`Message: {{ msg }}`

- **<script setup>**: Đơn giản hóa khai báo và sử dụng dữ liệu trong Vue 3.
- **<template>**: Định nghĩa cấu trúc giao diện của thành phần bằng HTML.
- **<style scoped>**: Áp dụng CSS chỉ cho thành phần hiện tại.

```
<script setup>
// Code js của bạn ở đây
</script>

<template>
    <!-- Phần này là code UI của bạn -->
</template>

<style scoped>
/* Code css của bạn ở đây */
</style>
```

```
<script setup>
const courseName = "Framework Vue 3";
const courseLevel = "Nâng cao";
const courseTime = 30; // giờ
const coursesActive = true;
</script>
<template>
<div class="container my-5">
    <h1 class="display-4 mb-3">Thông tin:</h1>
    <ul class="list-group">
        <li class="list-group-item">
            Tên khóa học: <strong>{{courseName}}</strong>
        </li>
        <li class="list-group-item">Cấp độ: {{courseLevel}}</li>
        <li class="list-group-item">Thời gian: {{courseTime}} giờ</li>
        <li class="list-group-item">
            Trạng thái: {{coursesActive ? "Đang mở" : "Đã đóng"}}
        </li>
    </ul>
</div>
</template>
```

Dữ liệu kiểu chuỗi

Thông tin khóa học:

Tên khóa học: Framework Vue 3

Cấp độ: Nâng cao

Thời gian: 30 giờ

Trạng thái: Đang mở

➤ [Xem Ví dụ](#)



demo

Ngoài ra còn rất nhiều cú pháp khác như:

- ❖ Attribute Binding: **v-bind**
- ❖ Conditional Rendering: **v-if, v-else, v-else-if**
- ❖ List Rendering: **v-for**
- ❖ Two-way Binding: **v-model**
- ❖ Event Handling: **v-on**
- ❖ Conditional Display: **v-show**
- ❖ Class Binding: **v-bind:class**
- ❖ Style Binding: **v-bind:style**

Chúng ta sẽ vào chi tiết ở các bài tiếp theo

- Framework là gì?
- Giới thiệu về Framework VueJS
- Môi trường phát triển
- Cài đặt
- Tạo và thực thi Project với Vue.Js
- Kiến trúc tổ chức của Vue.Js
- Cài đặt và sử Bootstrap trong Vuejs
- Giới thiệu một vài cú pháp trong VueJS



thank
you!



THỰC HỌC – THỰC NGHIỆP

LẬP TRÌNH FRONT-END FRAMEWORK

COMPONENT TRONG VUEJS

- 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 component cha
- Hiểu về Slot, Provide và Inject để gửi dữ liệu không cần sử dụng Props



📖 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

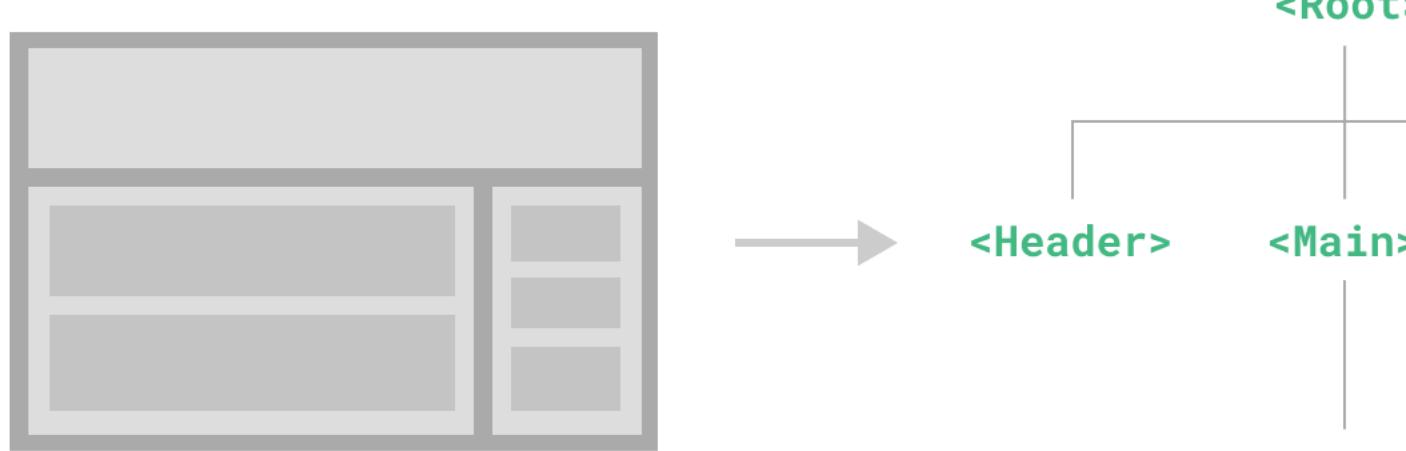




PHẦN 1

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.

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>
```

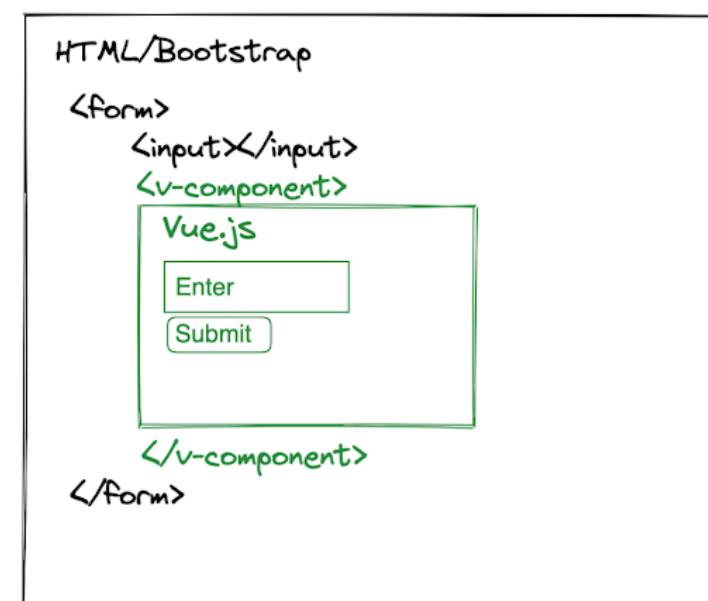
3. Style (CSS)

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



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.



- 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:
 - Template**: HTML xác định cấu trúc giao diện.
 - Script**: JavaScript chứa logic và trạng thái của component.
 - Style**: CSS để định dạng giao diện, **scoped** để chỉ áp dụng cho component đó.

SỬ DỤNG COMPONENT

FRONT-END-FRAMEWORK

- > .vscode
- > node_modules
- > public
- > src
 - > assets
 - > components
 - ✓ ButtonCounter.vue
 - ✓ App.vue
 - ✓ main.js
 - ✓ style.css
- ✓ .gitignore
- ✓ index.html
- ✓ package-lock.json
- ✓ package.json
- ✓ README.md
- ✓ vite.config.js

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.

You clicked me 0 times.

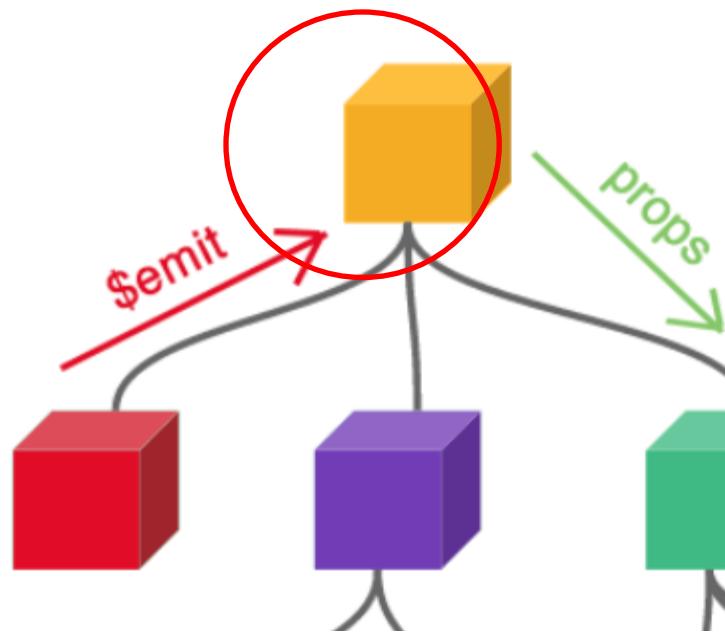
You clicked me 0 times.

click

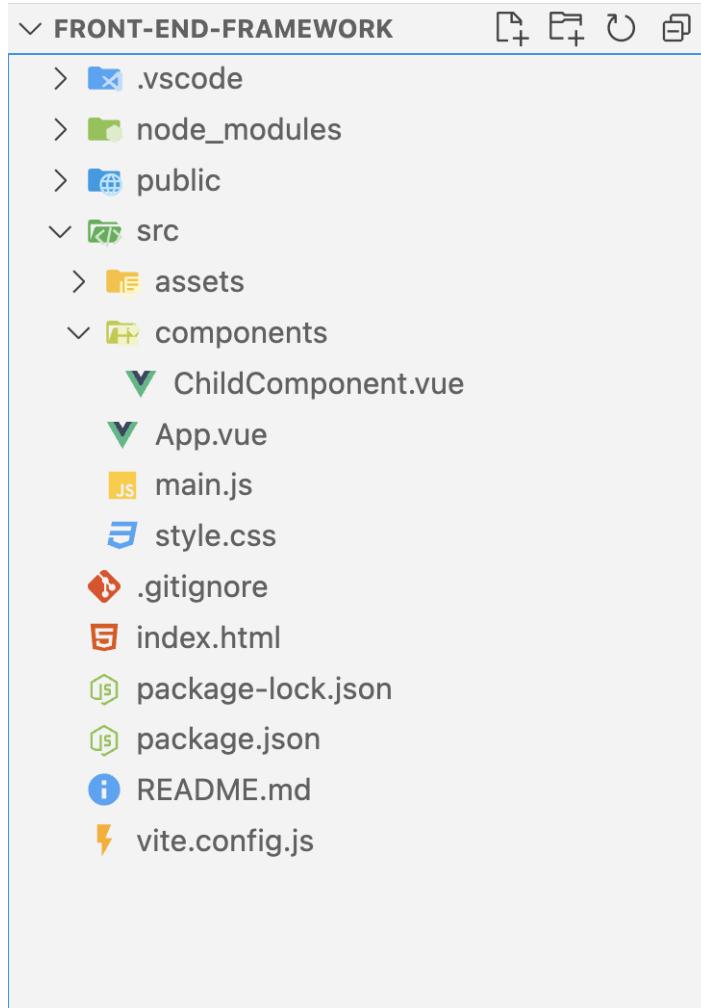
You clicked me 2 times.

You clicked me 3 times.

- ❑ **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



CÁCH SỬ DỤNG PROPS



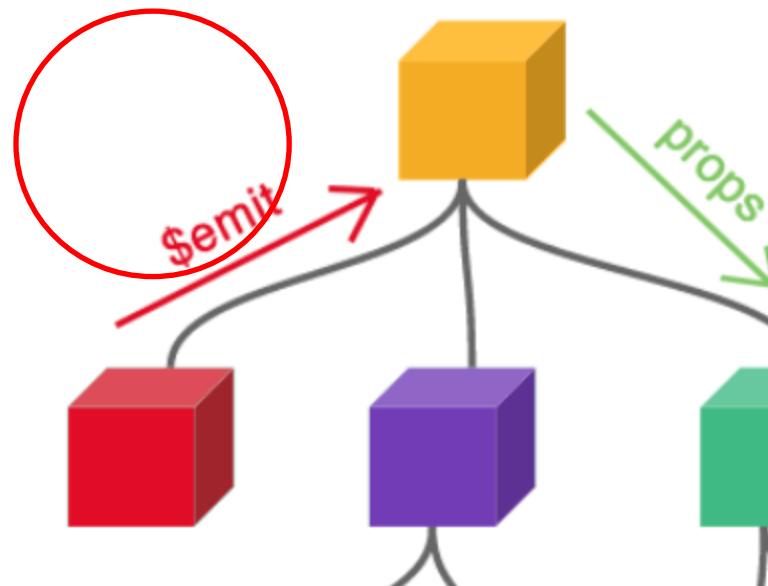
ChildComponent.vue

```
<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 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

FRONT-END-FRAMEWORK

```
> .vscode  
> node_modules  
> public  
> src  
> assets  
> components  
  > ChildComponent.vue  
  > App.vue  
  > main.js  
  > style.css  
> .gitignore  
> index.html  
> package-lock.json  
> package.json  
> README.md  
> vite.config.js
```

Lắng nghe sự kiện

```
<template>  
<div>  
<h2>Child Component</h2>  
<button @click="notifyParent">Click Me</button>  
</div>  
</template>  
<script setup>  
// 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>
```

ChildComponent.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>
```

App.vue



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.

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>
```

Kết quả:

Thông báo từ Component Con

Chưa có thông báo mới

Gửi thông báo

Click để nhận thông báo

Thông báo từ Component Con

Xin chào từ Component Con!

Gửi thông báo



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:

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

ChildComponent.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>
```

ParentComponent.vue

parent template

<FancyButton>

Click Me

repla

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>
```

parent template

<BaseLayout>

<template #header>



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:

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

ChildComponent.vue

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

ParentComponent.vue

<MyComponent> template

```
<div>
<slot
    :text="greetingMessage"
    :count="1"
/>
</div>
```

slot props

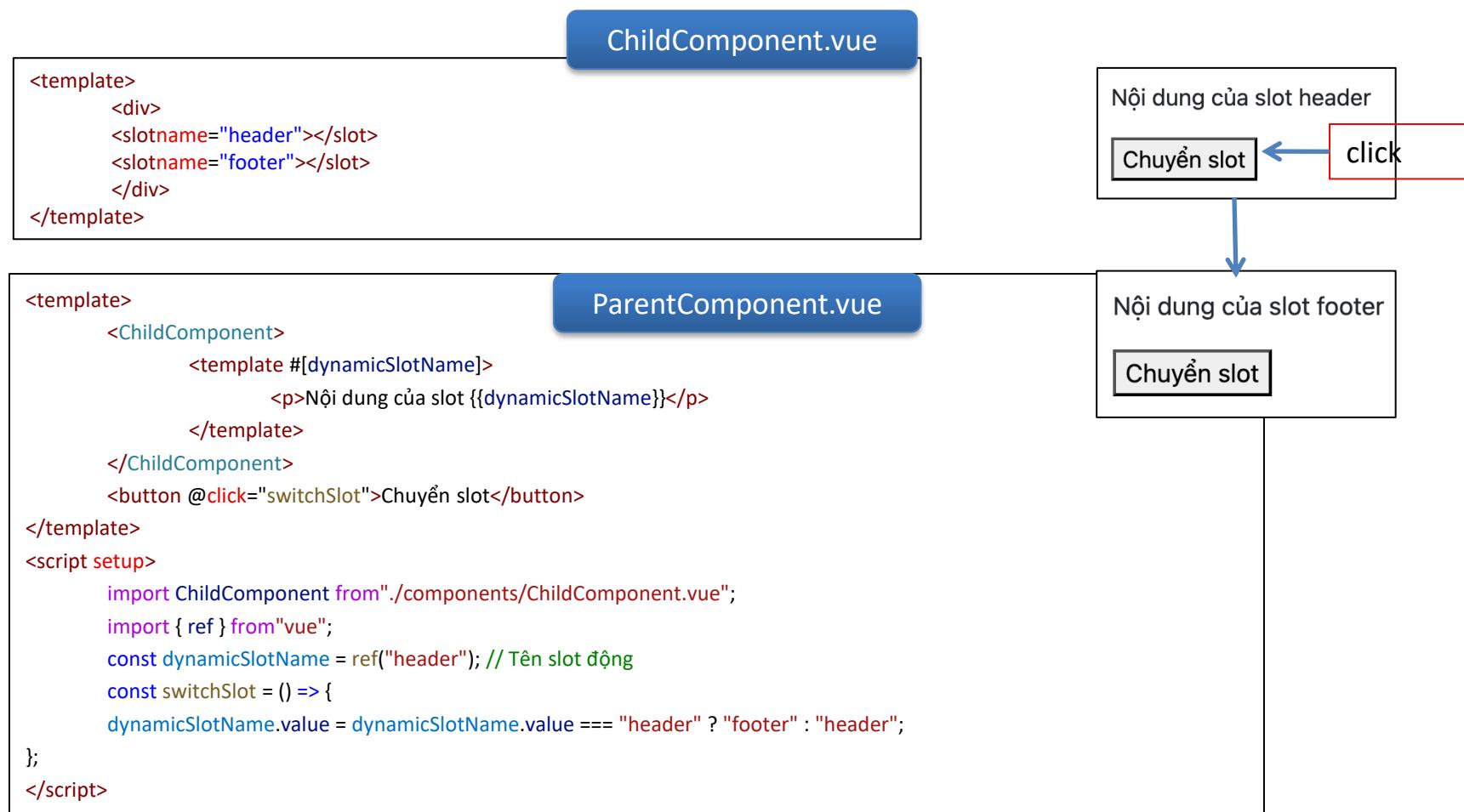
parent template

```
<MyComponent v-slot="slotProps">
    {{ slotProps.text }}
    {{ slotProps.count }}
</MyComponent>
```

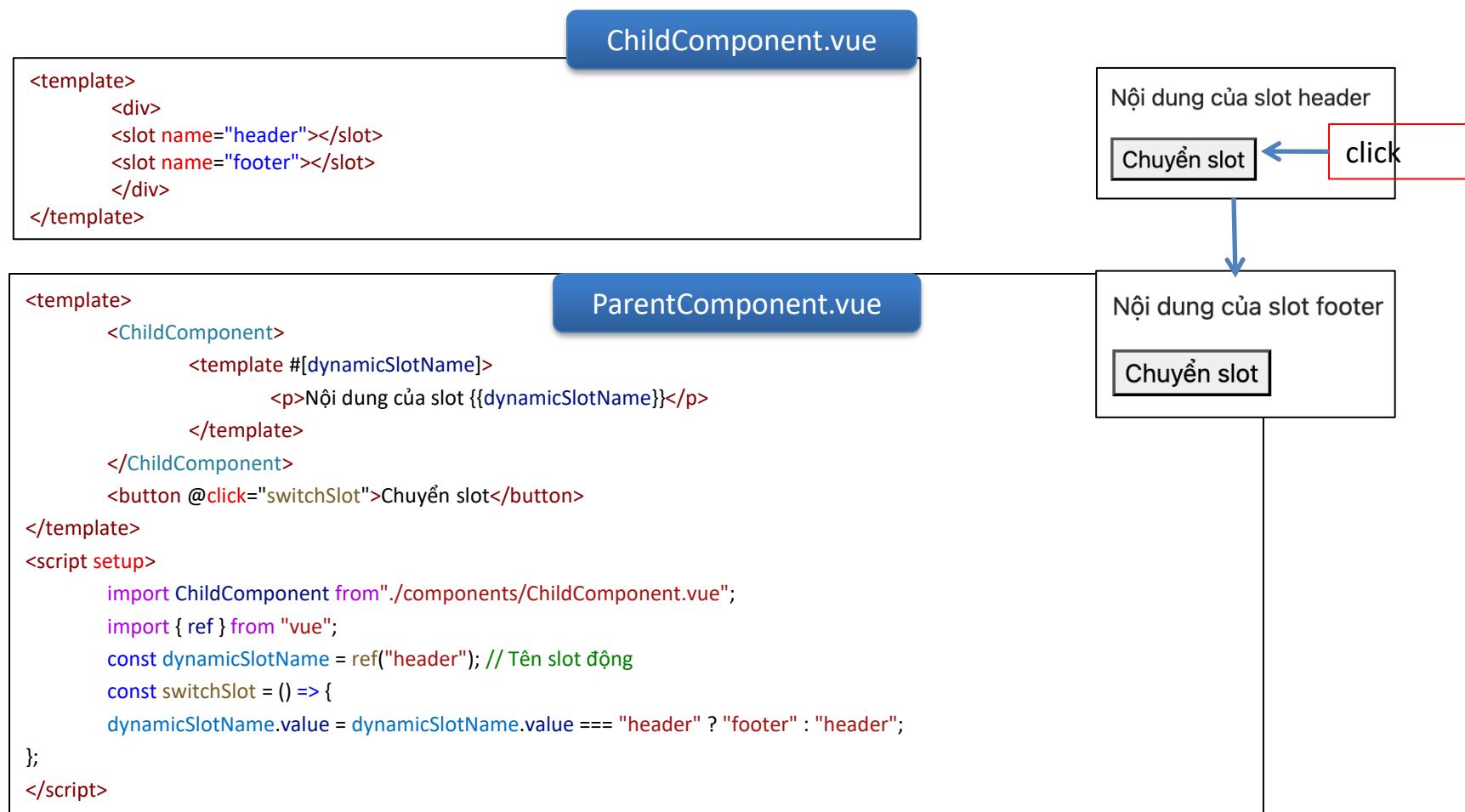
Kết quả

Đây là message từ ChildComponent

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

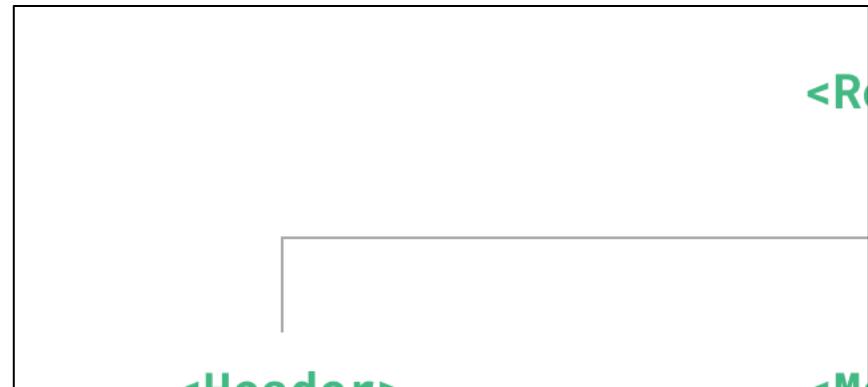


Dynamic Slot Names cho phép sử dụng tên slot thay đổi theo giá trị động, thay vì tên tĩnh như header hay footer.

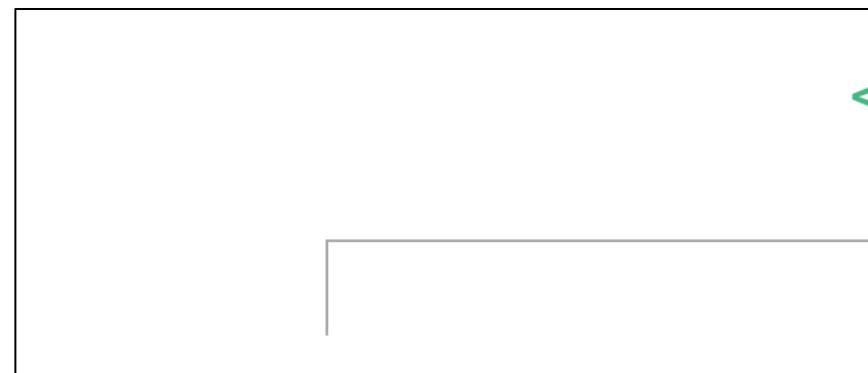


PROVIDE / INJECT

- ❑ 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** 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** 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.

☐ 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!

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!



Conceive Design Implement Operate



THỰC HỌC – THỰC NGHIỆP

FRONT-END FRAMEWORK

TỔNG QUAN VỀ VUE.JS

- Nắm được các khái niệm cơ bản về Vue.js
- Cài đặt môi trường phát triển Vue.js với thư viện Vitejs
- Giới thiệu về các cú pháp trong Vue.js
- Hiểu được cách sử dụng Bootstrap để tạo giao diện trong Vue



 Framework là gì?

 Giới thiệu về Framework VueJS

 Cài đặt môi trường phát triển

 Tạo và thực thi Project với Vue.Js

 Cài đặt và sử Bootstrap trong Vuejs

 Giới thiệu một vài cú pháp trong VueJS





PHẦN 1: TỔNG QUAN VỀ VUEJS

- Framework là bộ công cụ lập trình.
- Giúp tổ chức và quản lý mã nguồn tốt hơn
- Tiết kiệm thời gian và công sức khi phát triển website
- Giúp cải thiện tốc độ tải và phản hồi của ứng dụng.



- Vue** (phát âm /vju:/, như “view”) là một framework JavaScript
- Dùng để xây dựng các giao diện người dùng (UI).
- Cú pháp đơn giản và thân thiện với người mới bắt đầu.
- Sử dụng Virtual DOM để tối ưu hiệu năng ứng dụng



TẠI SAO DÙNG VUEJS?



HTML/CSS/JS
tách biệt

Reactive Data
Binding

Single-File
Components

Làm việc tốt với
Back-end

- ❑ **VueJS** được tạo bởi Evan You,
cựu nhân viên Google
- ❑ **VueJS** bắt đầu phát triển vào
năm 2013
- ❑ Ra mắt phiên bản đầu tiên chính
thức năm 2014
- ❑ Phiên bản mới nhất tính đến thời
điểm hiện tại (7/2024) là v3.4.33



VUE HOẠT ĐỘNG NHƯ NÀO?

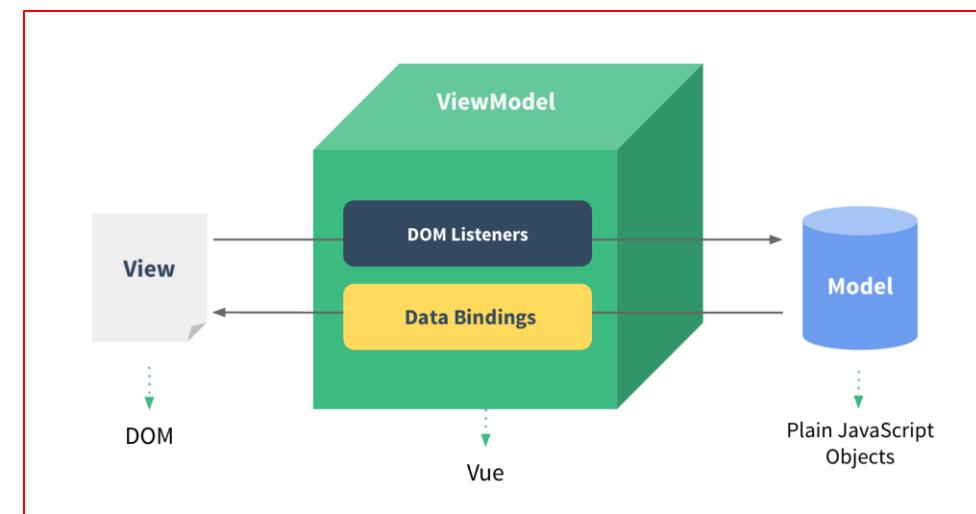
❑ Vue.js tự động đồng bộ dữ liệu giữa Model và View thông qua ViewModel, giúp mã nguồn dễ bảo trì và phát triển hơn. Mô hình này được gọi là **MVVM**

❑ **View:** Giao diện người dùng

❑ **ViewModel:**

- Data Bindings: Kết nối dữ liệu từ Model với View.
- DOM Listeners: Theo dõi sự kiện trên DOM và cập nhật Model.

❑ **Model:** Chứa dữ liệu.





Visual Studio Code

```
4. vue create hello-world (node)
Vue CLI v3.4.0
? Please pick a preset: Manually select features
? Check the features needed for your project:
  ● Babel
    ○ TypeScript
    ○ Progressive Web App (PWA) Support
    ○ Router
    ○ Vuex
    ○ CSS Pre-processors
    ● Linter / Formatter
    ○ Unit Testing
    ○ E2E Testing
```



- Truy cập: <https://nodejs.org>
- Tải và cài đặt

Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

[Download Node.js \(LTS\)](#)

Downloads Node.js v20.16.0¹ with long-term support.
Node.js can also be installed via package managers.

Want new features sooner? Get [Node.js v22.5.1¹](#) instead.



The screenshot shows the Node.js website's landing page. On the left, there's a green hexagonal background with the text "Run JavaScript Everywhere". Below it is a green button labeled "Download Node.js (LTS)". A red arrow points from this button towards the code editor on the right. On the right, there's a dark-themed code editor window with the following Node.js code:

```
1 // server.mjs
2 import { createServer } from 'node:http';
3
4 const server = createServer((req, res) => {
5   res.writeHead(200, { 'Content-Type': 'text/plain' });
6   res.end('Hello World!\n');
7 });
8
9 // starts a simple http server locally on port 3000
10 server.listen(3000, '127.0.0.1', () => {
11   console.log('Listening on 127.0.0.1:3000');
12 });
13
14 // run with `node server.mjs`
```

Below the code editor, there's a "JavaScript" label and a "Copy to clipboard" button.

Learn more what Node.js is able to offer with our Learning materials.

- ❑ Cách 1: CDN (Content delivery Network)
- ❑ Tạo file index.html và nhúng link sau ở đầu file

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

<div id="app">{{ message }}</div>

<script>
    const { createApp, ref } = Vue;

    createApp({
        setup() {
            const message = ref("Hello vue!");
            return {
                message,
            };
        },
    }).mount("#app");
</script>
```

➤ Xem ví dụ

Cách 2: Sử dụng NPM cài đặt Vitejs

Vitejs là công cụ để tạo 1 project
Vuejs cơ bản và có hiệu suất cao

- Do chính tác giả Vue tạo ra
- Sử dụng **npm** để tải **Vitejs**
- Sử dụng Terminal (hoặc Command Prompt):



```
npm create vue@latest front-end-framework
```

CÀI ĐẶT MÔI TRƯỜNG VUEJS

- Lệnh này sẽ cài đặt và thực thi **create-vue**, công cụ tạo dự án chính thức của **Vue**.
- Bạn sẽ được yêu cầu chọn các tính năng tùy chọn như hình sau.

- ✓ Add TypeScript? ... No / Yes
- ✓ Add JSX Support? ... No / Yes
- ✓ Add Vue Router for Single Page Application development? ... No / Yes
- ✓ Add Pinia for state management? ... No / Yes
- ✓ Add Vitest for Unit testing? ... No / Yes
- ✓ Add an End-to-End Testing Solution? ... No / Cypress / Nightwatch / Playwright
- ✓ Add ESLint for code quality? ... No / Yes
- ✓ Add Prettier for code formatting? ... No / Yes
- ✓ Add Vue DevTools 7 extension for debugging? (experimental) ... No / Yes

Scaffolding project in ./<front-end-framework>...
Done.

CÀI ĐẶT MÔI TRƯỜNG VUEJS

☐ Sử dụng Terminal VSC:

```
cd front-end-framework  
npm install  
npm run dev
```

Truy cập thư mục

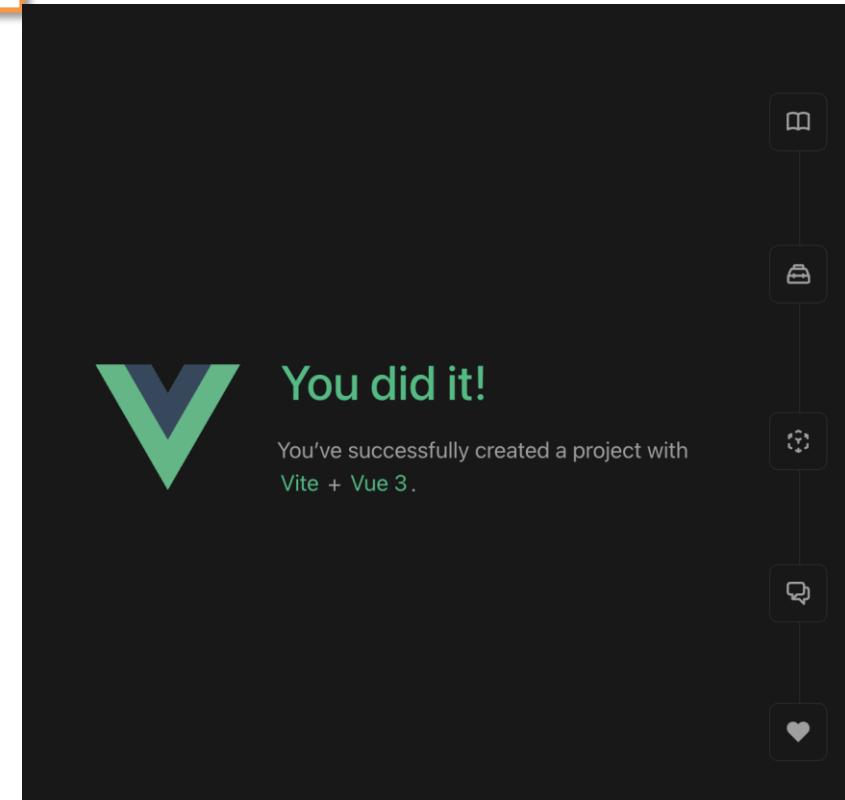
Cài đặt các thư viện

Chạy ứng dụng Vue

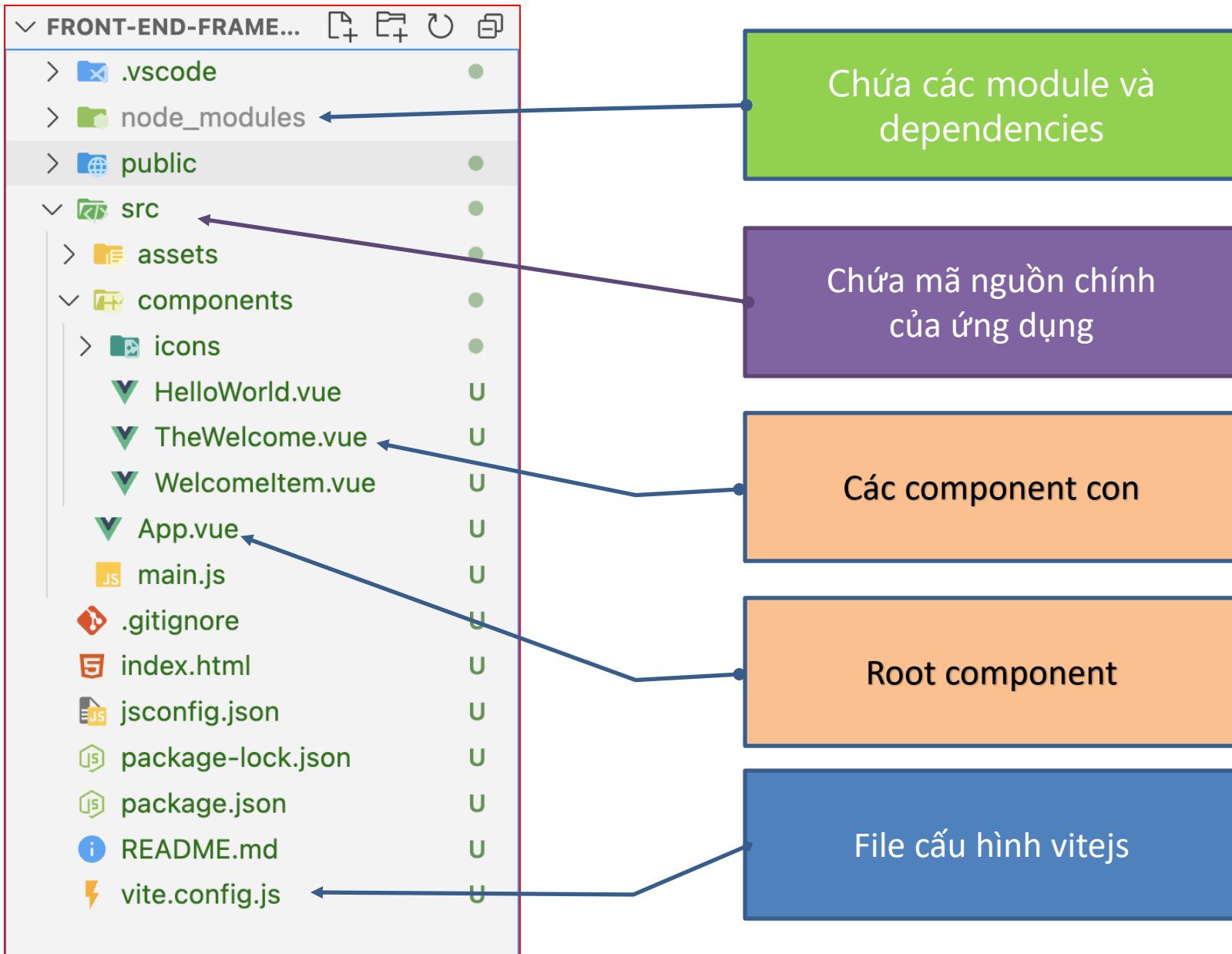
☐ Bật trình duyệt

VITE v5.3.5 ready in 659 ms

- Local: <http://localhost:5173/>
- Network: use **--host** to expose
- press **h + enter** to show help



TỔ CHỨC CODE TRONG VUEJS



- ❑ Mọi ứng dụng Vue đều bắt đầu bằng việc tạo một application instance mới với hàm **createApp**:

```
import { createApp } from "vue";
const app = createApp({
    /* Các tùy chọn của thành phần gốc */
});
```

- ❑ Thành phần gốc(root component) là thành phần đầu tiên trong một ứng dụng Vue.
- ❑ Chứa các thành phần con và là điểm khởi đầu của ứng dụng.
- ❑ Được truyền vào hàm **createApp** để tạo ra ứng dụng.
- ❑ Ví dụ: App.vue thường là thành phần gốc.

```
// main.js là file đầu tiên được chạy khi ứng dụng Vue.js được khởi tạo.
import { createApp } from "vue";
// Nhập thành phần gốc App từ một single-file component
import App from "./App.vue";
createApp(App).mount("#app");
```

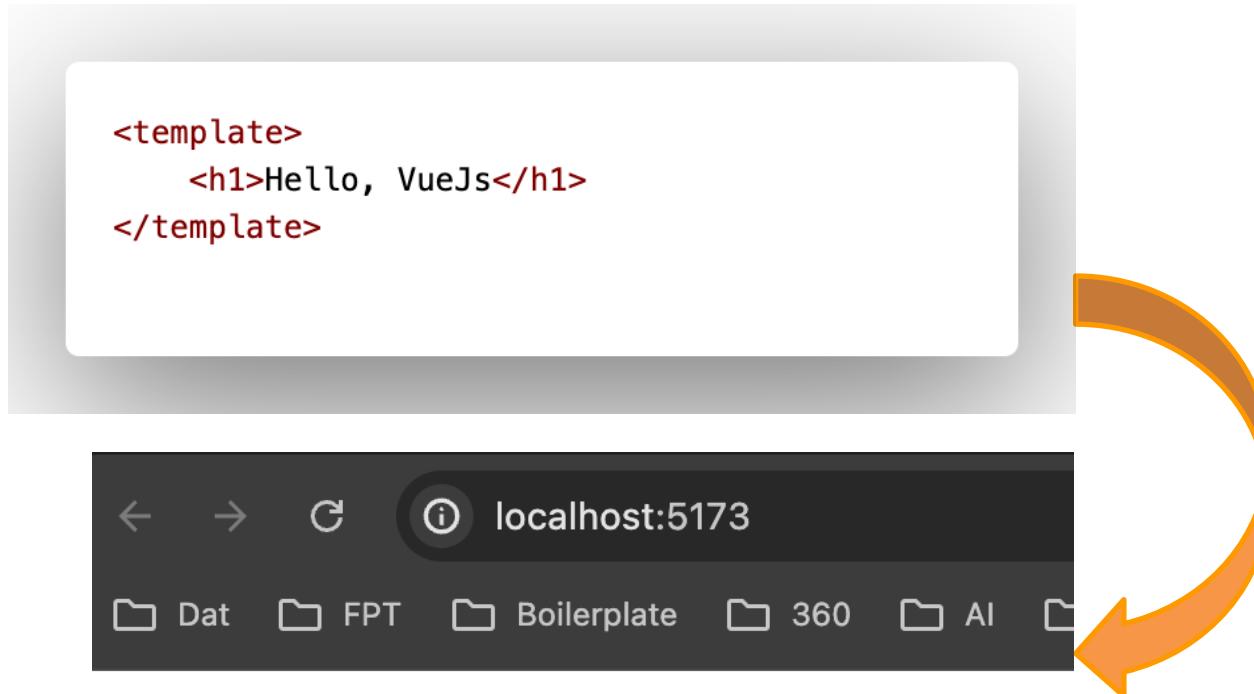
- **<script setup>**: Đơn giản hóa khai báo và sử dụng dữ liệu trong Vue 3.
- **<template>**: Định nghĩa cấu trúc giao diện của thành phần bằng HTML.
- **<style scoped>**: Áp dụng CSS chỉ cho thành phần hiện tại.

```
<script setup>
// Code js của bạn ở đây
</script>

<template>
    <!-- Code html của bạn ở đây -->
</template>

<style scoped>
/* Code css của bạn ở đây */
</style>
```

- ☐ Truy cập file **App.vue** cập nhật code sau và lưu lại.



Hello, VueJS!

➤ [Xem ví dụ](#)



demo



PHẦN 2: STYLE VÀ TEMPLATE SYNTAX

Sử dụng framework css **Bootstrap** ta thực hiện 2 bước

- Cài đặt Bootstrap: **npm i bootstrap**
- Để nhúng thư viện bootstrapp vào dự án vue:
Mở tệp **main.js** và thêm các dòng sau:

```
import { createApp } from "vue";
import App from "./App.vue";

import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap/dist/js/bootstrap.bundle.min.js";

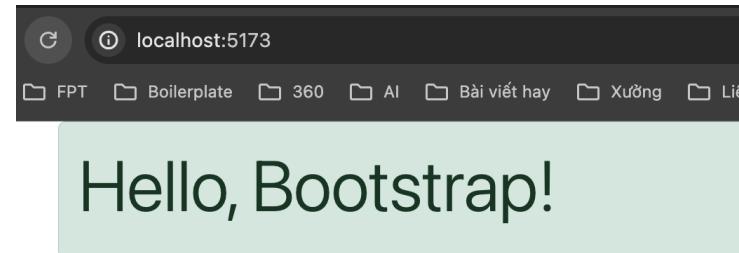
createApp(App).mount("#app");
```

- [Xem ví dụ](#)

Kiểm tra bootstrap có hoạt động chưa?

Mở file **App.vue** trong thư mục app và cập nhật code như sau:

```
<template>
  <div id="app">
    <div class="container">
      <div class="alert alert-success">
        <h1 class="display-4">Hello, Bootstrap!</h1>
      </div>
    </div>
  </div>
</template>
```



➤ Xem ví dụ

CÚ PHÁP TEMPLATE TRONG VUE

- ❑ **Vue** sử dụng cú pháp mẫu dựa trên HTML cho phép bạn liên kết một cách khai báo giữa DOM được render với dữ liệu của của đối tượng Vue bên dưới
- ❑ Hình thức ràng buộc dữ liệu cơ bản nhất là text interpolation
- ❑ Cú pháp “mustache” với hai dấu ngoặc. :
`Message: {{ msg }}`

- **<script setup>**: Đơn giản hóa khai báo và sử dụng dữ liệu trong Vue 3.
- **<template>**: Định nghĩa cấu trúc giao diện của thành phần bằng HTML.
- **<style scoped>**: Áp dụng CSS chỉ cho thành phần hiện tại.

```
<script setup>
// Code js của bạn ở đây
</script>

<template>
    <!-- Phần này là code UI của bạn -->
</template>

<style scoped>
/* Code css của bạn ở đây */
</style>
```

```
<script setup>
const courseName = "Framework Vue 3";
const courseLevel = "Nâng cao";
const courseTime = 30; // giờ
const coursesActive = true;
</script>
<template>
<div class="container my-5">
    <h1 class="display-4 mb-3">Thông tin:</h1>
    <ul class="list-group">
        <li class="list-group-item">
            Tên khóa học: <strong>{{courseName}}</strong>
        </li>
        <li class="list-group-item">Cấp độ: {{courseLevel}}</li>
        <li class="list-group-item">Thời gian: {{courseTime}} giờ</li>
        <li class="list-group-item">
            Trạng thái: {{coursesActive ? "Đang mở" : "Đã đóng"}}
        </li>
    </ul>
</div>
</template>
```

Dữ liệu kiểu chuỗi

Thông tin khóa học:

Tên khóa học: Framework Vue 3

Cấp độ: Nâng cao

Thời gian: 30 giờ

Trạng thái: Đang mở

➤ [Xem Ví dụ](#)



demo

Ngoài ra còn rất nhiều cú pháp khác như:

- ❖ Attribute Binding: **v-bind**
- ❖ Conditional Rendering: **v-if, v-else, v-else-if**
- ❖ List Rendering: **v-for**
- ❖ Two-way Binding: **v-model**
- ❖ Event Handling: **v-on**
- ❖ Conditional Display: **v-show**
- ❖ Class Binding: **v-bind:class**
- ❖ Style Binding: **v-bind:style**

Chúng ta sẽ vào chi tiết ở các bài tiếp theo

- Framework là gì?
- Giới thiệu về Framework VueJS
- Môi trường phát triển
- Cài đặt
- Tạo và thực thi Project với Vue.Js
- Kiến trúc tổ chức của Vue.Js
- Cài đặt và sử Bootstrap trong Vuejs
- Giới thiệu một vài cú pháp trong VueJS



thank
you!