



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?

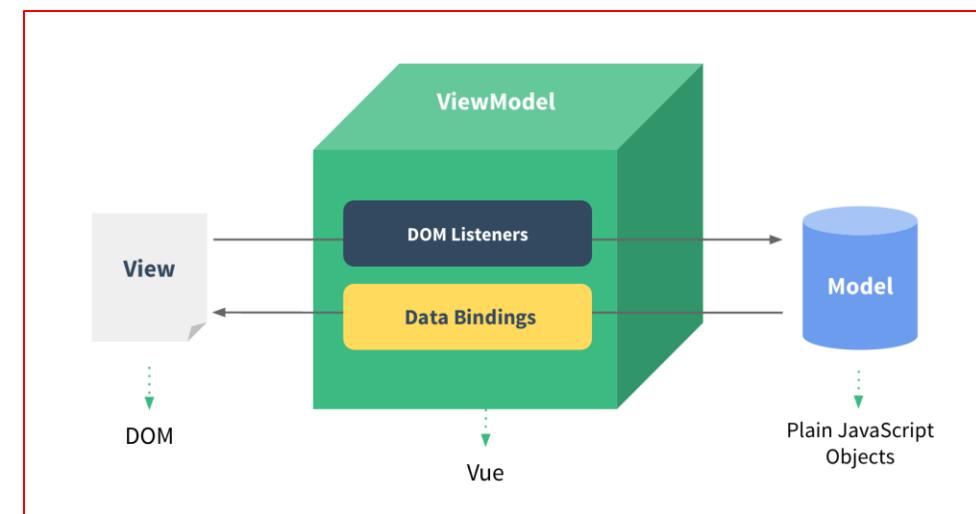
❑ Vue.js tự động đồng bộ dữ liệu giữa Model và View thông qua ViewModel, giúp mã nguồn dễ bảo trì và phát triển hơn. Mô hình này được gọi là **MVVM**

❑ **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 official Node.js website. On the left, there's a green hexagonal background with the text "Run JavaScript Everywhere". Below it is a 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 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

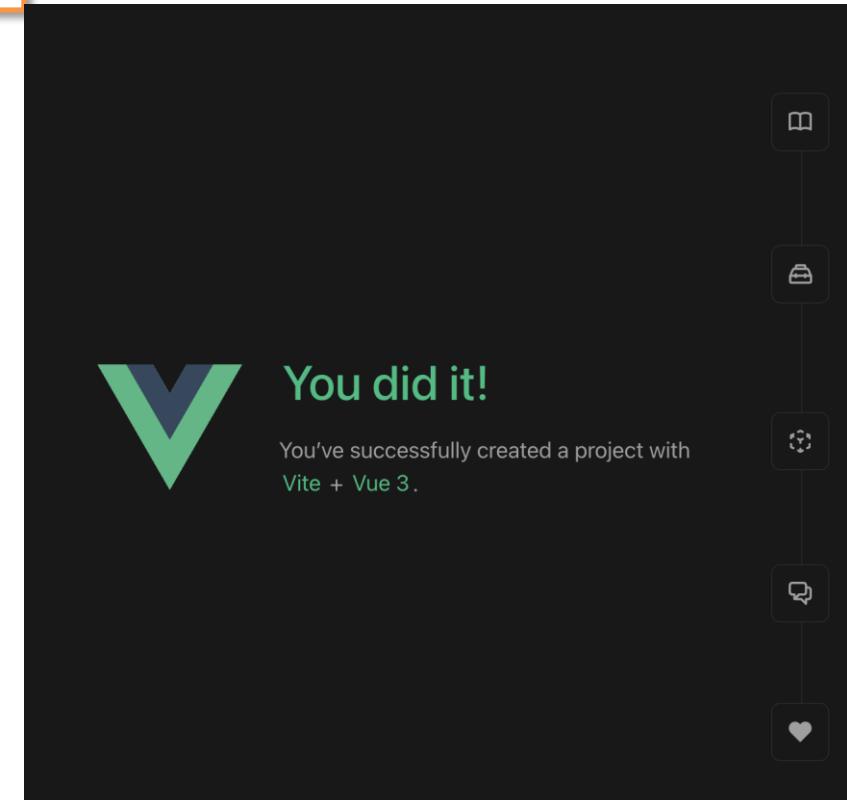
Cài đặt các thư viện

Chạy ứng dụng Vue

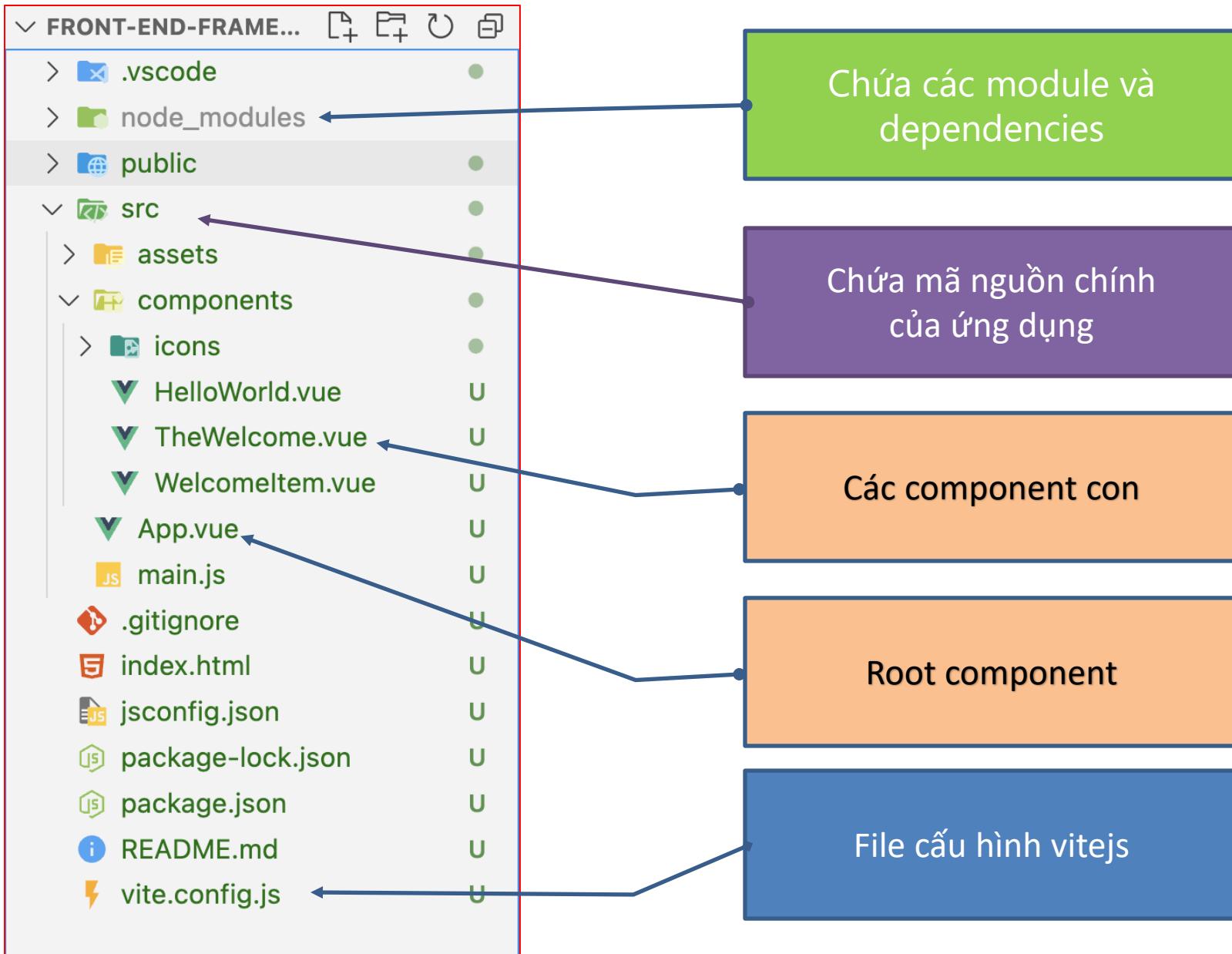
☐ Bật trình duyệt

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");
```

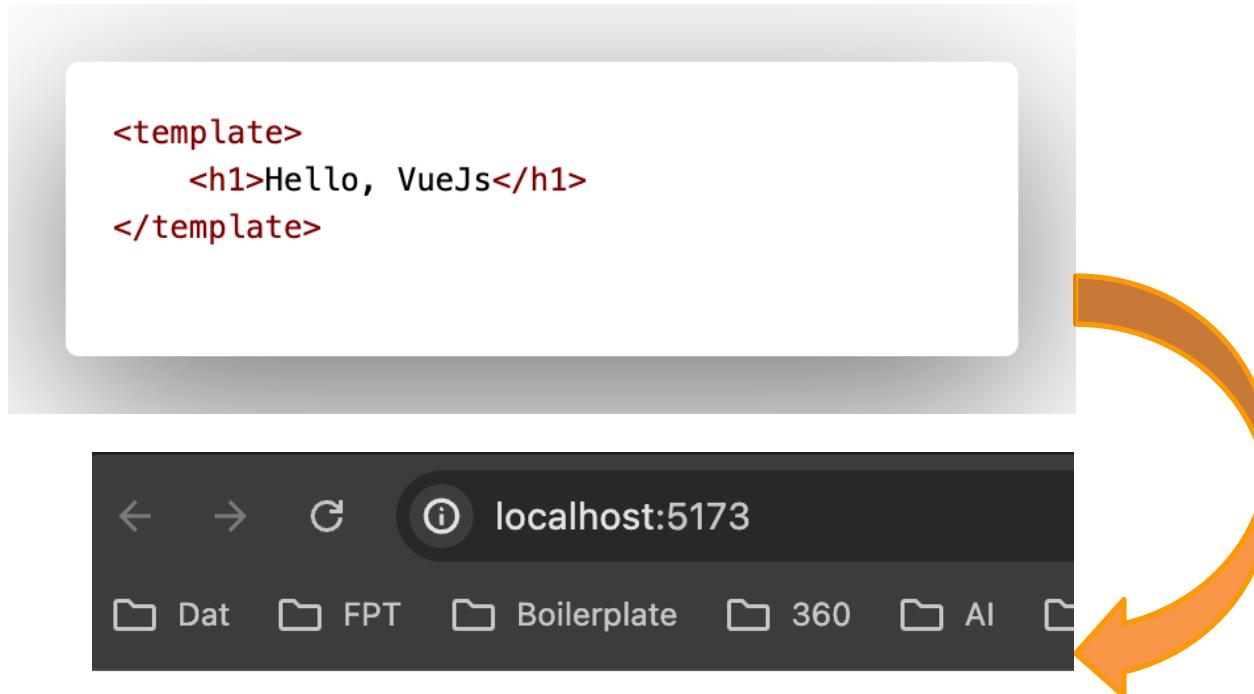
- **<script setup>**: Đơn giản hóa khai báo và sử dụng dữ liệu trong Vue 3.
- **<template>**: Định nghĩa cấu trúc giao diện của thành phần bằng HTML.
- **<style scoped>**: Áp dụng CSS chỉ cho thành phần hiện tại.

```
<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";

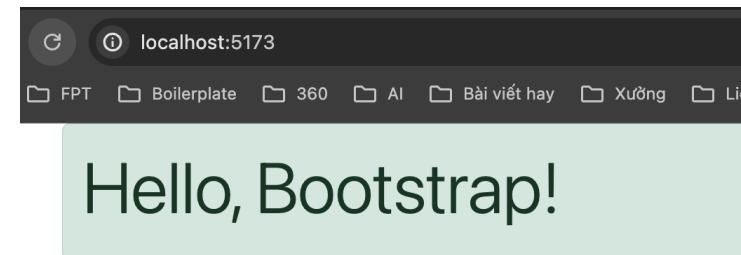
createApp(App).mount("#app");
```

- [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

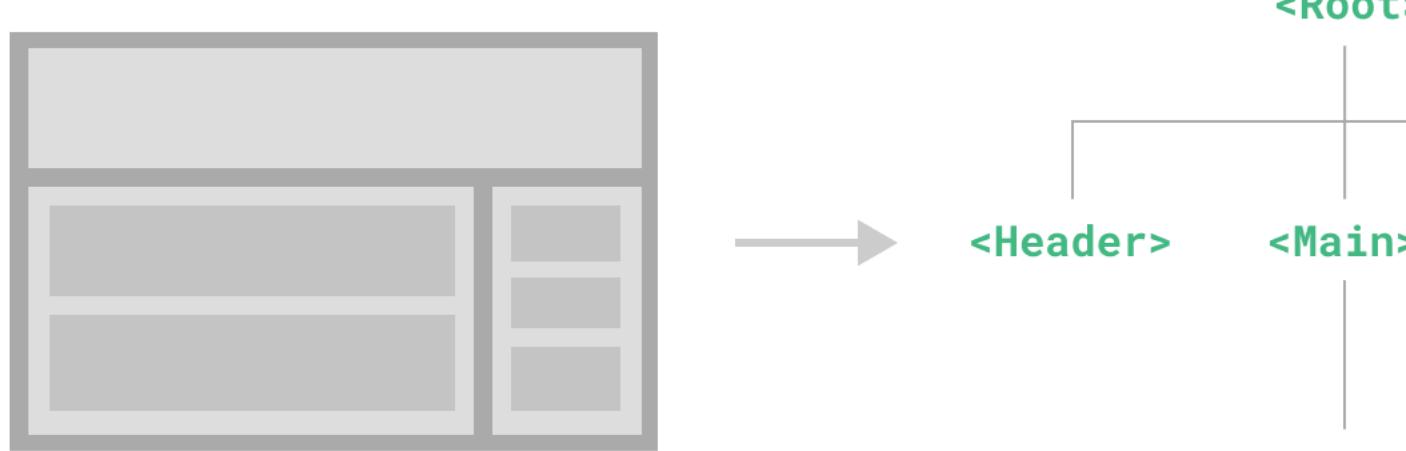




PHẦN 1

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>
```

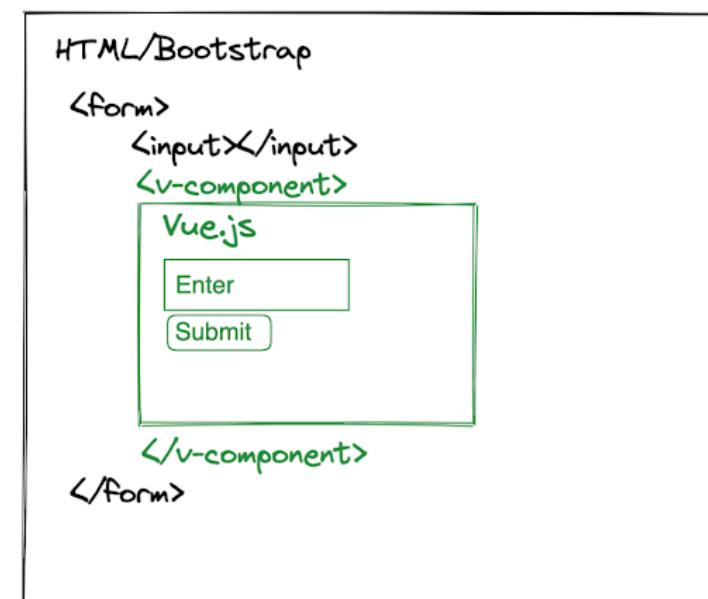
3. Style (CSS)

```
<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.

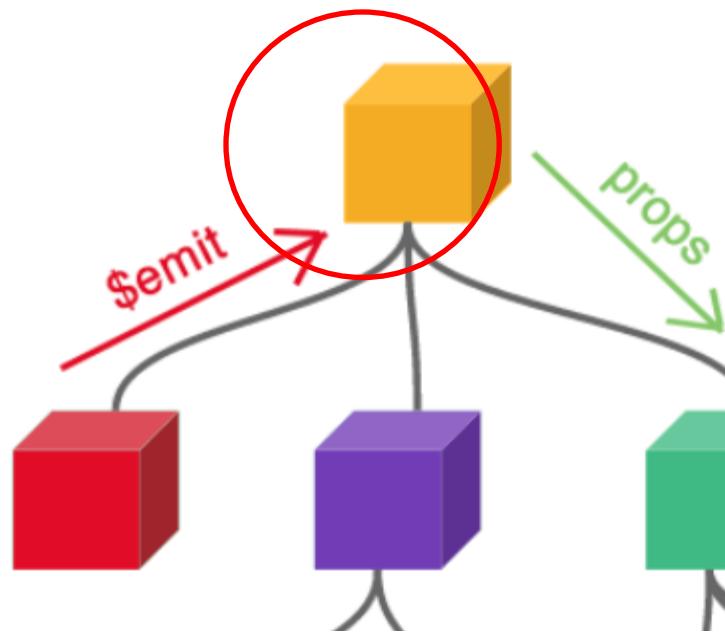
You clicked me 0 times.

click

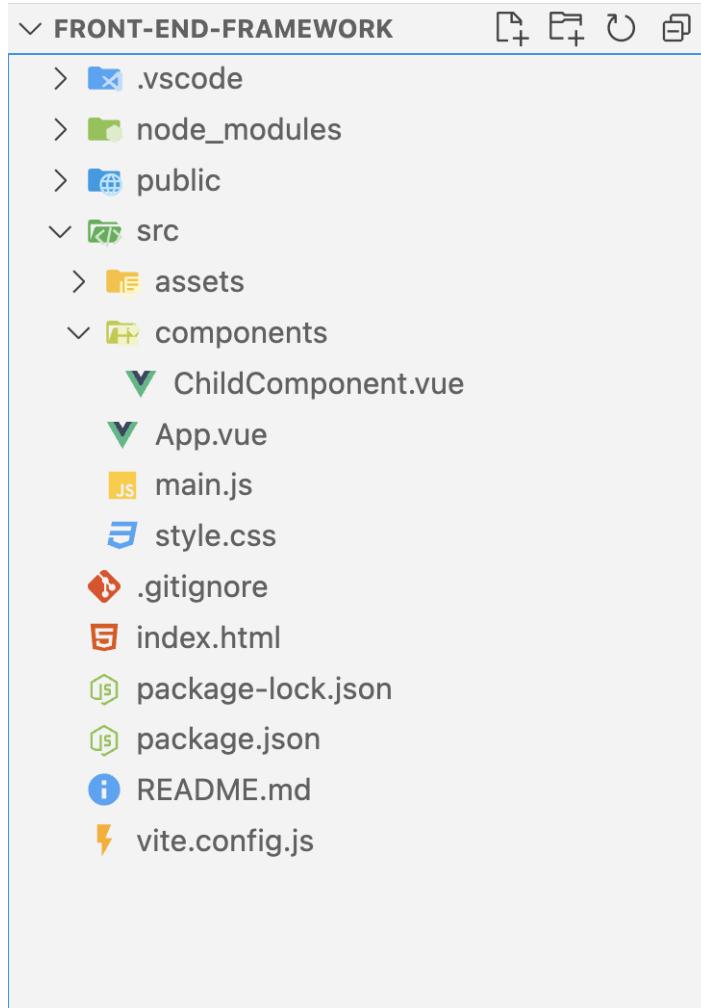
You clicked me 2 times.

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



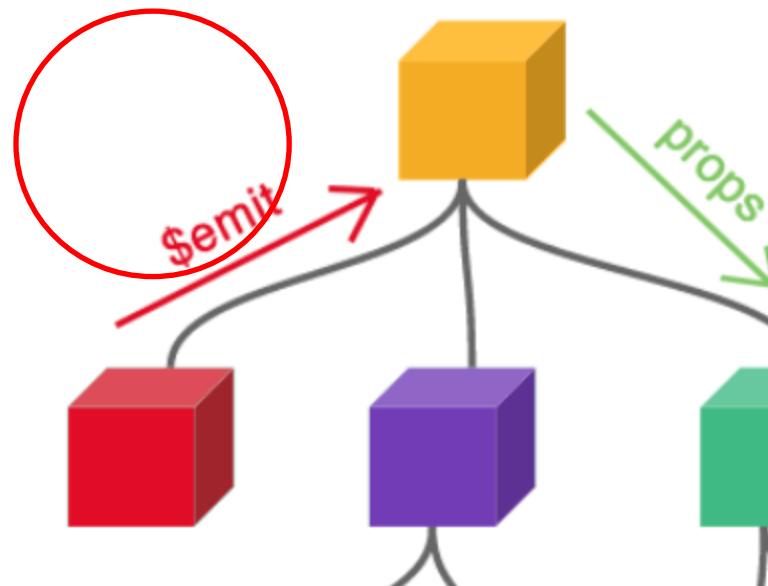
ChildComponent.vue

```
<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

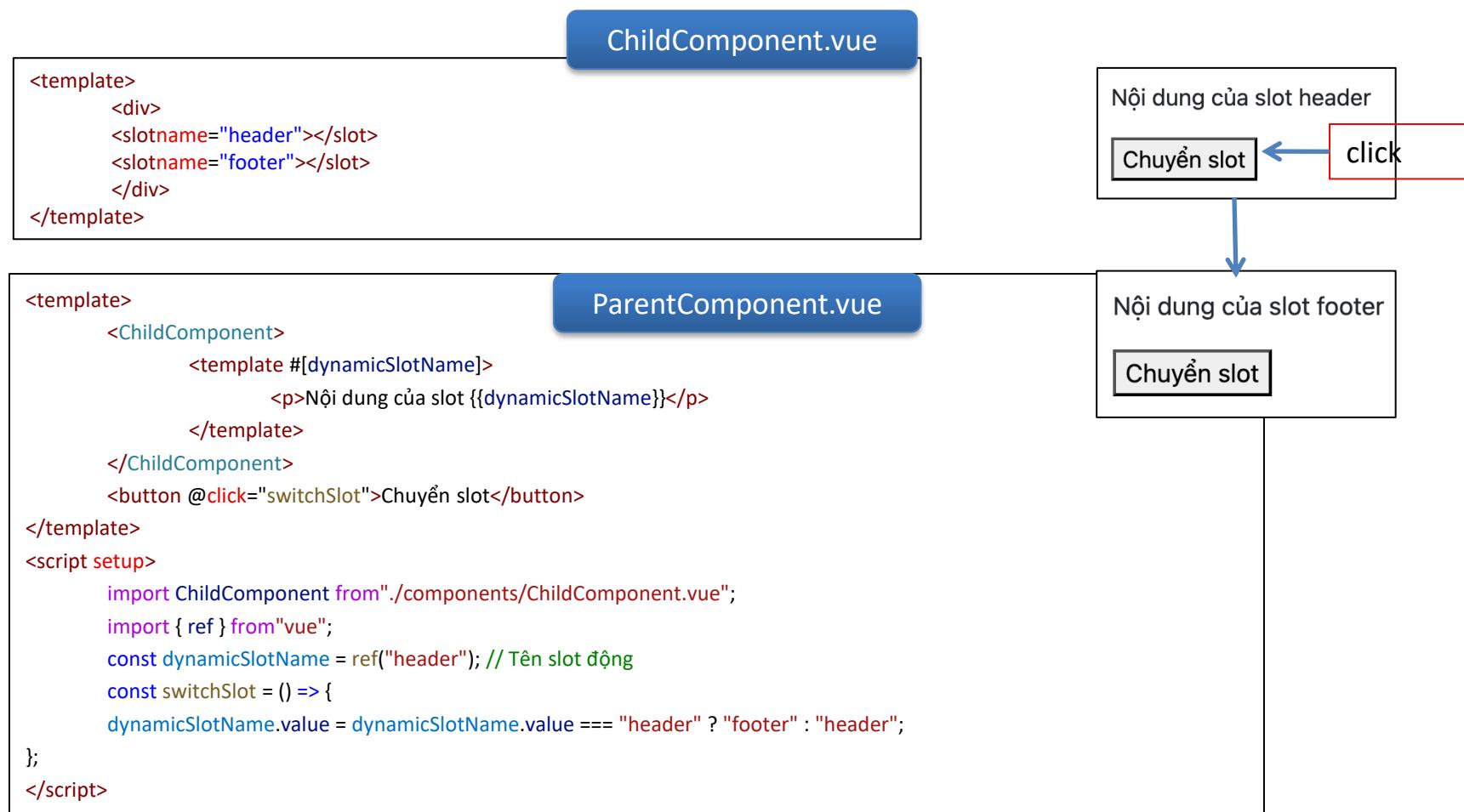
parent template

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

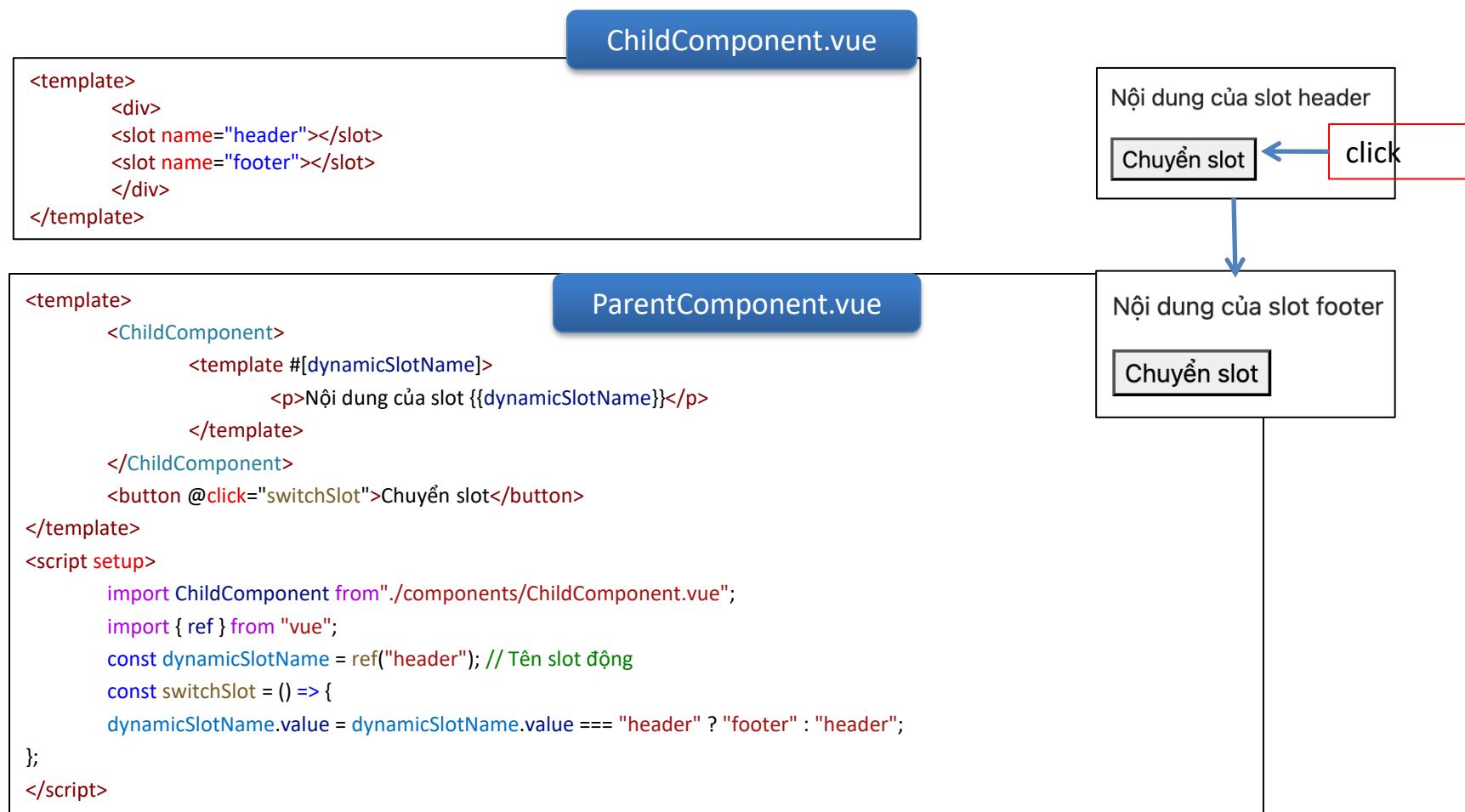
Kết quả

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.

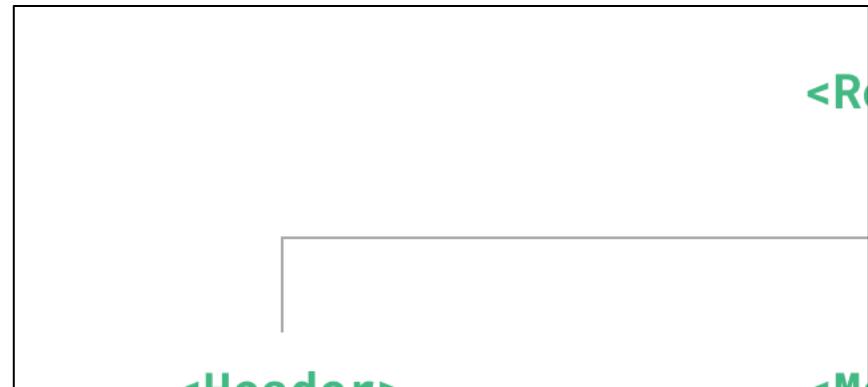


Dynamic Slot Names cho phép sử dụng tên slot thay đổi theo giá trị động, thay vì tên tĩnh như header hay footer.

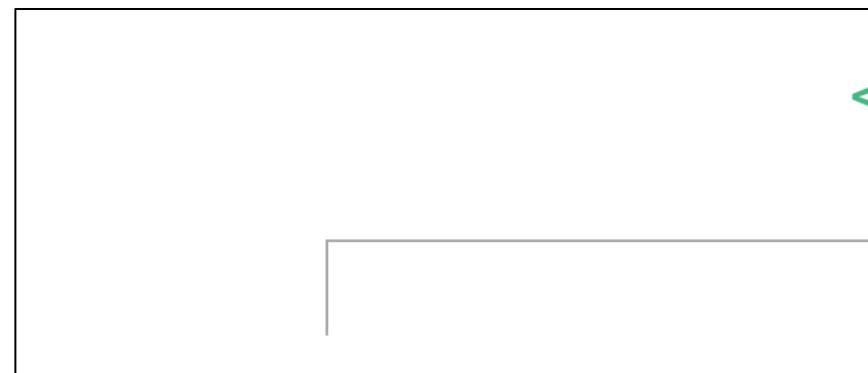


PROVIDE / INJECT

- ❑ 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!



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

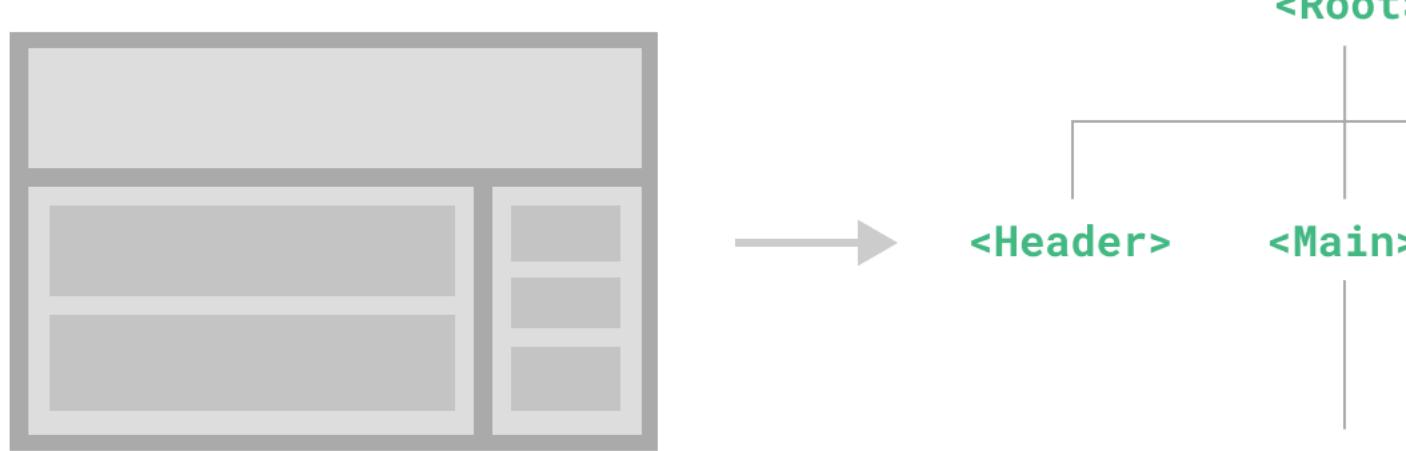




PHẦN 1

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>
```

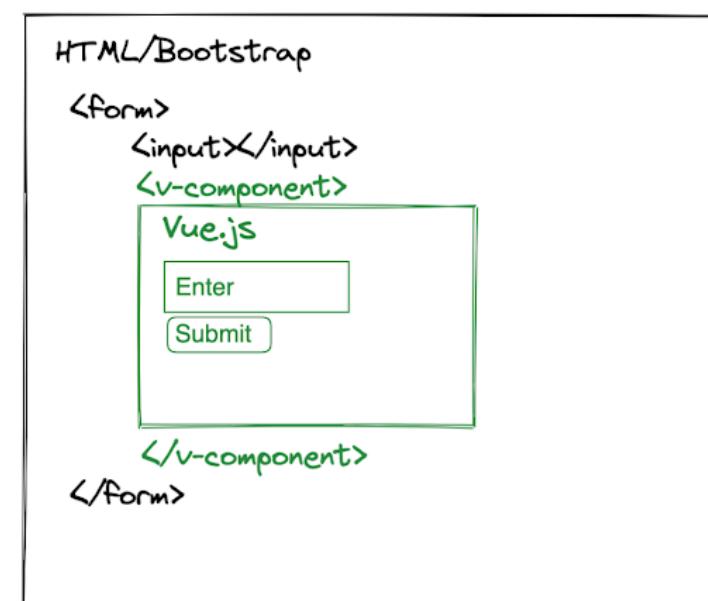
3. Style (CSS)

```
<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.

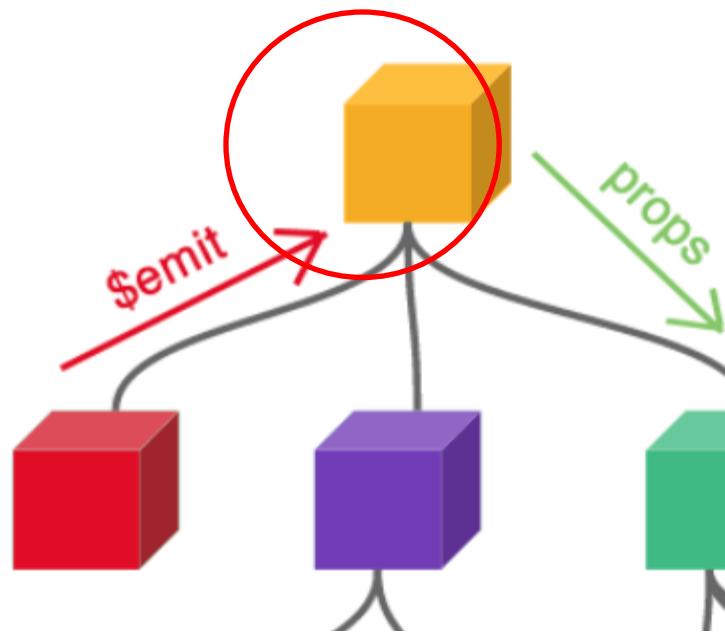
You clicked me 0 times.

click

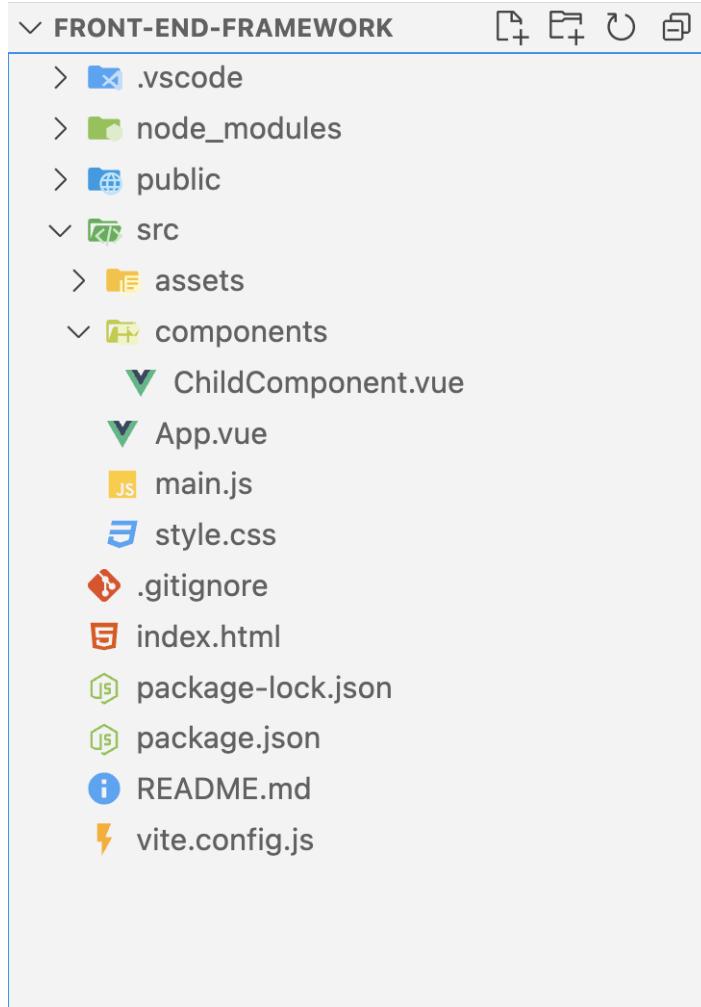
You clicked me 2 times.

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



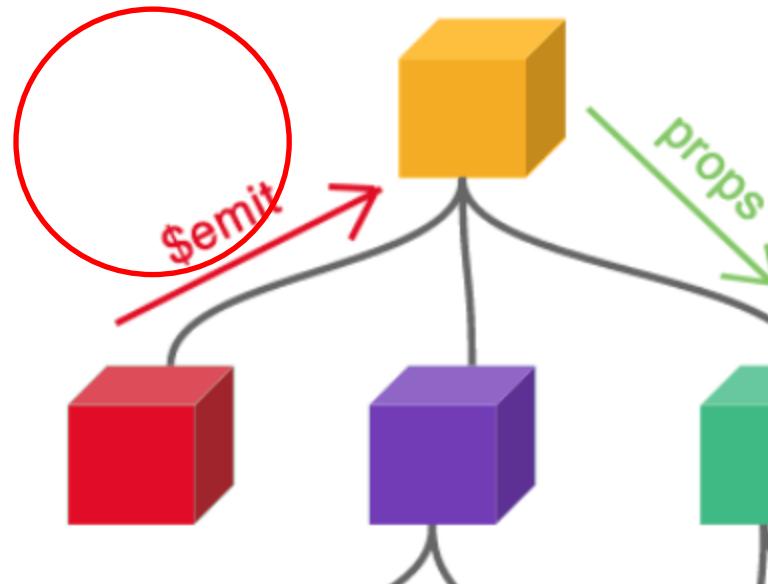
ChildComponent.vue

```
<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 --&gt;
&lt;ChildComponent @childEvent="handleChildEvent" /&gt;
<!-- Hiển thị giá trị từ ref --&gt;
&lt;p&gt;Message from child: {{messageFromChild}}&lt;/p&gt;
&lt;/template&gt;</pre>
```

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

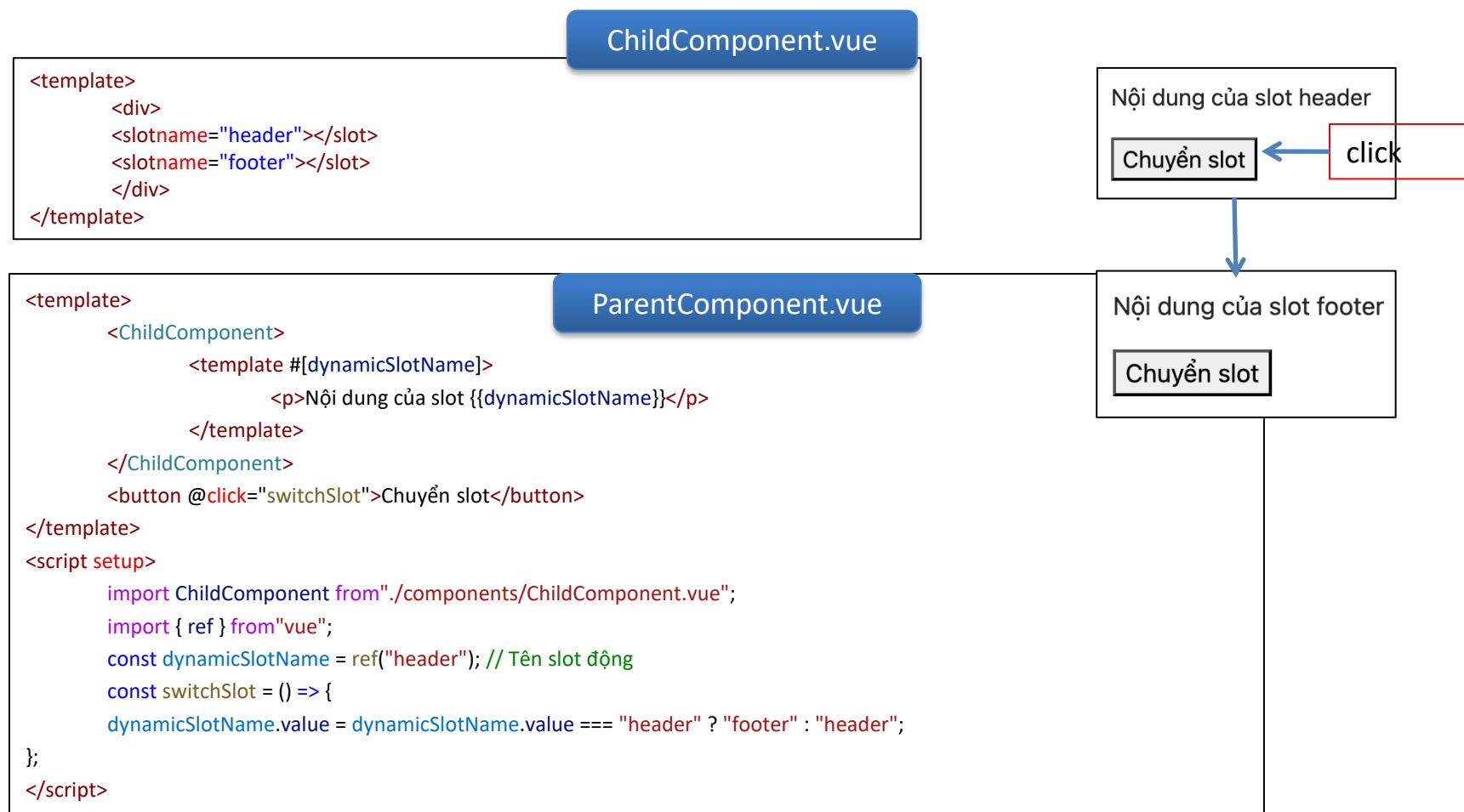
parent template

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

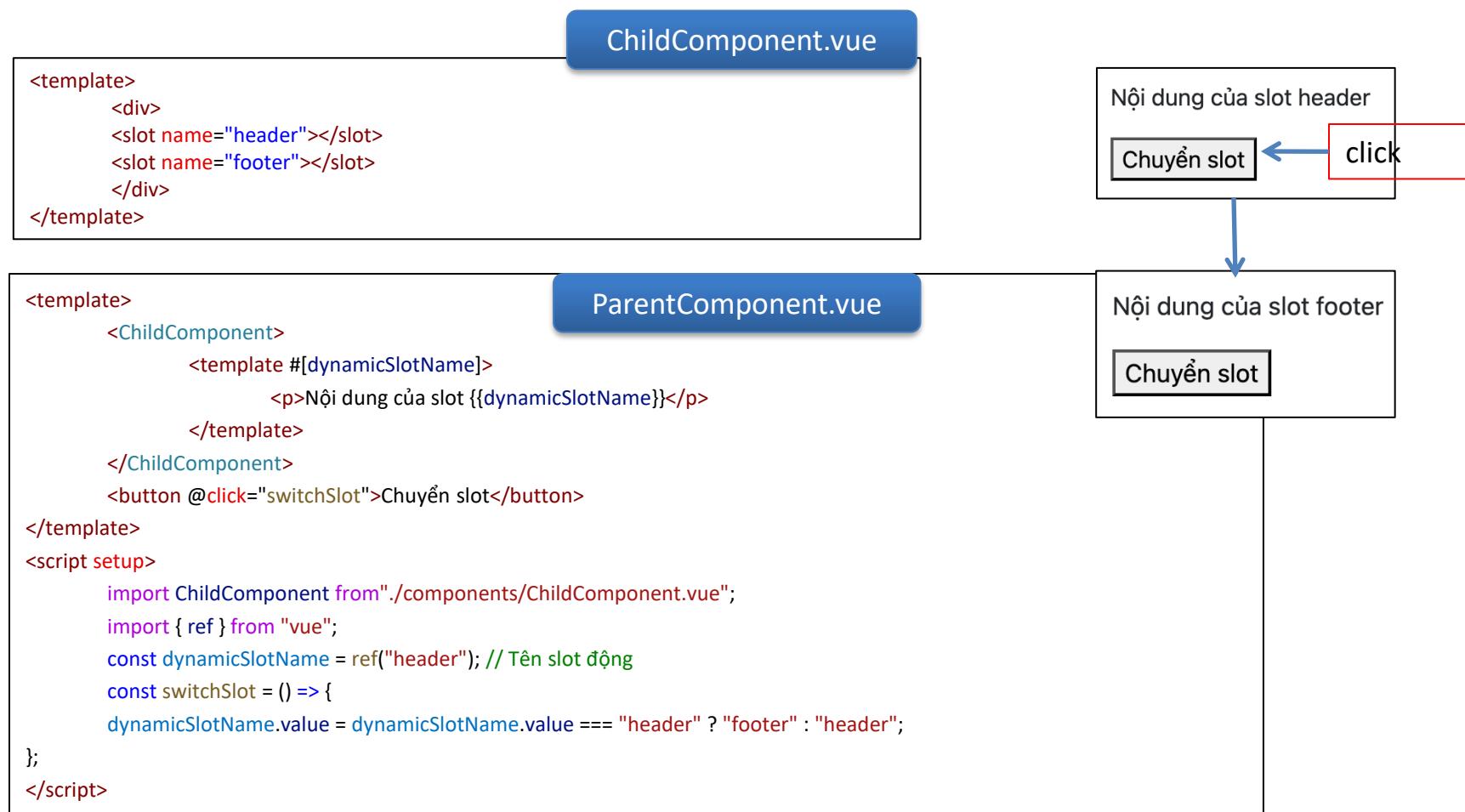
Kết quả

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.

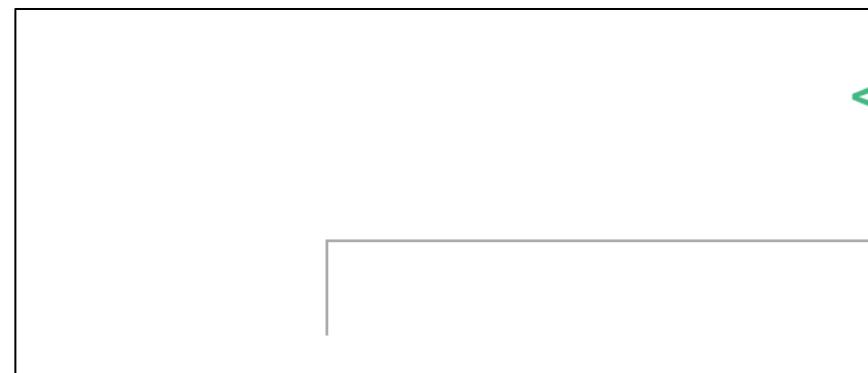


PROVIDE / INJECT

- ❑ 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?

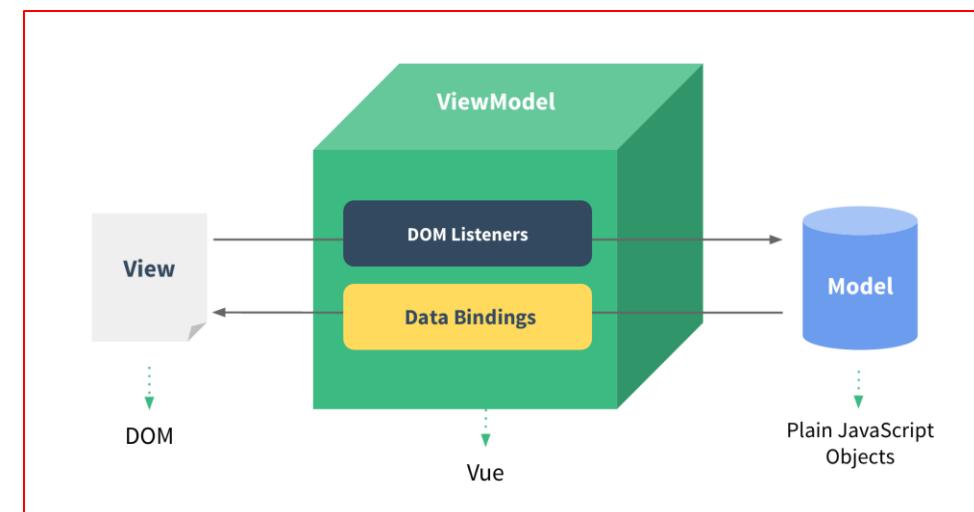
❑ Vue.js tự động đồng bộ dữ liệu giữa Model và View thông qua ViewModel, giúp mã nguồn dễ bảo trì và phát triển hơn. Mô hình này được gọi là **MVVM**

❑ **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
  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" 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

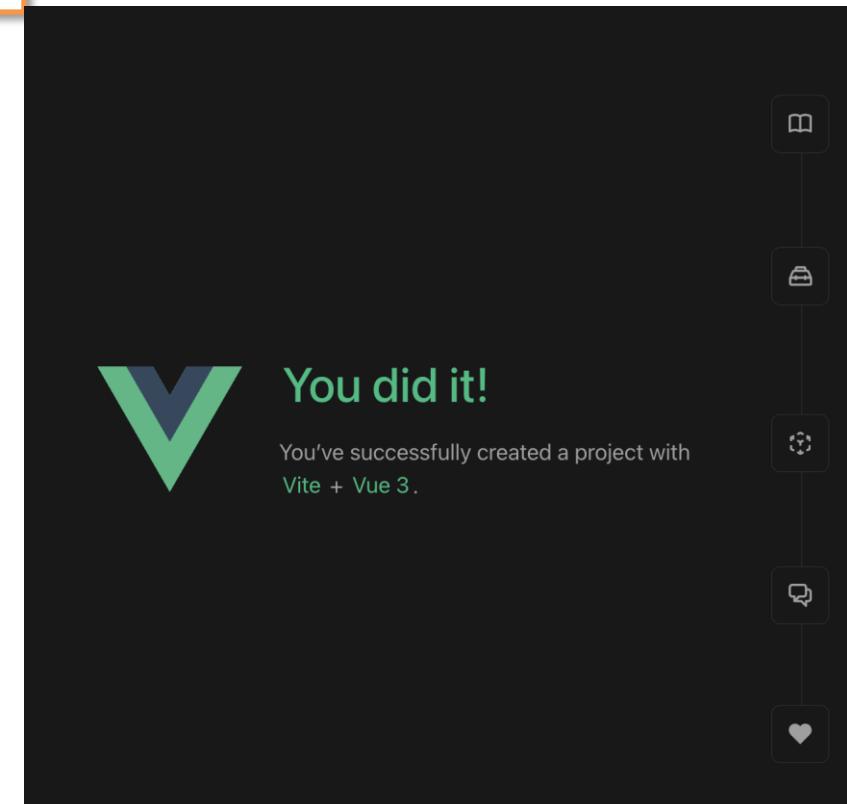
Cài đặt các thư viện

Chạy ứng dụng Vue

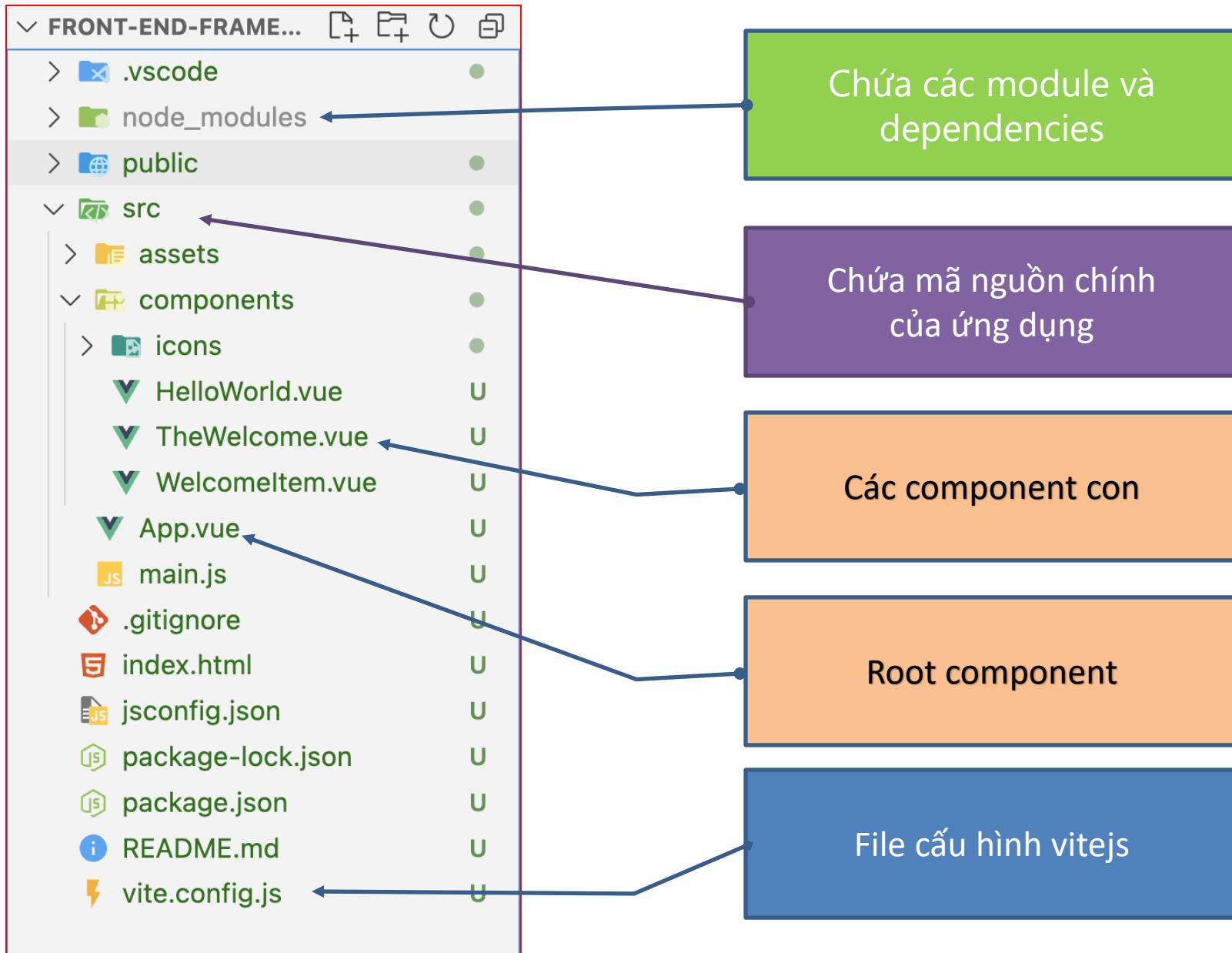
☐ Bật trình duyệt

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");
```

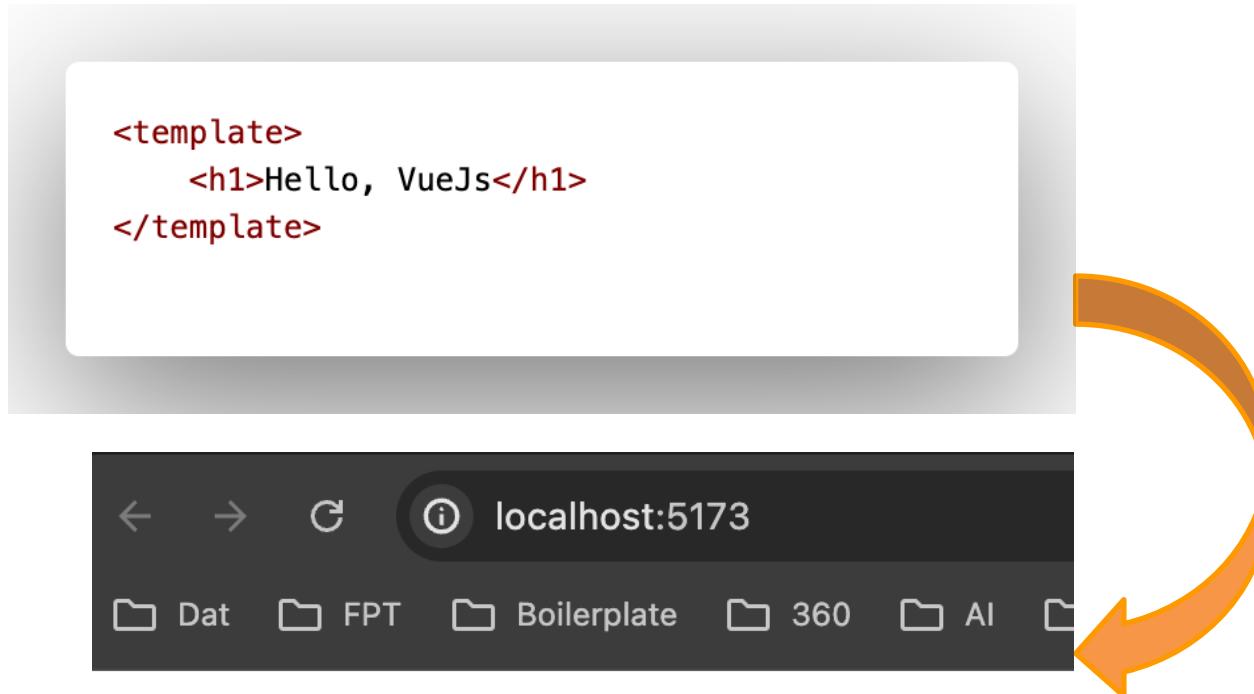
- **<script setup>**: Đơn giản hóa khai báo và sử dụng dữ liệu trong Vue 3.
- **<template>**: Định nghĩa cấu trúc giao diện của thành phần bằng HTML.
- **<style scoped>**: Áp dụng CSS chỉ cho thành phần hiện tại.

```
<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";

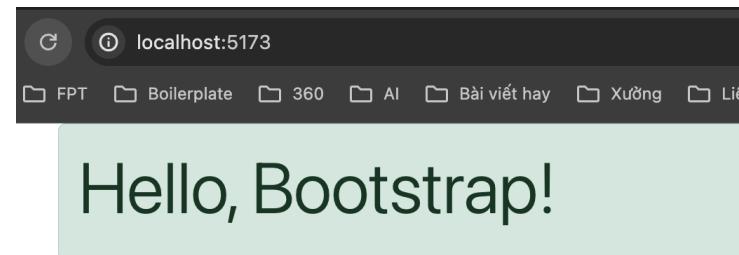
createApp(App).mount("#app");
```

- [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!



Conceive Design Implement Operate



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

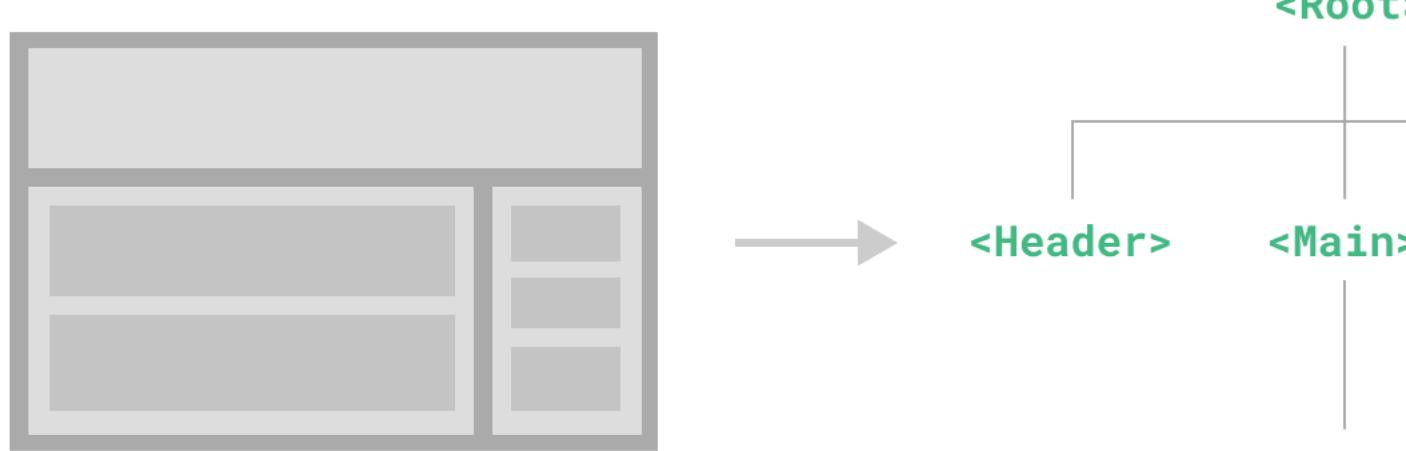




PHẦN 1

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>
```

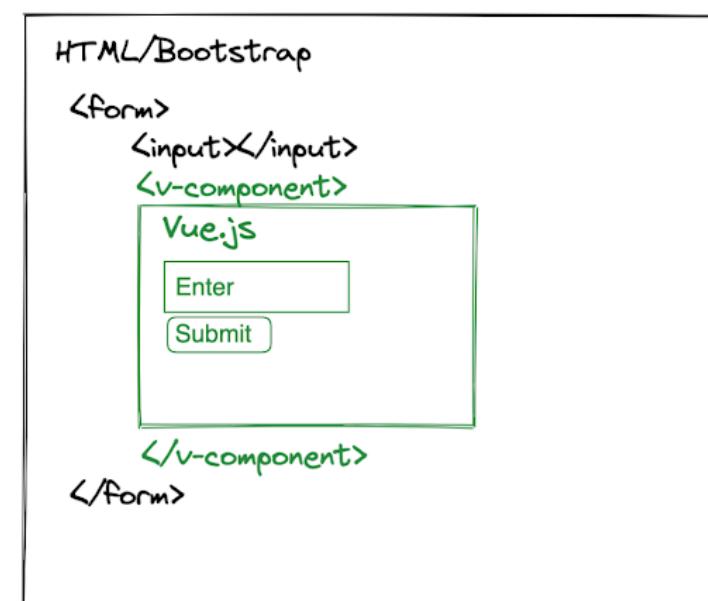
3. Style (CSS)

```
<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.

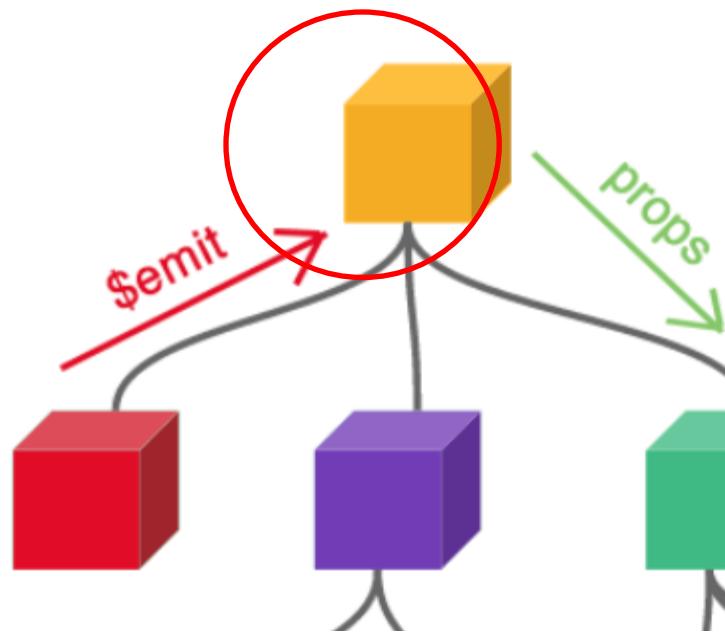
You clicked me 0 times.

click

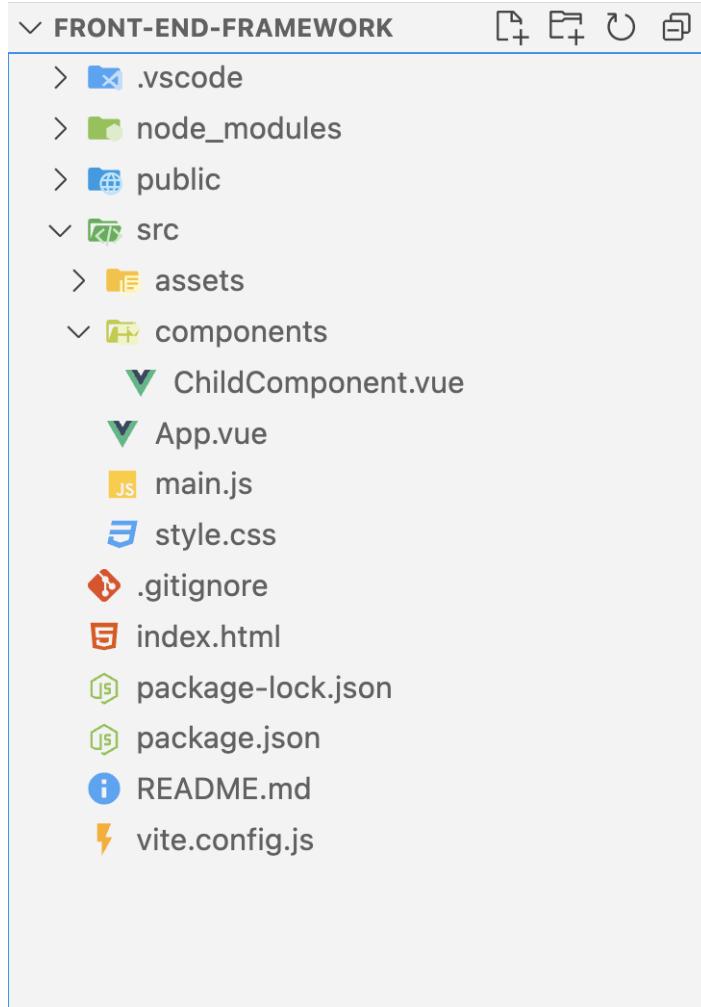
You clicked me 2 times.

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



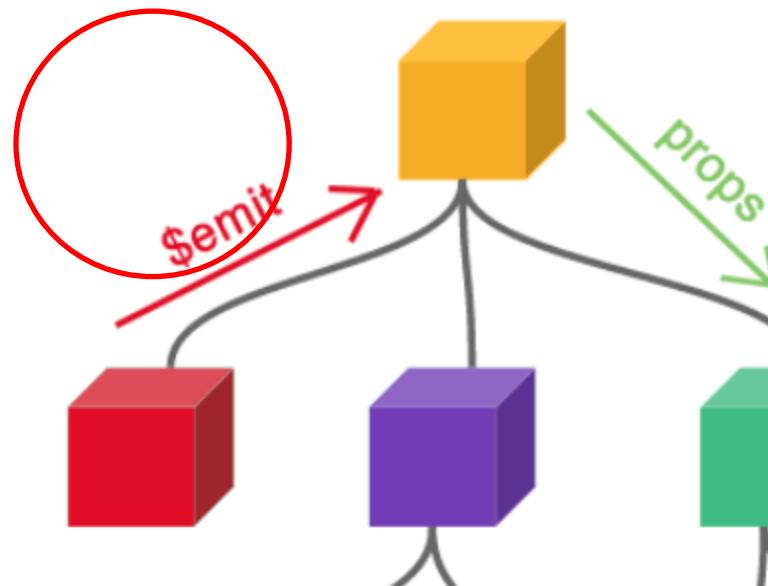
ChildComponent.vue

```
<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

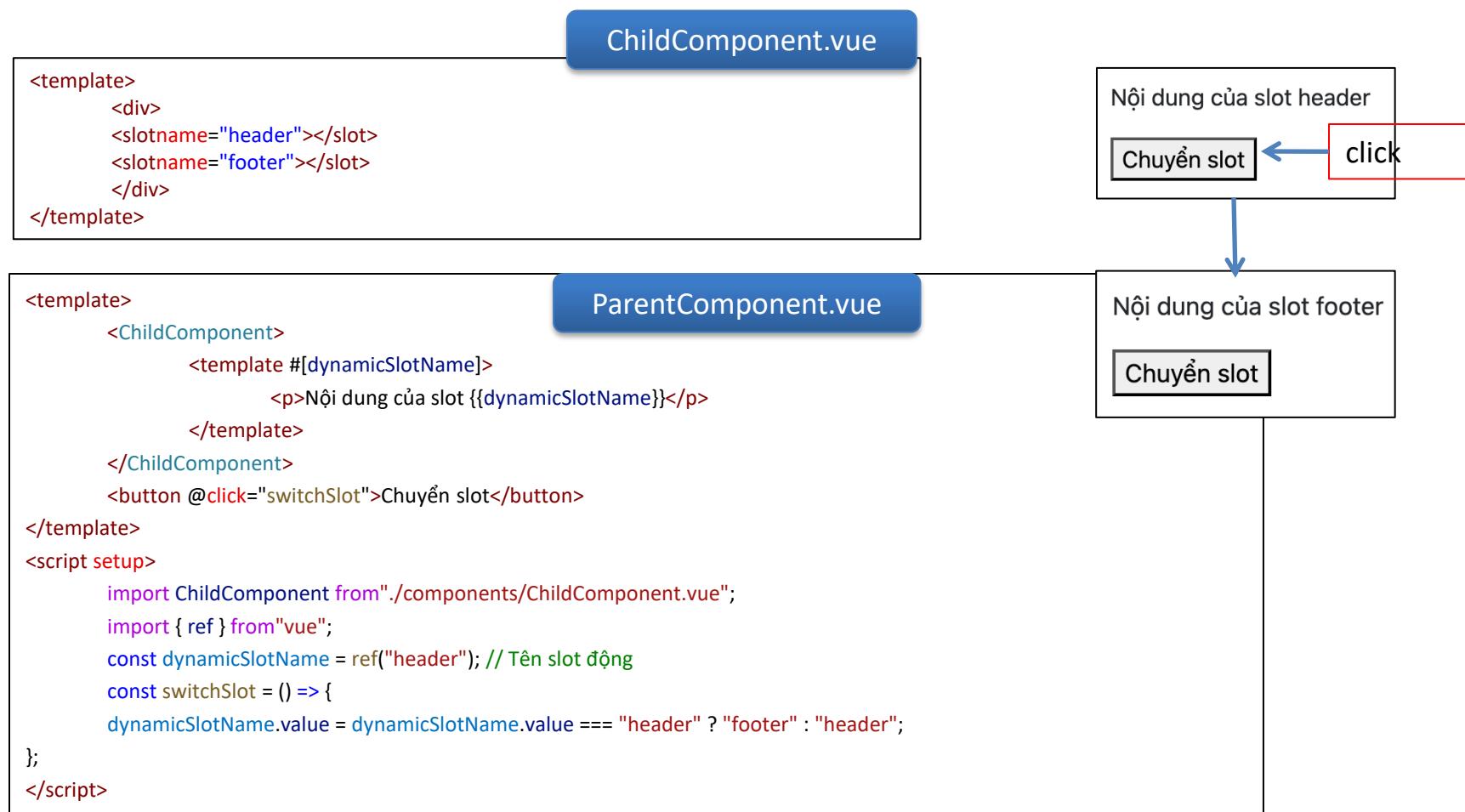
parent template

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

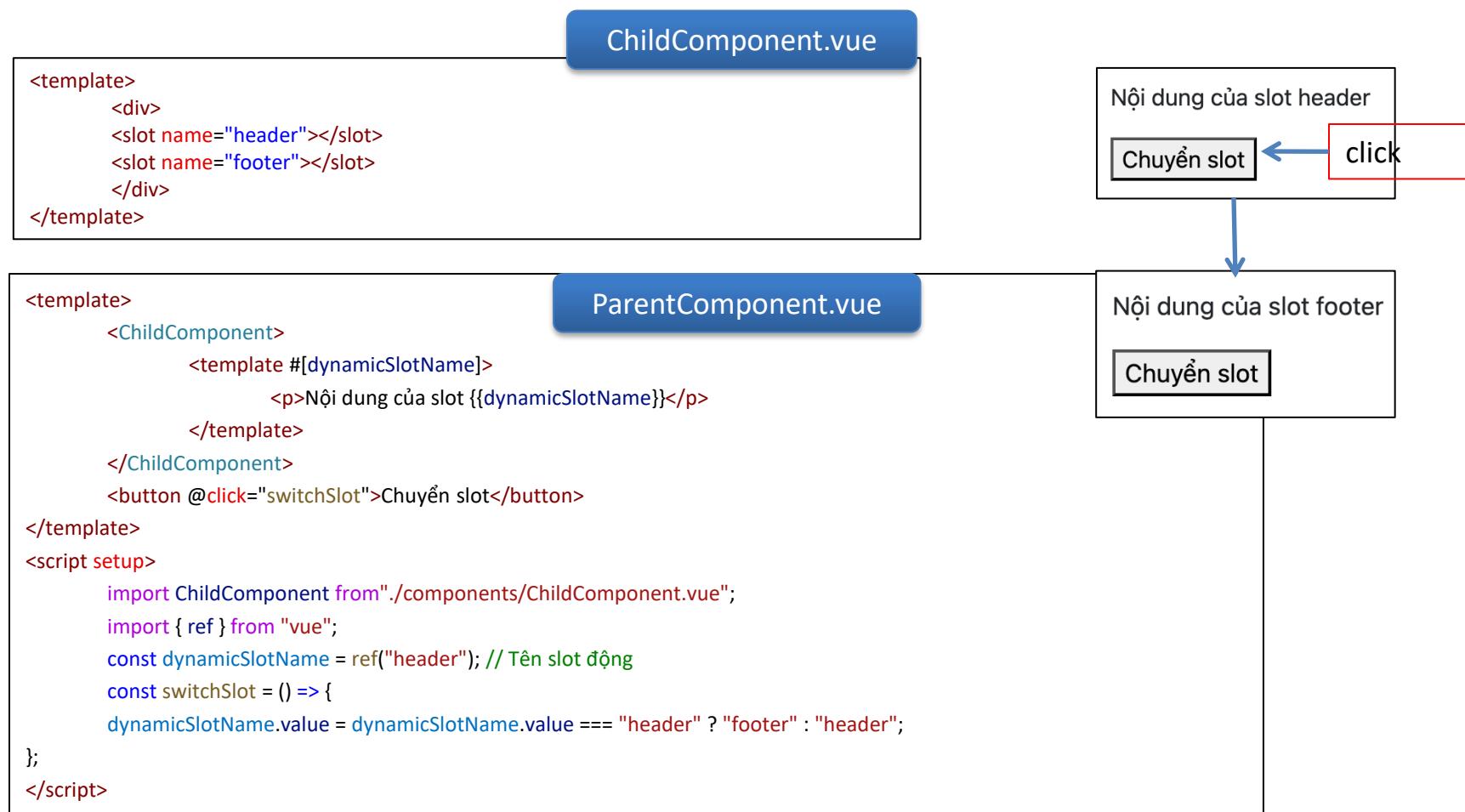
Kết quả

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.

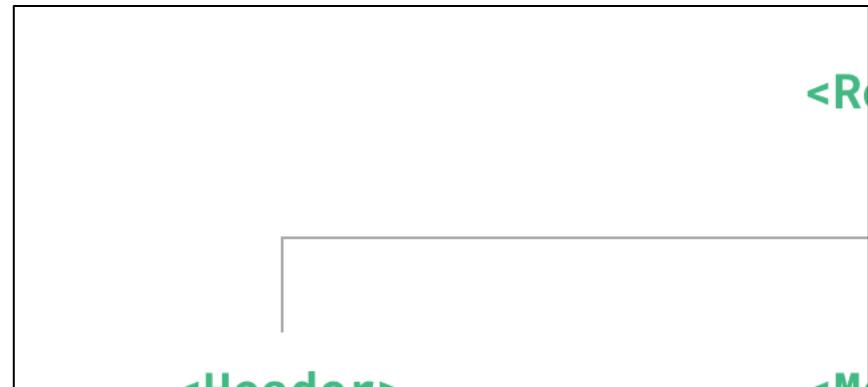


Dynamic Slot Names cho phép sử dụng tên slot thay đổi theo giá trị động, thay vì tên tĩnh như header hay footer.

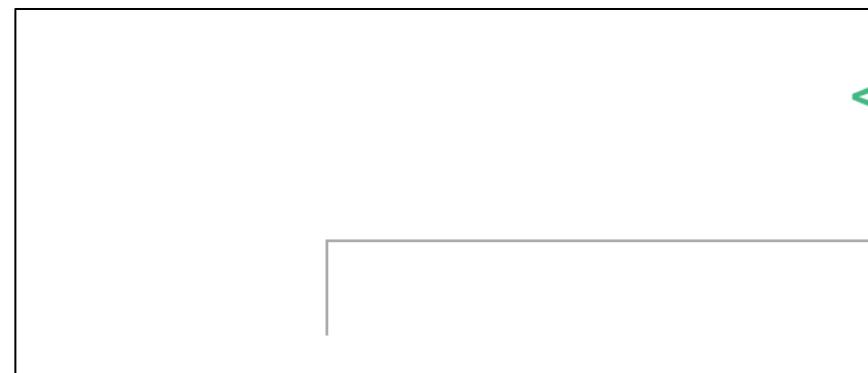


PROVIDE / INJECT

- ❑ 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?

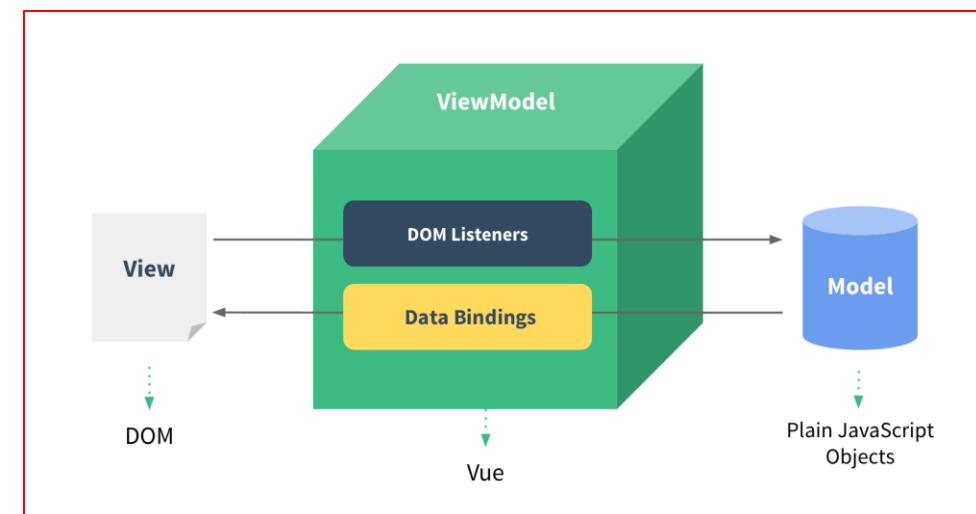
❑ Vue.js tự động đồng bộ dữ liệu giữa Model và View thông qua ViewModel, giúp mã nguồn dễ bảo trì và phát triển hơn. Mô hình này được gọi là **MVVM**

❑ **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
  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 official Node.js website. On the left, there's a green hexagonal background with the text "Run JavaScript Everywhere". Below it is a 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 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

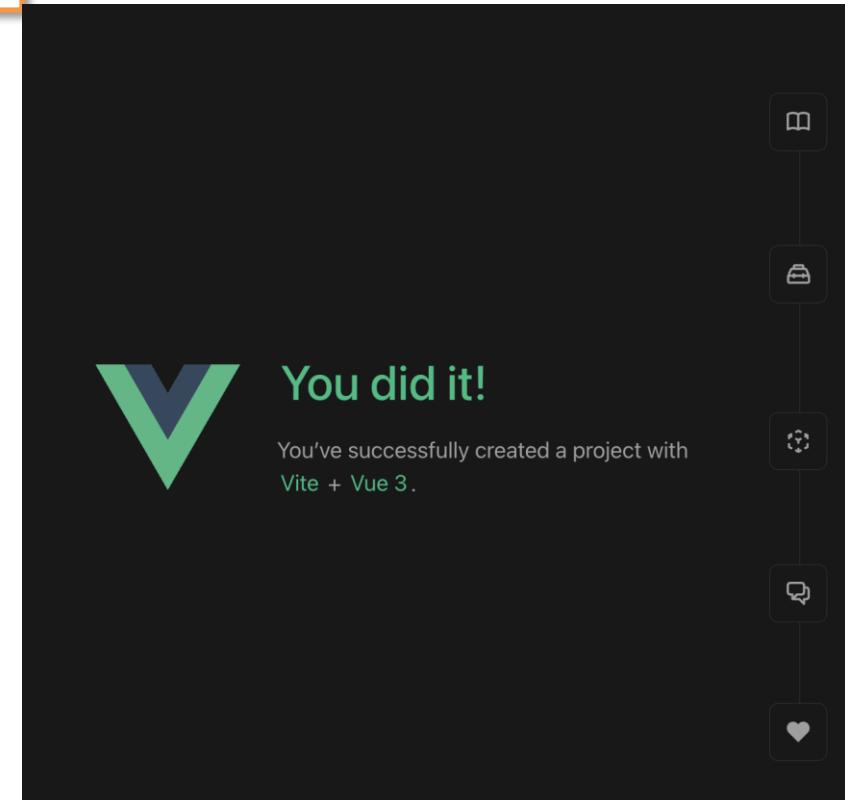
Cài đặt các thư viện

Chạy ứng dụng Vue

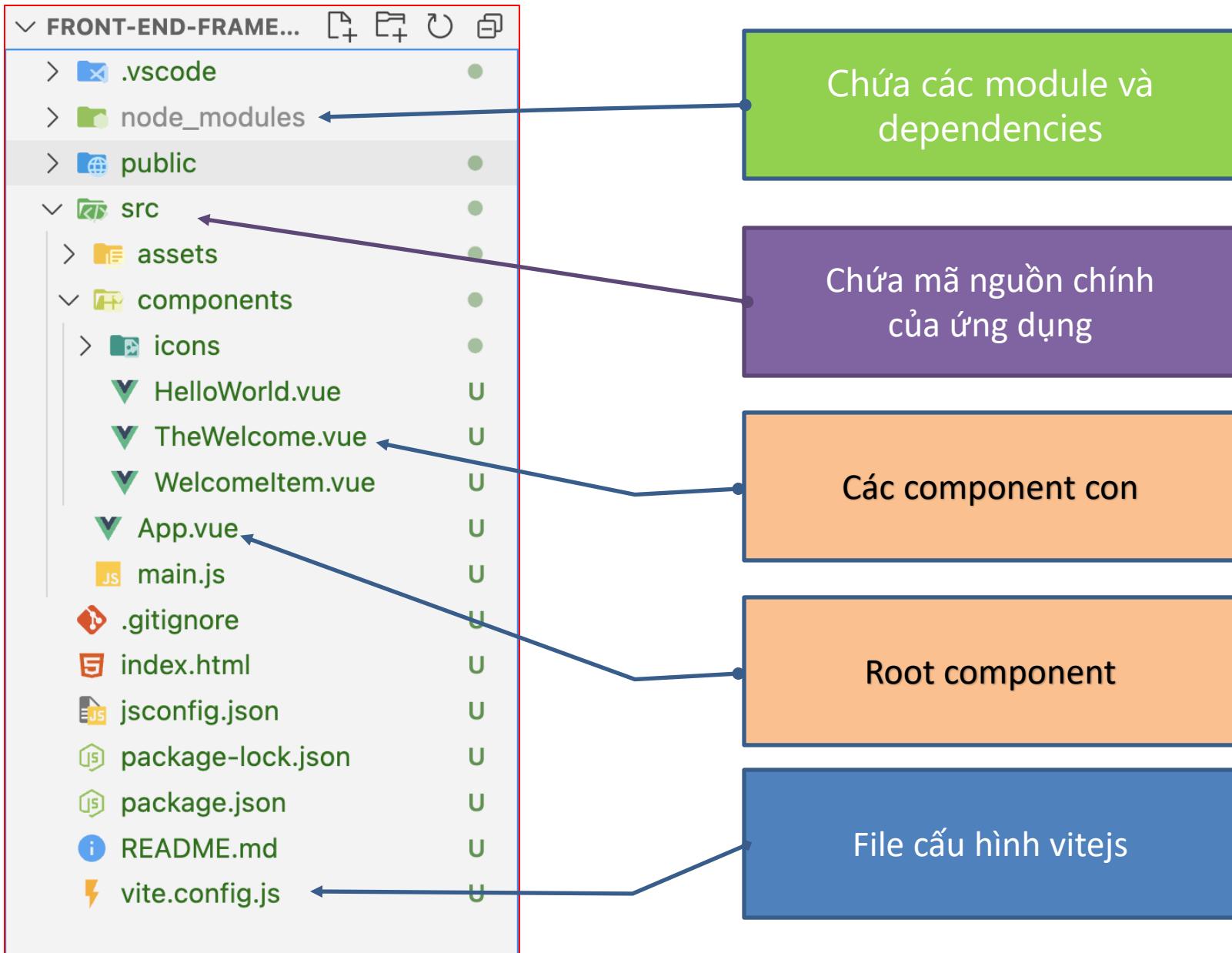
☐ Bật trình duyệt

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");
```

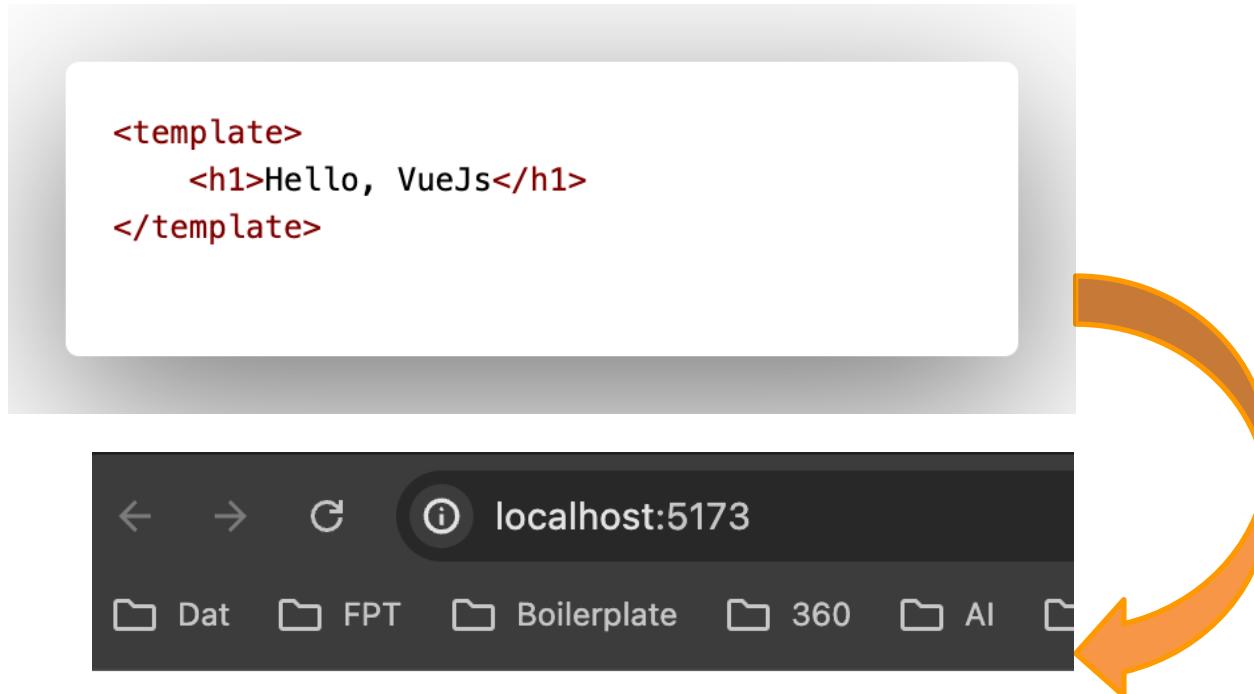
- **<script setup>**: Đơn giản hóa khai báo và sử dụng dữ liệu trong Vue 3.
- **<template>**: Định nghĩa cấu trúc giao diện của thành phần bằng HTML.
- **<style scoped>**: Áp dụng CSS chỉ cho thành phần hiện tại.

```
<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";

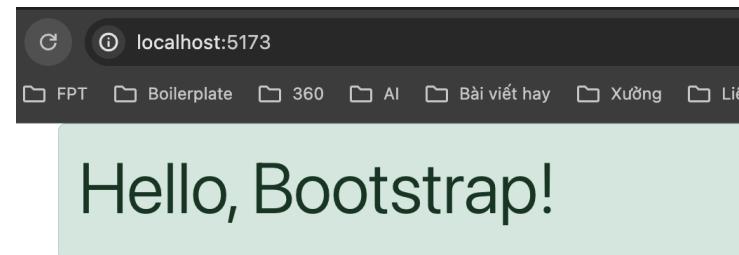
createApp(App).mount("#app");
```

- [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!



TRÍ TUỆ NHÂN TẠO (A.I) DÀNH CHO MỌI NGƯỜI

*Bài giảng biên soạn theo giáo trình “A.I for everyone” của Nhà Khoa học A.I Ông Andrew Ng
(Google Brain - Baidu AI - Deeplearning.ai - Đại học Stanford)*

Understanding the Science for Tomorrow: Myth and Reality của GS Jeffrey C. Grossman

Học viện Công nghệ Massachusetts (MIT)



Nhà khoa học A.I Ông Andrew Ng
Google Brain - Baidu AI - Deeplearning.ai - Đại học Stanford



Thạc sĩ - Nguyễn Ngọc Tú

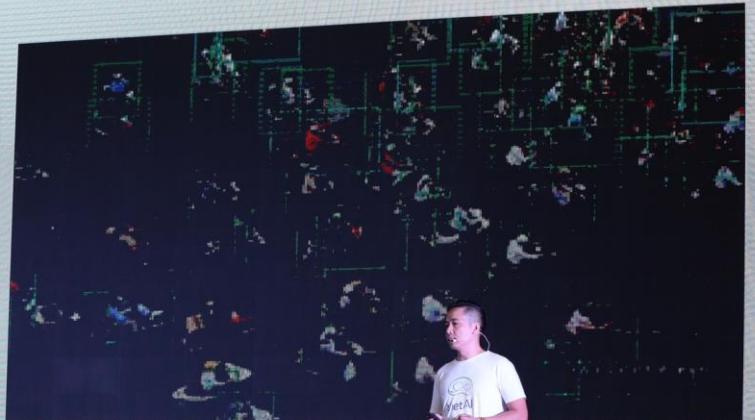
NCS Tiến sỹ tại Strasbourg University - CFVG HCM.

Giám đốc điều hành của Tổ chức Trí Tuệ Nhân Tạo Việt (VietAI). (www.vietai.org)

Chuyên gia Phân tích dữ liệu Doanh Nghiệp (Business Data Analytics), Giám đốc Công ty Dataservices

WHY WE EXIST

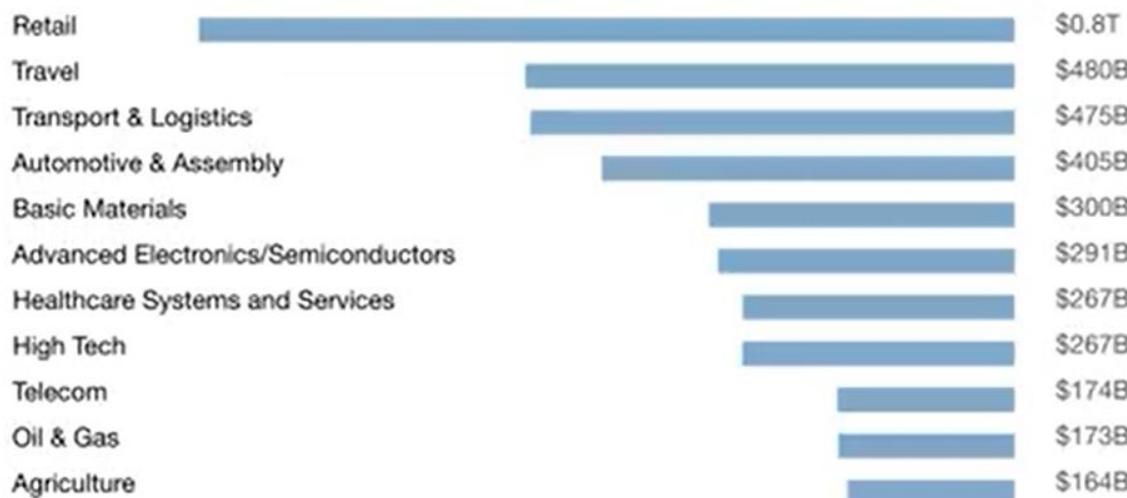
Artificial Intelligence (AI) is all around us under many forms from smart mobile devices, virtual assistants, to robots and self-driving cars. Despite rapid progress in AI over the last few years in many countries, Vietnam still lags behind, merely touching at the surface of the technology.



Introduction

AI value creation
by 2030

\$13
trillion



[Source: McKinsey Global Institute.]

GIẢI MÃ A.I

A.I

ANI

ARTIFICIAL NARROW INTELLIGENCE

TRÍ TUỆ NHÂN TẠO HẸP

VD: XE TỰ HÀNH, LOA THÔNG MINH, TRÍ
TUỆ NHÂN TẠO ỨNG DỤNG TRONG NHÀ
MÁY, NÔNG TRẠI...

AGI

ARTIFICIAL GENERAL INTELLIGENCE

TRÍ TUỆ NHÂN TẠO PHỒ QUÁT

LÀM ĐƯỢC TẤT CẢ NHỮNG GÌ NHƯ MỘT
CON NGƯỜI THẬT

NỘI DUNG CHƯƠNG TRÌNH

1. A.I – TRÍ TUỆ NHÂN TẠO LÀ GÌ?

- HỌC MÁY – MACHINE LEARNING
- DỮ LIỆU - DATA
- MỘT CÔNG TY AI LÀ NHƯ THẾ NÀO?
- NHỮNG ĐIỀU HỌC MÁY CÓ THỂ VÀ KHÔNG THỂ LÀM.
- GIẢI THÍCH VỀ HỌC SÂU (DEEP LEARNING)

2. XÂY DỰNG MỘT DỰ ÁN A.I

3. XÂY DỰNG A.I CHO CÔNG TY/ TỔ CHỨC CỦA BẠN

4. A.I VÀ TÁC ĐỘNG XÃ HỘI CỦA NÓ

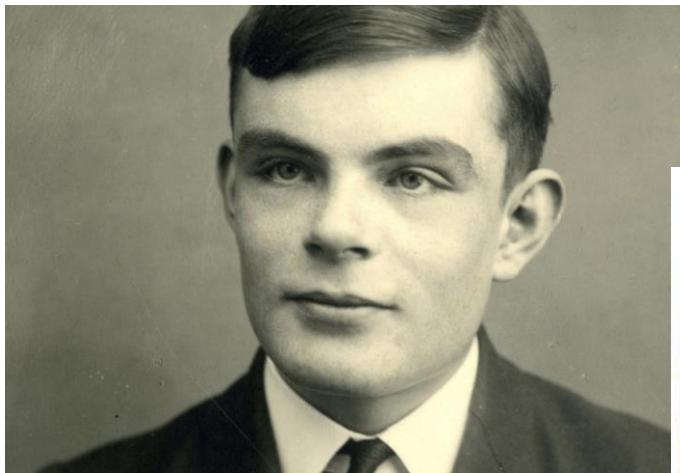
CHÚNG TA MONG MUỐN ĐẠT ĐƯỢC GÌ SAU BUỔI CHIA SẺ?

- 1. BIẾT ĐƯỢC A.I LÀM ĐƯỢC GIÀU.**
- 2. BIẾT ĐƯỢC CÁCH XÂY DỰNG CHIẾN LƯỢC A.I CHO
CÔNG TY MÌNH.**
- 3. BIẾT CÁCH PHÂN BỐ NGUỒN LỰC XD A.I**

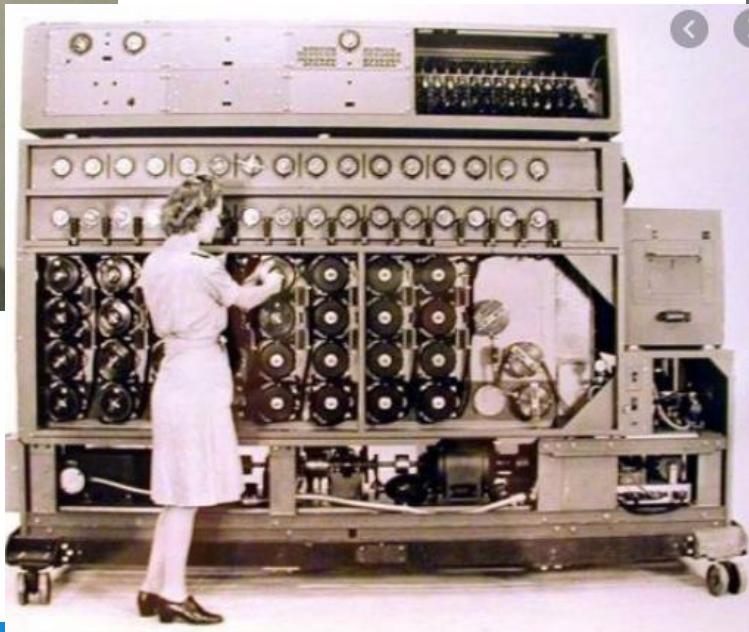
LƯỢC SỬ AI

History of AI

Engima



Alan Turing - 1927



Bombe



A.I – TRÍ TUỆ NHÂN TẠO LÀ GÌ ?

➤ ***Học Máy – Machine Learning***



Ví Dụ

```
93197245103243759034986680965164954  
30242948320135357468514169690142131  
28232382498291391119966979422633316  
63690360301139315049687103799181722  
33807056988414446453343420432614063  
17958043775054209812493520051939618  
95005111747726518241156523304385467  
02161709563266471523235685020279246  
94321002087409793693431483703929632  
55166276756658168710538319574243978  
71759239430458004046669348131311301  
16796411413123481550794845652540711  
016167555068817728376555002835558045  
64687713073869167364880210608898024  
79731327936249214503851916575991545
```



Mommy...?

(Nguồn: [Simple Neural Network implementation in Ruby](#))



ARTIFICIAL INTELLIGENCE

The ability of a computer program or a machine to think like humans do.

MACHINE LEARNING

Subfield of AI giving machines the skills to learn from examples without being explicitly programmed.

Examples: Fraud detection, marketing personalization, email classification



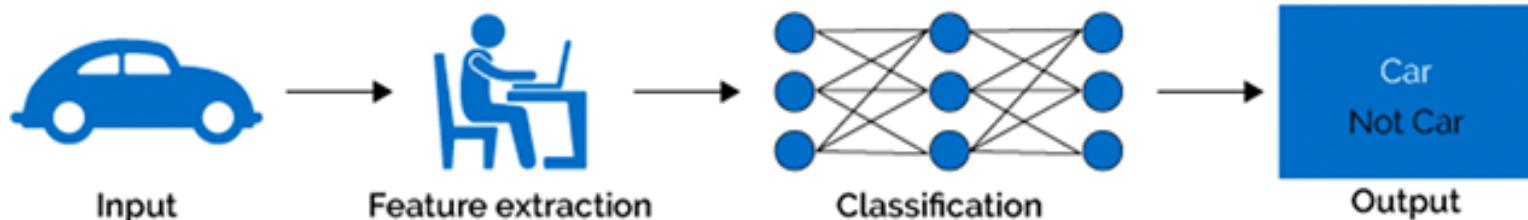
DEEP LEARNING

Specialized machine learning technique enabling machines to train themselves to perform tasks.

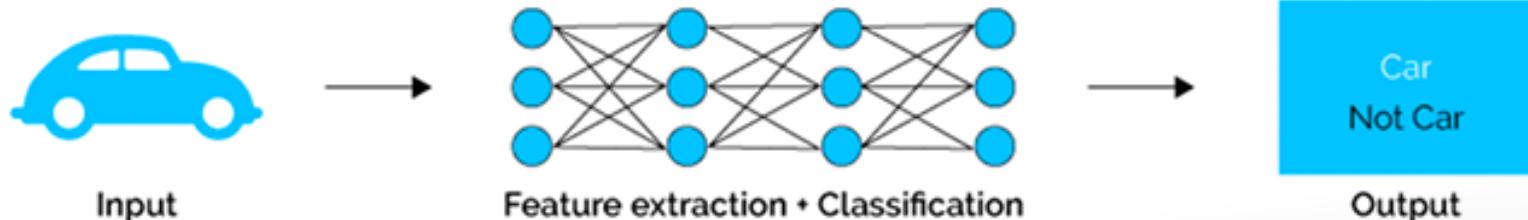
Examples: Image classification, vehicle detection, sentiment analysis



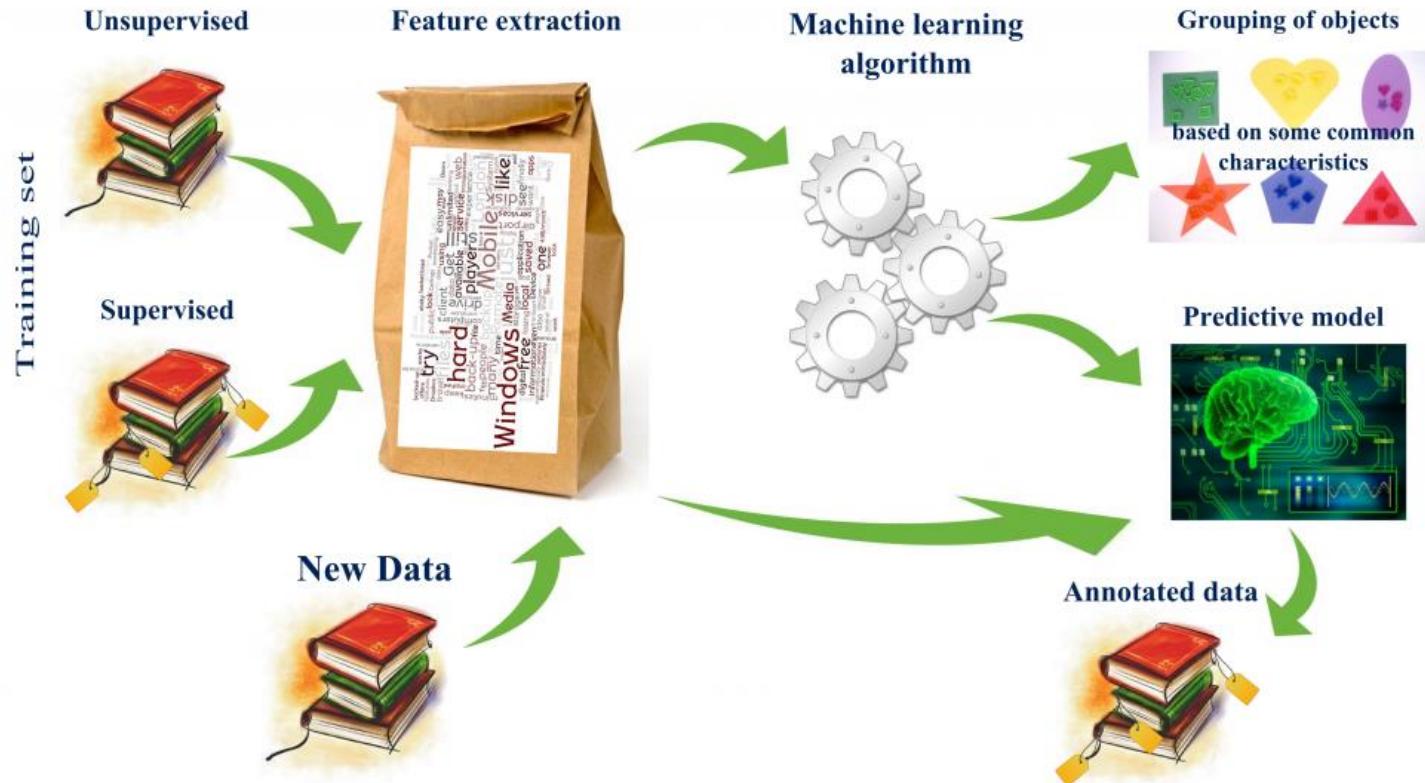
Machine Learning



Deep Learning



Machine learning workflow



SUPERVISED LEARNING

HỌC CÓ GIÁM SÁT



Input
(Đầu vào) -----> Output
(Đầu ra)

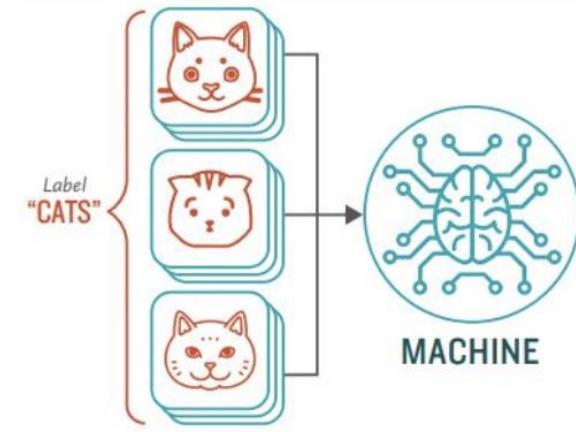
Đầu vào (Input)	Đầu ra (Output)	Ứng dụng (Applications)
Email	Thư rác?	Bộ lọc thư rác.
Âm thanh	Văn bản	Nhận dạng giọng nói
Tiếng Anh	Tiếng Việt	Máy dịch (Machine translation)
Q/C, thông tin người dùng	Nhấp chuột? (0/1)	Quảng cáo online

Đầu vào (Input)	Đầu ra (Output)	Ứng dụng (Applications)
Hình ảnh, Thông tin rada	Vị trí của các xe ô tô	Xe tự lái (tự hành)
Hình ảnh của chiếc điện thoại	Bị hư hỏng ? (0/1)	Kiểm tra trực quan

How Supervised Machine Learning Works

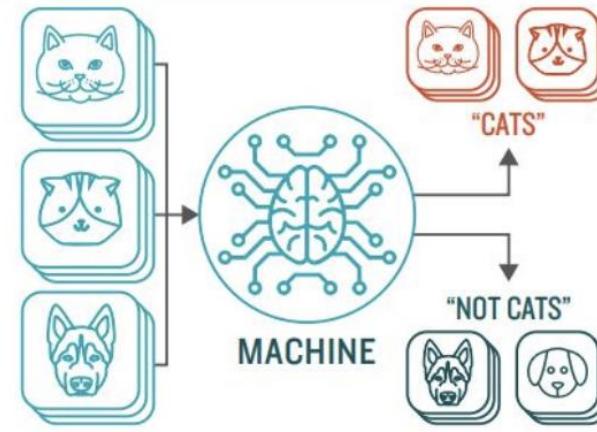
STEP 1

Provide the machine learning algorithm categorized or "labeled" input and output data from to learn

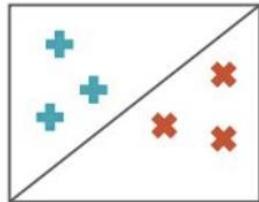


STEP 2

Feed the machine new, unlabeled information to see if it tags new data appropriately. If not, continue refining the algorithm

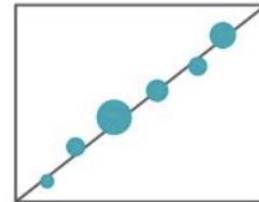


TYPES OF PROBLEMS TO WHICH IT'S SUITED



CLASSIFICATION

Sorting items into categories



REGRESSION

Identifying real values (dollars, weight, etc.)

Bài toán:

Có dữ liệu bệnh nhân như bảng dưới đây. Hãy xây dựng model để tiên đoán khả năng bị bệnh tim của Bệnh nhân số 09 -10

STT	Cân nặng	Chiều cao	Huyết áp	Vận động	Bệnh tim
1	Nhẹ	Trung bình	Trung bình	Nhiều	Không
2	Nặng	Thấp	Cao	Ít	Có
3	Nhẹ	Thấp	Cao	Ít	Có
4	Nặng	Cao	Cao	Trung bình	Không
5	Nhẹ	Cao	Cao	Nhiều	Không
6	Trung bình	Thấp	Trung bình	Nhiều	Không
7	Trung bình	Trung bình	Trung Bình	Ít	Không
8	Nặng	Thấp	Thấp	Nhiều	Có
9	Nhẹ	Cao	Trung bình	Ít	???
10	Nhẹ	Cao	Trung bình	Nhiều	???

Bài toán:

Có dữ liệu bệnh nhân như bảng dưới đây. Hãy xây dựng model để tiên đoán khả năng bị bệnh tim của Bệnh nhân số 09 -10

STT	Cân nặng	Chiều cao	Huyết áp	Vận động	Bệnh tim
1	Nhẹ	Trung bình	Trung bình	Nhiều	Không
2	Nặng	Thấp	Cao	Ít	Có
3	Nhẹ	Thấp	Cao	Ít	Có
4	Nặng	Cao	Cao	Trung bình	Không
5	Nhẹ	Cao	Cao	Nhiều	Không
6	Trung bình	Thấp	Trung bình	Nhiều	Không
7	Trung bình	Trung bình	Trung Bình	Ít	Không
8	Nặng	Thấp	Thấp	Nhiều	Có
9	Nhẹ	Cao	Trung bình	Ít	???
10	Nhẹ	Cao	Trung bình	Nhiều	???

QUY ƯỚC

STT	Tên	Giá trị
Input		
1	Nhẹ	1
2	Thấp	2
3	Trung bình	3
4	Cao	4
5	Nặng	5
6	Ít	6
7	Nhiều	7
Output	Có	1
	Không	0

DỰ ĐOÁN

9	Nhẹ	Cao	<i>Trung bình</i>	Ít	???
dactrung	1	4	3	6	???

UN-SUPERVISED LEARNING

HỌC KHÔNG GIÁM SÁT

A -----> B

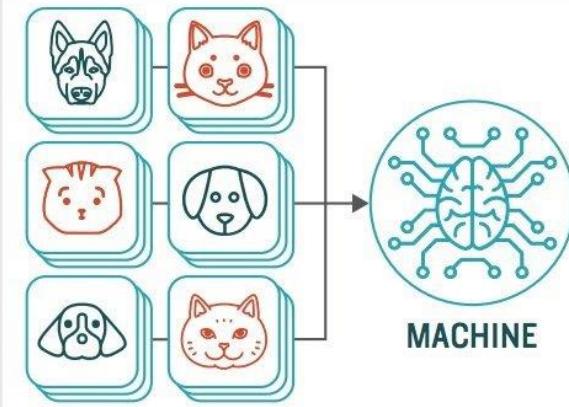
Input -----> Output??
(Đầu vào) (Đầu ra)



How Unsupervised Machine Learning Works

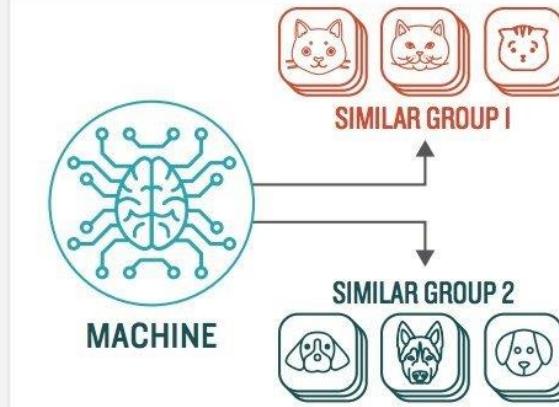
STEP 1

Provide the machine learning algorithm uncategorized, unlabeled input data to see what patterns it finds

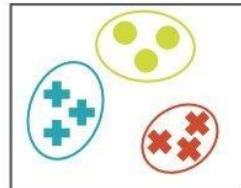


STEP 2

Observe and learn from the patterns the machine identifies



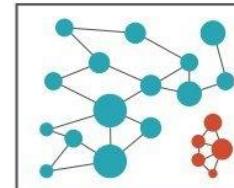
TYPES OF PROBLEMS TO WHICH IT'S SUITED



CLUSTERING

Identifying similarities in groups

For Example: Are there patterns in the data to indicate certain patients will respond better to this treatment than others?



ANOMALY DETECTION

Identifying abnormalities in data

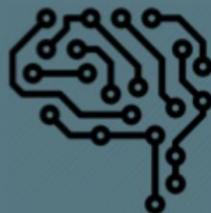
For Example: Is a hacker intruding in our network?

ARTIFICIAL INTELLIGENCE

IS NOT NEW

ARTIFICIAL INTELLIGENCE

Any technique which enables computers to mimic human behavior



1950's

1960's

1970's

1980's

1990's

2000's

2010s

ORACLE

MACHINE LEARNING

AI techniques that give computers the ability to learn without being explicitly programmed to do so

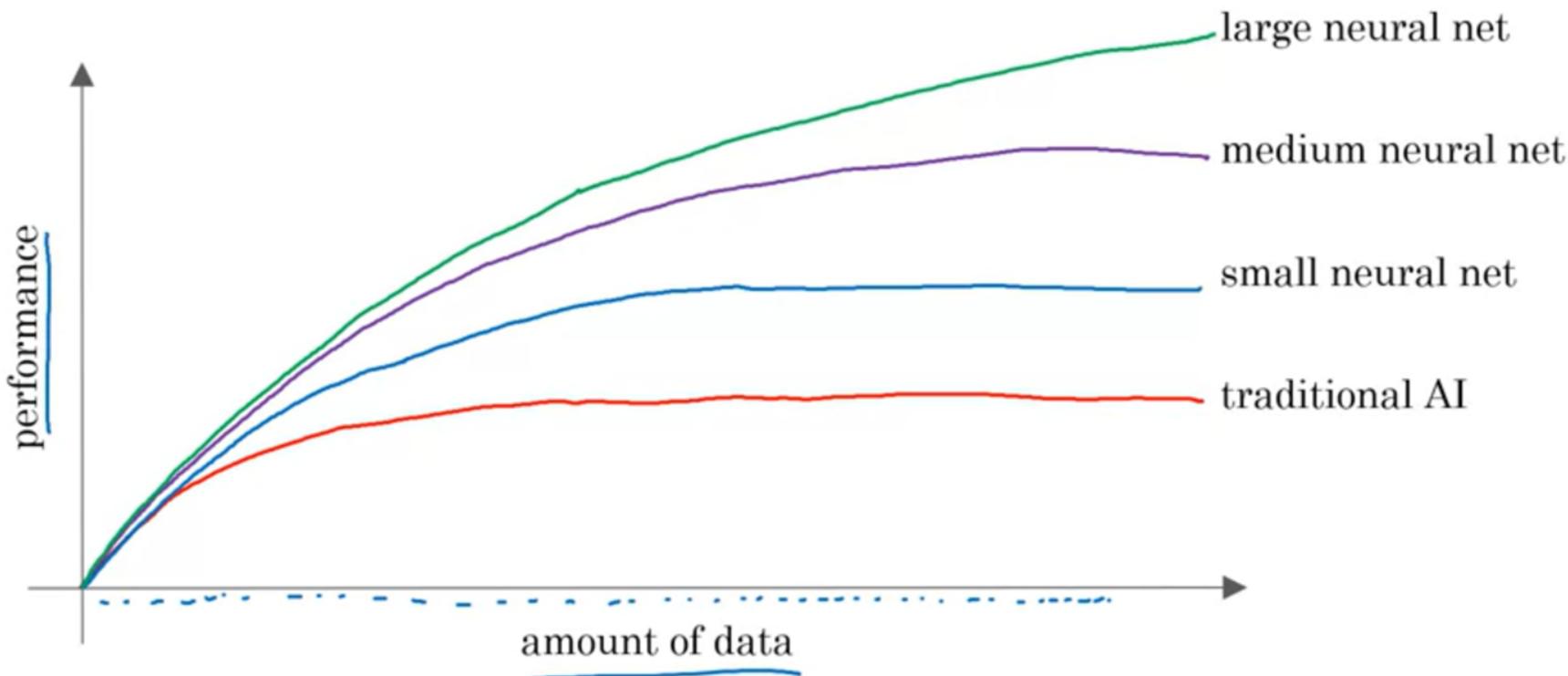


DEEP LEARNING

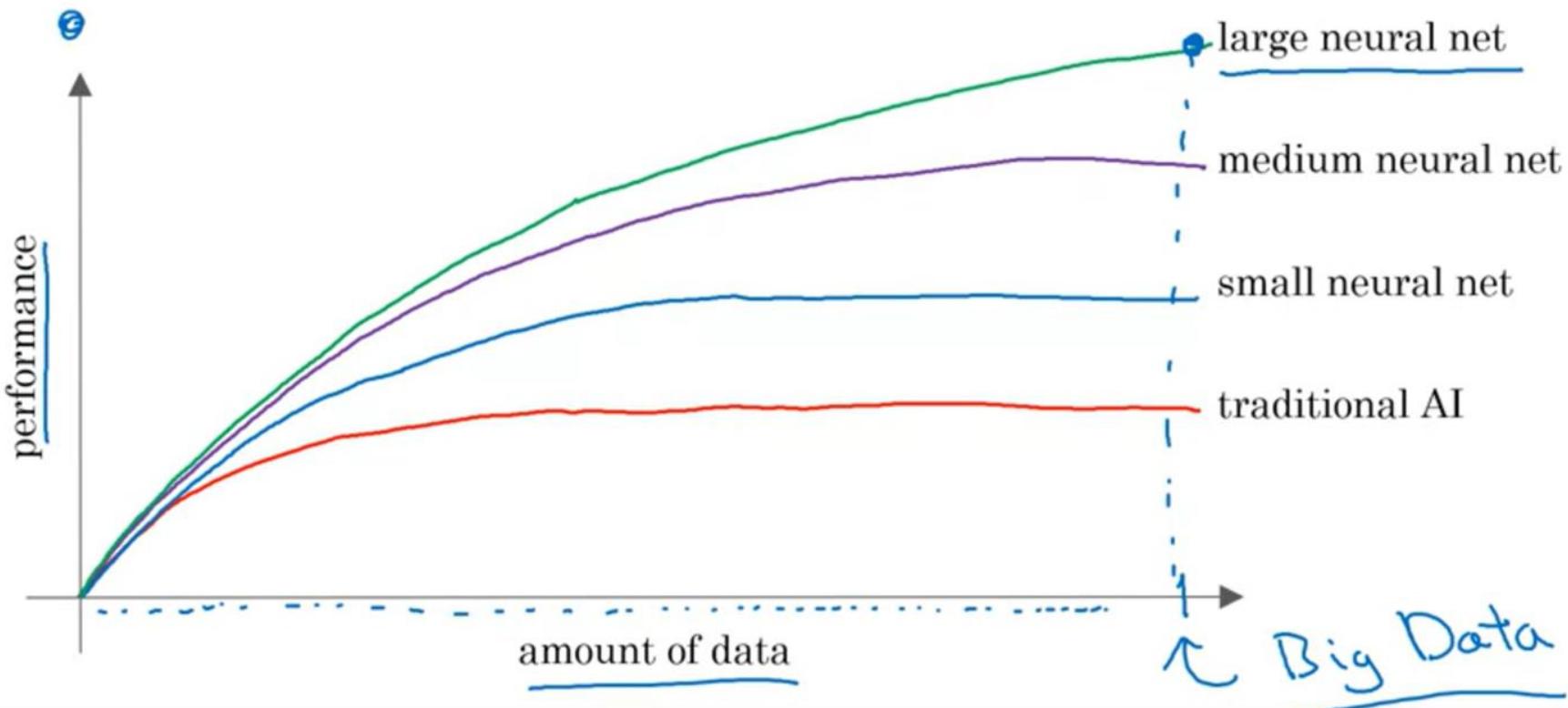
A subset of ML which make the computation of multi-layer neural networks feasible



TẠI SAO BÂY GIỜ LẠI NÓI VỀ A.I?



DỮ LIỆU LỚN !



A.I – TRÍ TUỆ NHÂN TẠO LÀ GÌ ?

➤ *Dữ Liệu – Data*

DATA



SORTED



ARRANGED



PRESENTED
VISUALLY



VÍ DỤ CỦA BẢNG SỐ LIỆU (TẬP DỮ LIỆU - DATASET)

Diện tích căn nhà (m ²)		Giá nhà (tỷ đồng)
100		4.5
150		6.0
135		5.5
200		6.7
500		10
600		15
1,000	A	20



VÍ DỤ CỦA BẢNG SỐ LIỆU (TẬP DỮ LIỆU - DATASET)

Diện tích căn nhà (m ²)	Số lượng phòng ngủ	Giá nhà (tỷ đồng)
100	3	4.5
150	4	6.0
135	4	5.5
200	5	6.7
500	5	10
600	6	15
1,000	8	20

The diagram illustrates a linear relationship between two variables. On the left, a vertical stack of seven data points from the table is labeled 'A'. A horizontal bracket below these points also contains the letter 'A'. On the right, a vertical stack of five data points is labeled 'B'. A horizontal bracket below these points also contains the letter 'B'. A dashed arrow points from the 'A' bracket to the 'B' bracket, indicating a positive correlation or mapping between the two sets of values.

VÍ DỤ CỦA BẢNG SỐ LIỆU (TẬP DỮ LIỆU - DATASET)

Diện tích căn nhà (m ²)	Số lượng phòng ngủ	Giá nhà (tỷ đồng)
100	3	4.5
150	4	6.0
135	4	5.5
200	5	6.7
500	5	10
600	6	15
1,000	8	20

The diagram illustrates a linear regression model. It shows a blue bracket spanning the first two columns of the table, labeled 'A' at the bottom left. Another blue bracket spans the last two columns, labeled 'B' at the bottom right. A dashed arrow points from 'A' to 'B', representing the prediction function $\hat{y} = f(x)$.

VÍ DỤ CỦA BẢNG SỐ LIỆU (TẬP DỮ LIỆU - DATASET)

Hình ảnh	Nhãn
	Mèo
	Không phải mèo
	Mèo
	Không phải mèo

VÍ DỤ CỦA BẢNG SỐ LIỆU - (TẬP DỮ LIỆU - DATASET)

A → B

Hình ảnh A	Nhãn B
	Mèo
	Không phải mèo
	Mèo
	Không phải mèo



THU THẬP DỮ LIỆU (ACQUIRING)

➤ DÁN NHÃN THỦ CÔNG



Mèo



Mèo



Không phải
Mèo



Không phải
Mèo

THU THẬP DỮ LIỆU (ACQUIRING)

➤ QUAN SÁT HÀNH VI

user ID	time	price (\$)	purchased
4783	Jan 21 08:15.20	7.95	yes
3893	March 3 11:30.15	10.00	yes
8384	June 11 14:15.05	9.50	no
0931	Aug 2 20:30.55	12.90	yes

machine	temperature (°C)	pressure (psi)	machine fault
17987	60	7.65	N
34672	100	25.50	N
08542	140	75.50	Y
98536	165	125.00	Y

THU THẬP DỮ LIỆU (ACQUIRING)

➤ QUAN SÁT HÀNH VI

user ID	time	price (\$)	purchased
4783	Jan 21 08:15.20	7.95	yes
3893	March 3 11:30.15	10.00	yes
8384	June 11 14:15.05	9.50	no
0931	Aug 2 20:30.55	12.90	yes

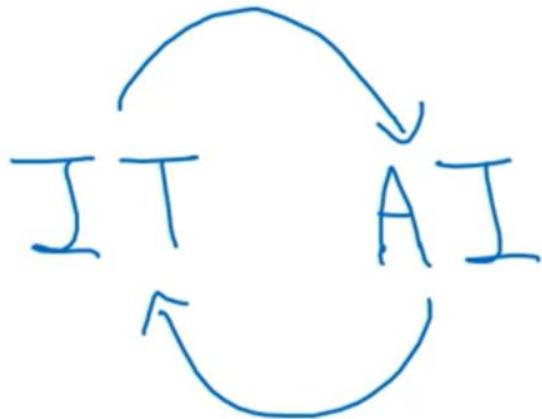
machine	temperature (°C)	pressure (psi)	machine fault
17987	60	7.65	N
34672	100	25.50	N
08542	140	75.50	Y
98536	165	125.00	Y



➤ LƯU GIỮ XUỐNG TỪ WEBSITES/ĐỒI TÁC

SỬ DỤNG DỮ LIỆU HIỆU QUẢ

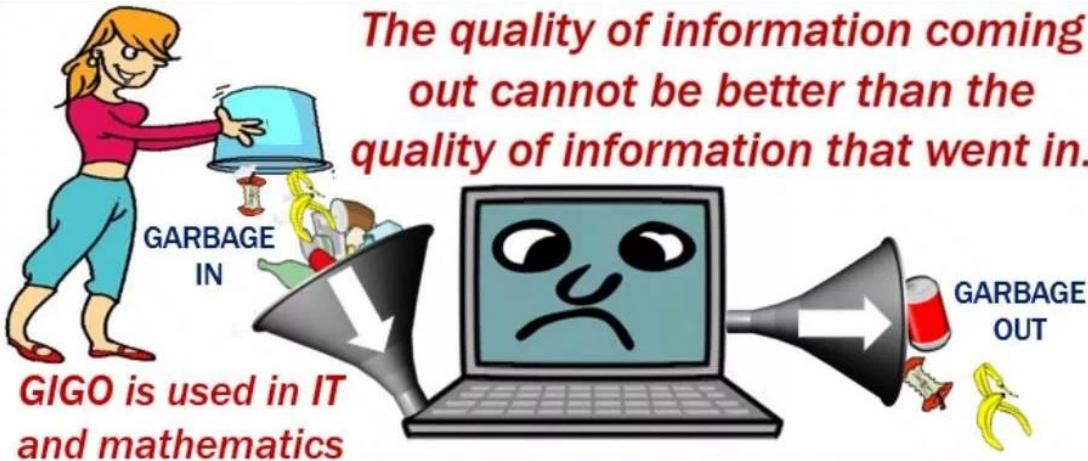
Đừng “ném” dữ liệu bạn có
cho đội nhóm AI và tự cho
rằng nó sẽ có giá trị!



Don't throw data
at an AI team and
assume it will be
valuable.

VÂN ĐÈ CỦA DỮ LIỆU

- Vào thế nào, ra như vậy !
Garbage in, garbage out!



Garbage In, Garbage Out

VÂN ĐỀ CỦA DỮ LIỆU

- **Dữ liệu có vân đề của nó!**
 - Sai lệch nhẫn
 - Thiếu giá trị

Diện tích căn nhà (m ²)	Số lượng phòng ngủ	Giá nhà (tỷ đồng)
100	3	4.5
150	4	#0.09
135	4	5.5
200	#Không biết	6.7
500	5	10
#Không biết	6	#Không biết
1,000	#Không biết	20

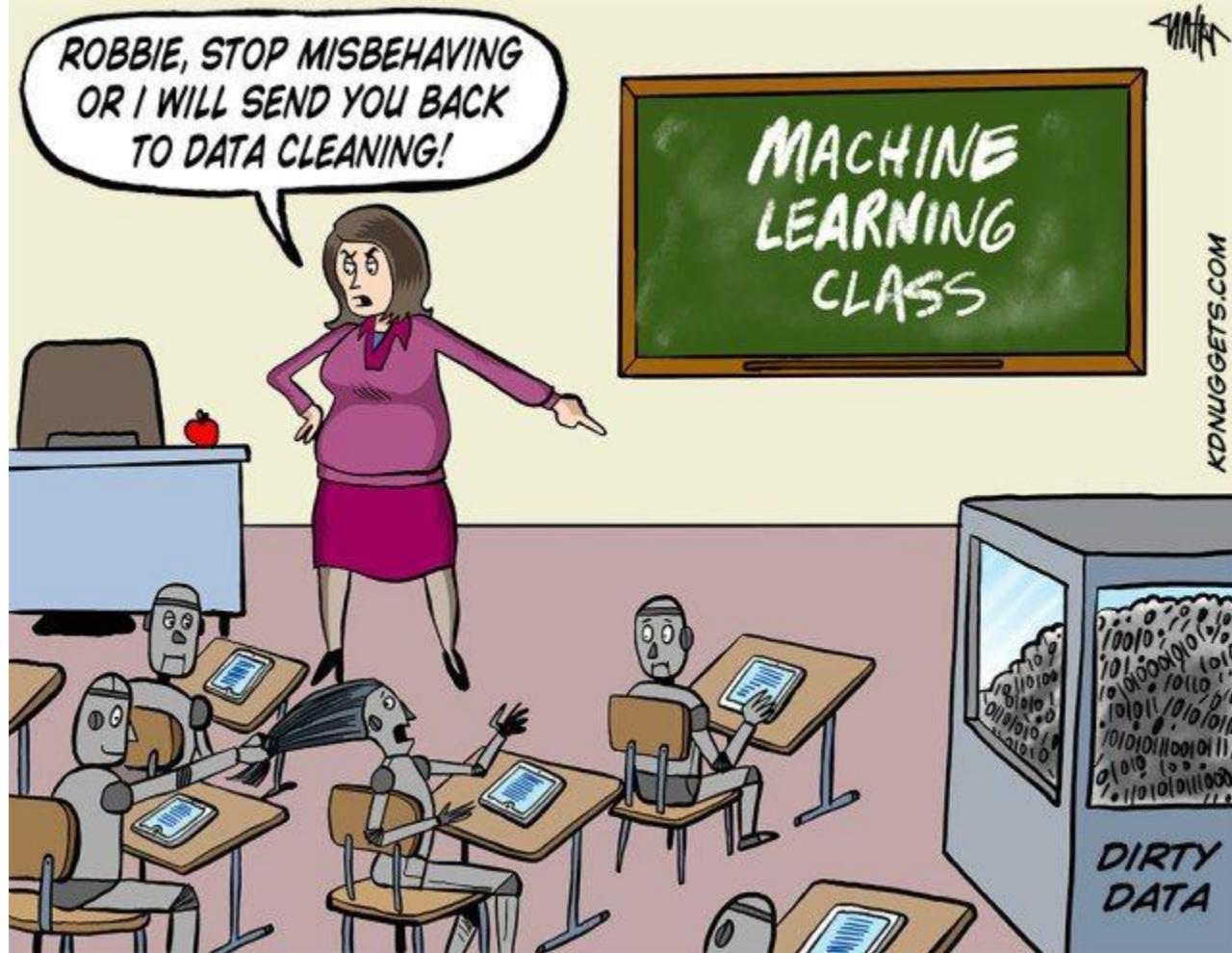
VĂN ĐỀ CỦA DỮ LIỆU

- **Đa dạng về loại dữ liệu**
 - Hình ảnh, âm thanh, câu chữ

(Dữ liệu Phi cấu trúc Un-structured)

Dữ liệu có cấu trúc
(Structured data)

Diện tích căn nhà (m ²)	Số lượng phòng ngủ	Giá nhà (tỷ đồng)
100	3	4.5
150	4	#0.09
135	4	5.5
200	#Không biết	6.7
500	5	10
#Không biết	6	#Không biết
1,000	#Không biết	20



KDNUGGETS.COM

A.I – TRÍ TUỆ NHÂN TẠO LÀ GÌ ?

➤ *Các Thuật ngữ trong AI –
The terminology of AI*

HỌC MÁY vs. KHOA HỌC DỮ LIỆU

Machine Learning vs. Data Science

Diện tích căn nhà (m ²)	Số lượng phòng ngủ	Số lượng phòng tắm	Mới sửa chữa	Giá nhà (tỷ đồng)
100	3	2	N	4.5
150	4	3	Y	5.0
135	4	3	Y	5.5
200	5	4	N	6.7
500	5	5	Y	10
600	6	5	Y	15
1,000	8	8	N	20



HỌC MÁY vs. KHOA HỌC DỮ LIỆU

Machine Learning vs. Data Science

Diện tích căn nhà (m ²)	Số lượng phòng ngủ	Số lượng phòng tắm	Mới sửa chữa	Giá nhà (tỷ đồng)
100	3	2	N	4.5
150	4	3	N	5.0
145	4	3	Y	5.5
200	5	4	N	6.7
500	5	5	Y	10
600	6	5	Y	15
1,000	8	8	N	20

A

B

Data Science

- ✓ Giá nhà có 3 phòng ngủ thường đắt hơn 2 phòng ngủ nếu cùng diện tích.
- ✓ Cùng một diện tích thì nhà mới sửa bán được giá hơn 10%.

HỌC MÁY vs. KHOA HỌC DỮ LIỆU

Machine Learning vs. Data Science

HỌC MÁY

Lĩnh vực khoa học nghiên cứu mà giúp cho các máy tính có khả năng tự học được mà không cần đến một chương trình/phần mềm cụ thể.

Arthur Samuel (1959)

PHẦN MỀM/ SOFTWARE

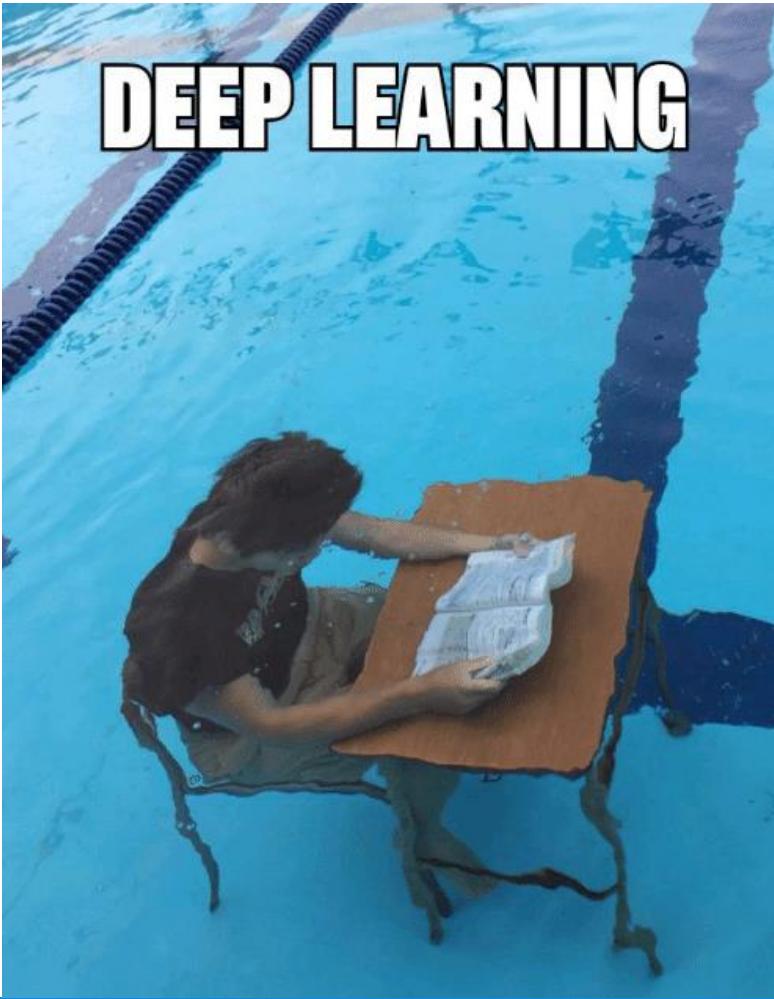
KHOA HỌC DỮ LIỆU

Ngành khoa học khai thác tri thức và tìm ra những ý nghĩa ẩn từ dữ liệu.

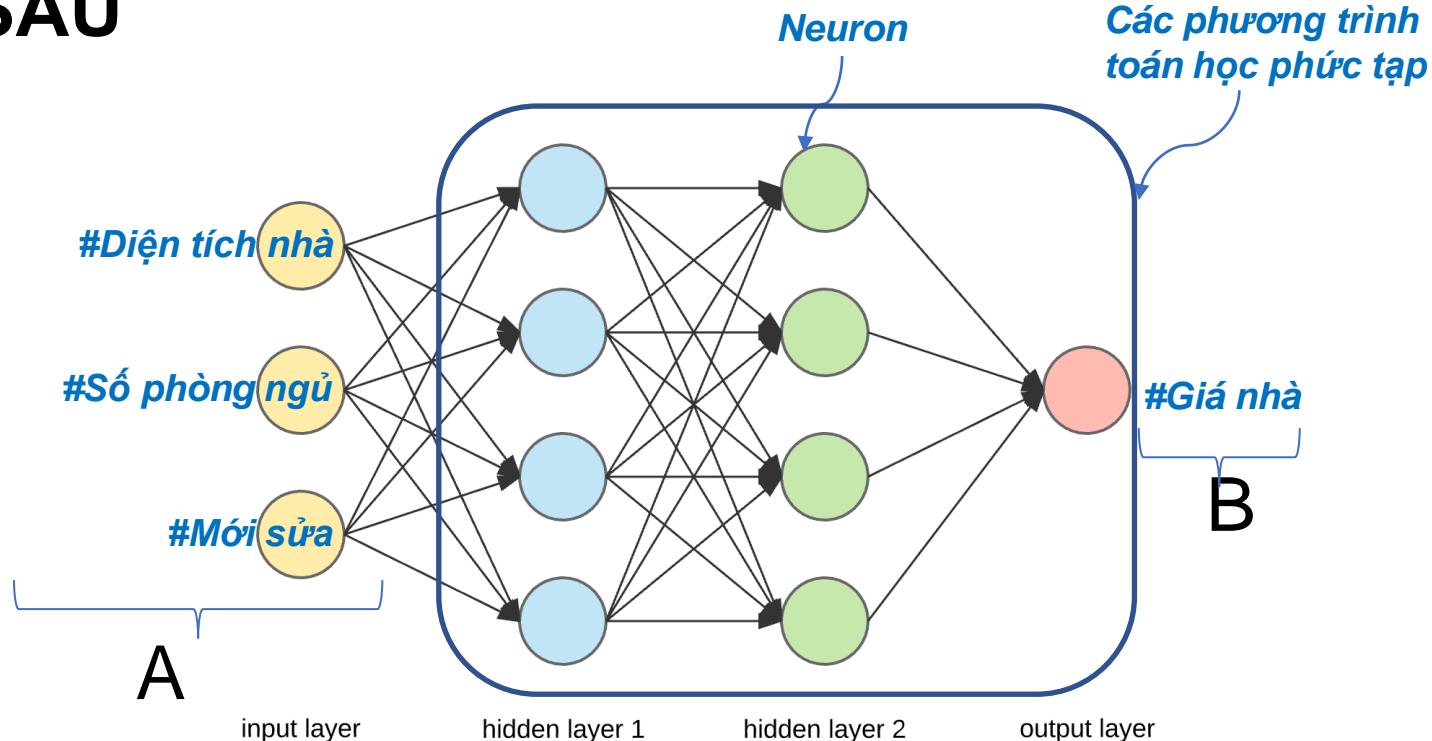
BÁO CÁO/TỔNG HỢP

HỌC SÂU

DEEP LEARNING



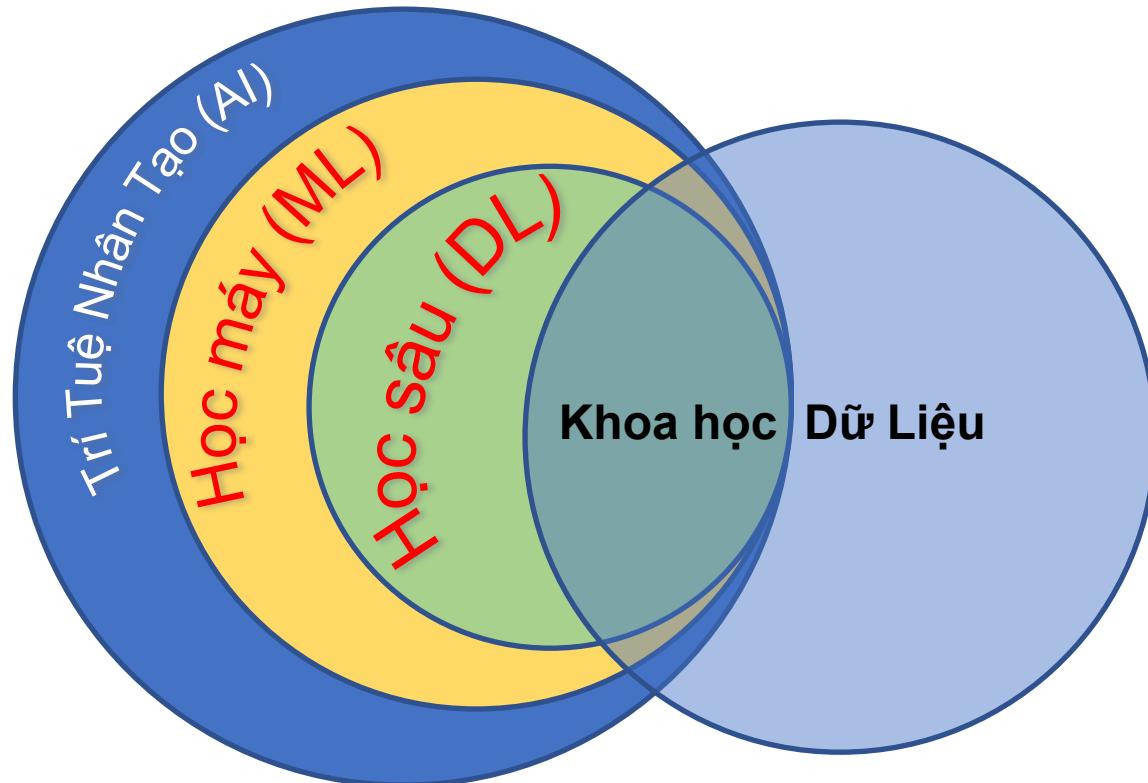
HỌC SÂU



Mạng nơ-ron được thiết kế dựa trên bộ não con người, nhưng cách thức chúng vận hành thì hoàn toàn không giống cách bộ não sinh học của con người hoạt động.

AI – TRÍ TUỆ NHÂN TẠO CÓ NHIỀU CÔNG CỤ

- Học máy và Khoa học dữ liệu
- Học sâu và Mạng nơ-ron
- Và các thuật ngữ biến khác như: Học không giám sát (Un-supervised Learning), Học tăng cường (Reinforcement Learning), mô hình đồ họa (Graphical models)...



A.I – TRÍ TUỆ NHÂN TẠO LÀ GÌ ?

➤ *Những điều ML có thể và không thể làm.*

HỌC CÓ GIÁM SÁT

Đầu vào (Input)	Đầu ra (Output)	Ứng dụng (Applications)
Email	Thư rác?	Bộ lọc thư rác.
Âm thanh	Văn bản	Nhận dạng giọng nói
Tiếng Anh	Tiếng Việt	Máy dịch (Machine translation)
Q/C, thông tin người dùng	Nhấp chuột? (0/1)	Quảng cáo online

Những điều mà bạn có thể làm mà chỉ cần tốn 01 giây để suy nghĩ, thì sớm muộn cũng tự động hóa

Supervised Learning

Input (A)	Output (B)	Application
email	spam? (0/1)	spam filtering
audio	text transcripts	speech recognition
English	Chinese	machine translation
ad, user info	click? (0/1)	online advertising
image, radar info	position of other cars	Self-driving car
image of phone	defect? (0/1)	visual inspection

Anything you can do with 1 second of thought,
we can probably now or soon automate.

HỌC MÁY CÓ THỂ VÀ KHÔNG THỂ LÀM!

Ví dụ: Bạn đặt mua trên mạng 1 món hàng để làm quà tặng cho cháu gái của bạn nhân dịp sinh nhật của cô bé. Không may, món hàng bị giao trễ. Bạn gọi điện thoại đến tổng đài CSKH và nói : “Tôi có thể trả lại không nhận được không?” Nếu tổng đài là 01 AI (Voice chat). Chuyện gì sẽ xảy ra?

“Yêu cầu hoàn tiền”

“Tôi xin lỗi phải nghe thấy vậy!”

Input text → Hoàn tiền/Phí giao hàng/
Một số yêu cầu khác

“Chúc cháu 1 ngày sinh nhật vui vẻ”
“Vâng, tôi có thể giúp gì với....”

A → B

NẾU BẠN TIẾP TỤC CỐ GẮNG?

Đầu vào ----->
(Người dùng email)

Đầu ra
(Cài đặt 2 -3 dòng tự động trả lời)

1000 ví dụ

“Cái hộp đựng của tôi bị bể?” ----->

Cám ơn bạn đã gửi email!

“Tôi có thể viết nhận xét ở đâu?”----->

Cám ơn bạn đã gửi email!

“Chính sách trả hàng là ntn?” ----->

Cám ơn bạn đã gửi email!

“Khi nào hàng của tôi được giao?”----->

Cám ơn bạn, chúng tôi sẽ...

What happens if you try?

Input (A)

User email



Output (B)

2-3 paragraph response

1000 examples

“My box was damaged.”



Thank you for your email.

“Where do I write a review?”



Thank you for your email.

“What’s the return policy?”



Thank you for your email.

“When is my box arriving?”



Thank yes now your....

CÁI GÌ GIÚP M/L TRỞ THÀNH ĐƠN GIẢN?

1. Học 1 vấn đề đơn giản? **<= 1 giây**
2. Cần rất nhiều dữ liệu. **A → B**

A.I – TRÍ TUỆ NHÂN TẠO LÀ GÌ ?

➤ *Những điều ML có thể và
không thể làm. Thêm ví dụ!*

Self-driving car

Can do



A → B

Cannot do



stop

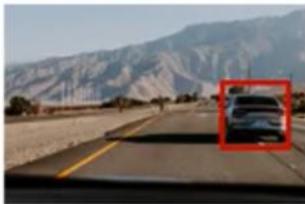
hitchhiker

bike turn
left signal

A → B

Self-driving car

Can do



A → B

10,000

Cannot do



stop

hitchhiker

10,000

1. Data
2. Need high accuracy

A → B

X-ray diagnosis



Can do

Diagnose pneumonia from
~10,000 labeled images

$$A \rightarrow B$$

Cannot do

Diagnose pneumonia from
10 images of a medical textbook
chapter explaining pneumonia

$$A \rightarrow B$$

ĐIỂM MẠNH – YẾU CỦA HỌC MÁY

HỌC MÁY CÓ THỂ LÀM RẤT TỐT KHI:

1. Học một khái niệm “đơn giản”.
2. Có sẵn nhiều dữ liệu.

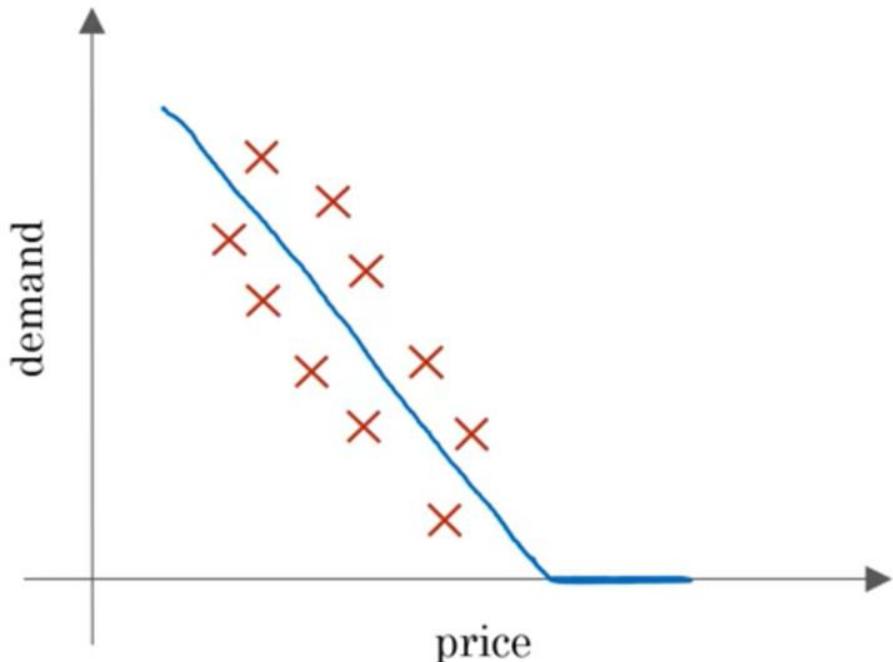
HỌC MÁY CÓ THỂ LÀ KÉM:

1. Học các khái niệm phức tạp với số lượng dữ liệu hạn chế.
2. Yêu cầu thực hiện một công việc cũ với kiểu dữ liệu mới, khác.

A.I – TRÍ TUỆ NHÂN TẠO LÀ GÌ ?

➤ *Học sâu (Deep Learning) Dành
cho Không chuyên Kỹ thuật*

Demand prediction

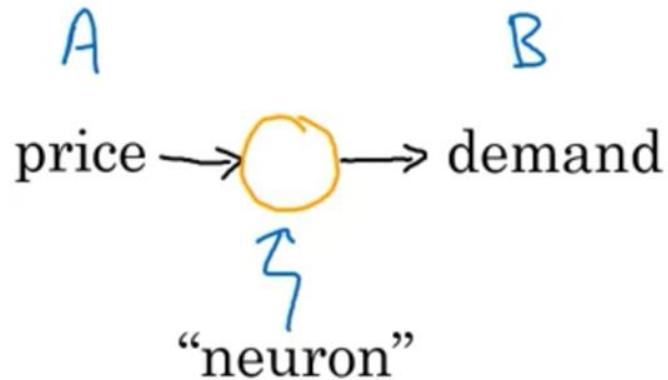
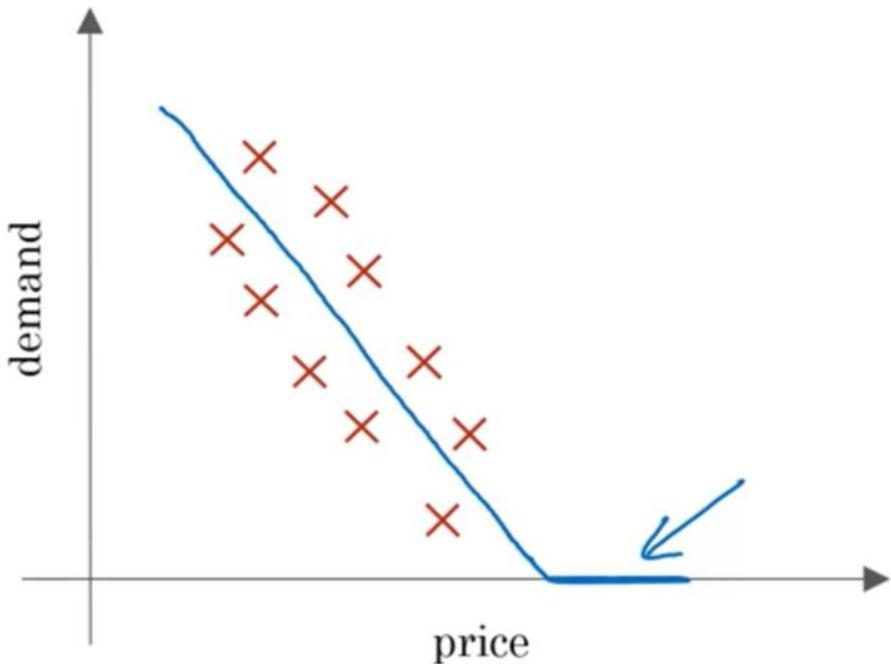


A
price

B
demand

DỰ ĐOÁN NHU CẦU NGƯỜI DÙNG

Demand prediction



DỰ ĐOÁN NHU CẦU NGƯỜI DÙNG

GIÁ BÁN (PRICE)

**PHÍ VẬN CHUYỂN
(SHIPPING COST)**

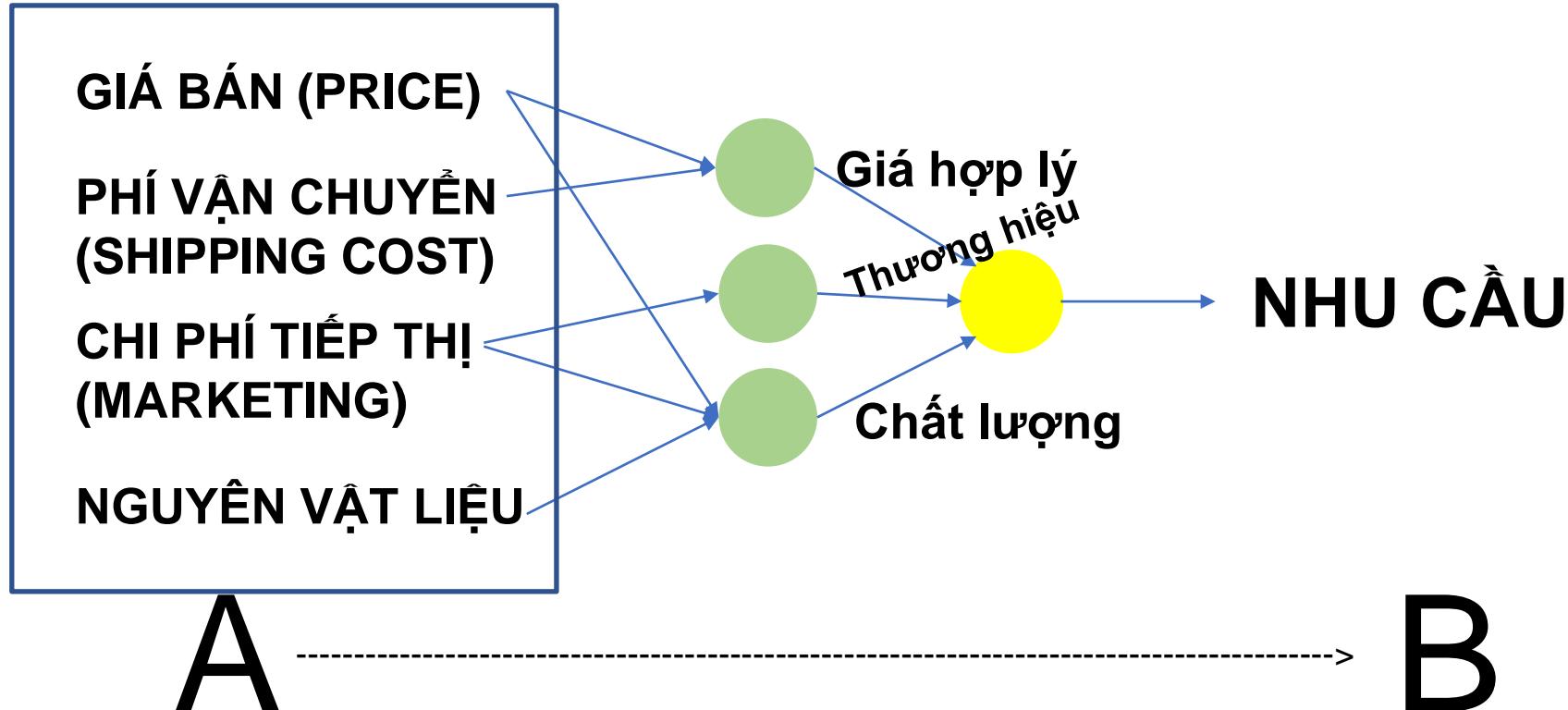
**CHI PHÍ TIẾP THỊ
(MARKETING)**

NGUYÊN VẬT LIỆU

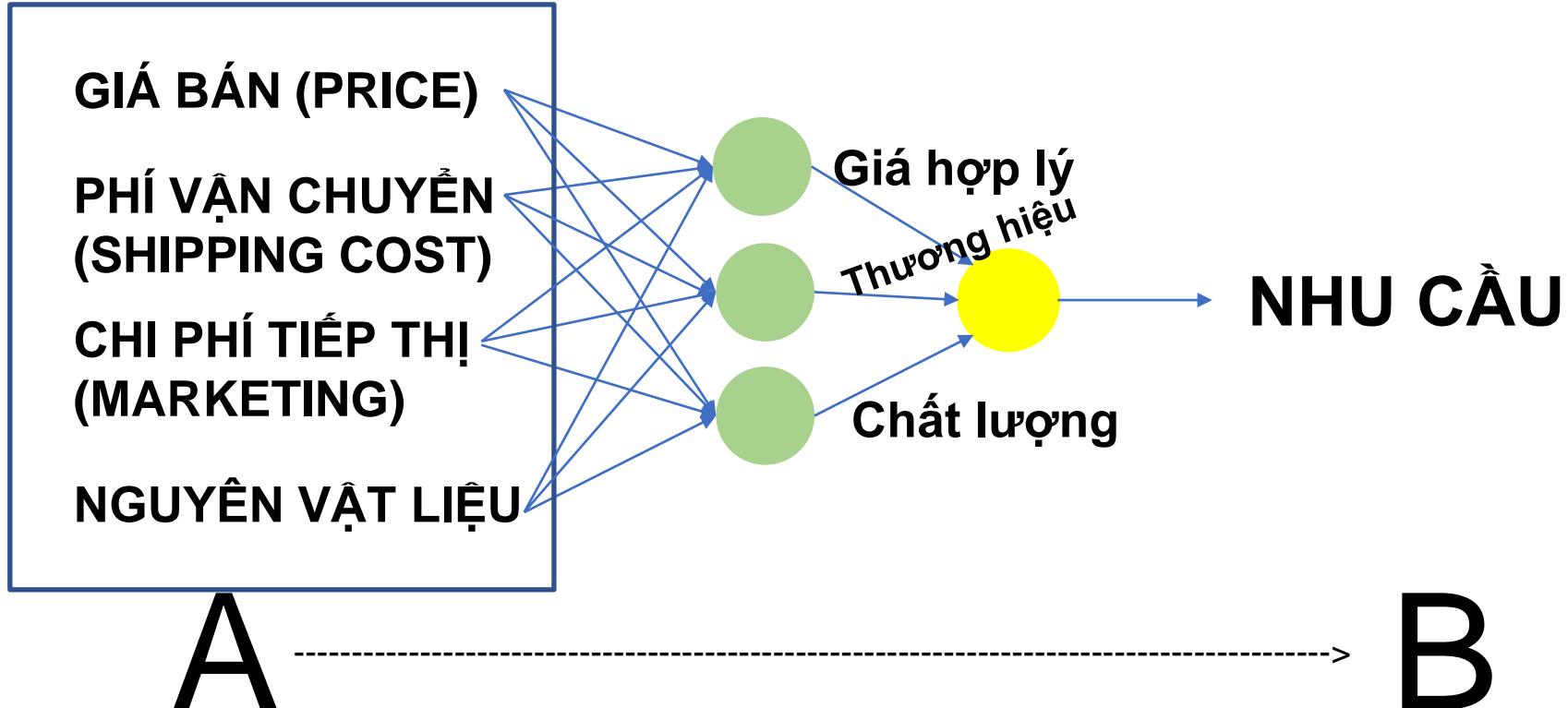


NHU CẦU

DỰ ĐOÁN NHU CẦU NGƯỜI DÙNG



DỰ ĐOÁN NHU CẦU NGƯỜI DÙNG



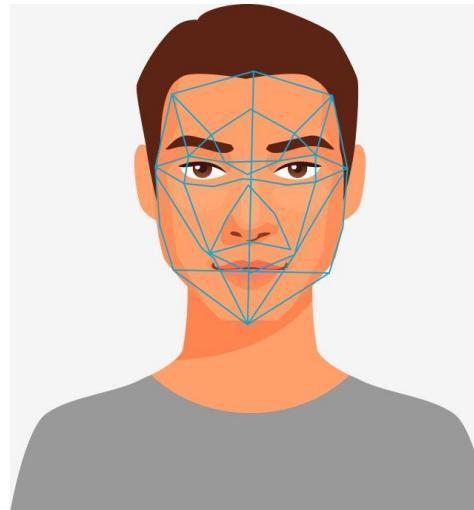
A.I – TRÍ TUỆ NHÂN TẠO LÀ GÌ ?

➤ *Học sâu (Deep Learning) Dành
cho Không chuyên Kỹ thuật*

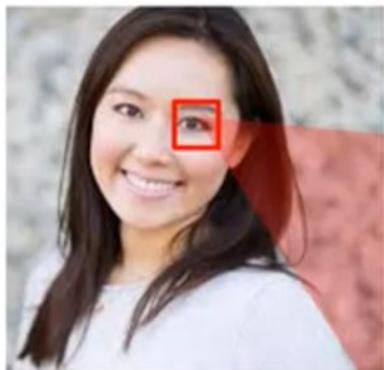
Face recognition



NHẬN DẠNG KHUÔN MẶT



Face recognition



30	32	22	12	10	10	12	33	35	30
12	11	12	234	170	176	13	15	12	12
234	222	220	230	200	222	230	234	56	78
190	220	186	112	110	110	112	180	30	32
49	250	250	250	4	2	254	200	44	6
55	250	250	250	3	1	250	245	25	3
189	195	199	150	110	110	182	190	199	55
200	202	218	222	203	200	200	208	215	222
219	215	220	220	222	214	215	210	220	220
220	220	220	220	221	220	221	220	220	222

Face recognition

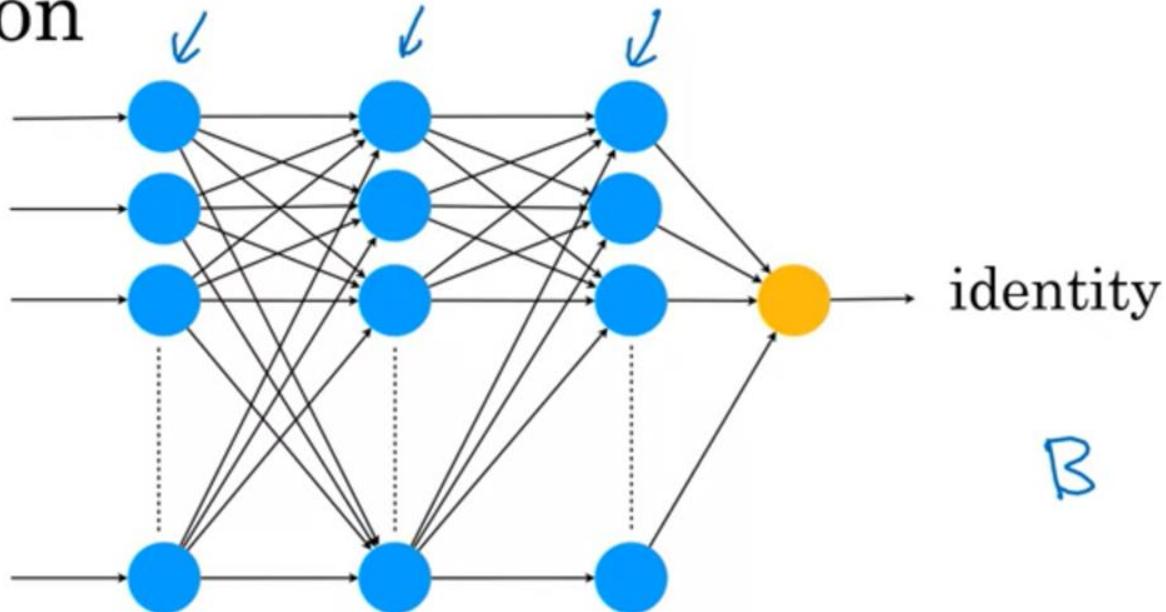
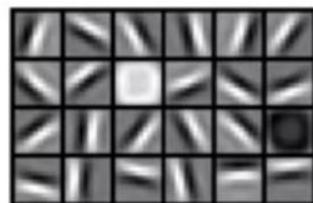


1000

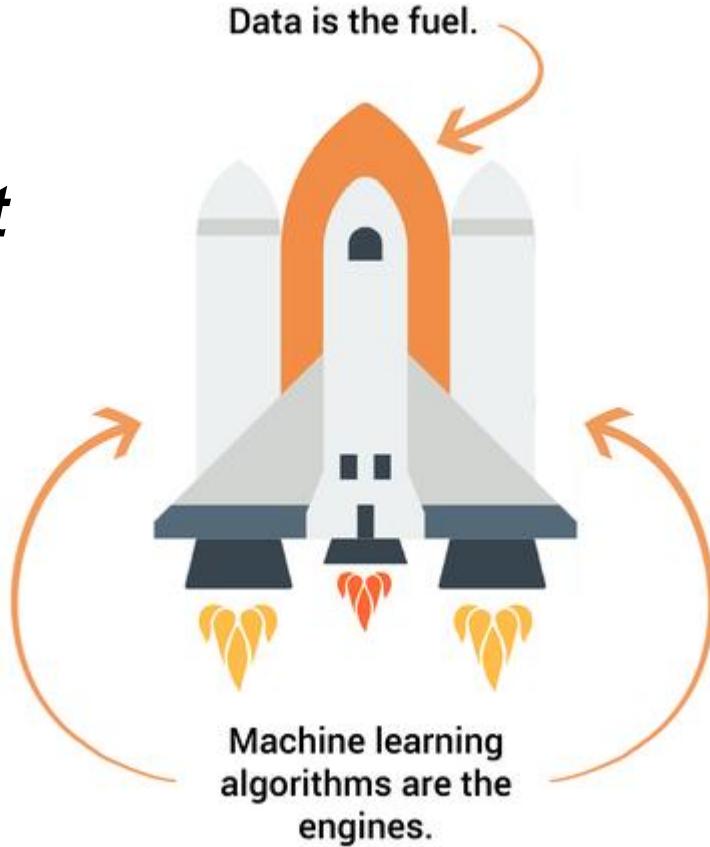
1000

1,000,000

3,000,000



H.I: Pilot



KIỂM TRA KIẾN THỨC

THỰC HIỆN 1 DỰ ÁN A.I

➤ **Giới thiệu**

THỰC HIỆN 1 DỰ ÁN A.I

- **Các bước thực hiện**
- **Lựa chọn 1 dự án**
- **Tổ chức đội ngũ và cơ sở dữ liệu cho dự án.**

THỰC HIỆN DỰ ÁN NHẬN GIỌNG NÓI



Amazon
Echo / Alexa



Google
Home



Apple
Siri



Baidu
DuerOS

Các bước chính:

1. Thu thập dữ liệu.



Hey
Google...



Các bước chính:

2. Huấn luyện mô hình (Model).

Lặp đi, lặp lại cho
đến khi đủ tốt.

A → B

#Giọng nói 1 --- “Hello”

#Giọng nói 2 ----”Siri”

#Giọng nói 3 – “Âm nhạc”

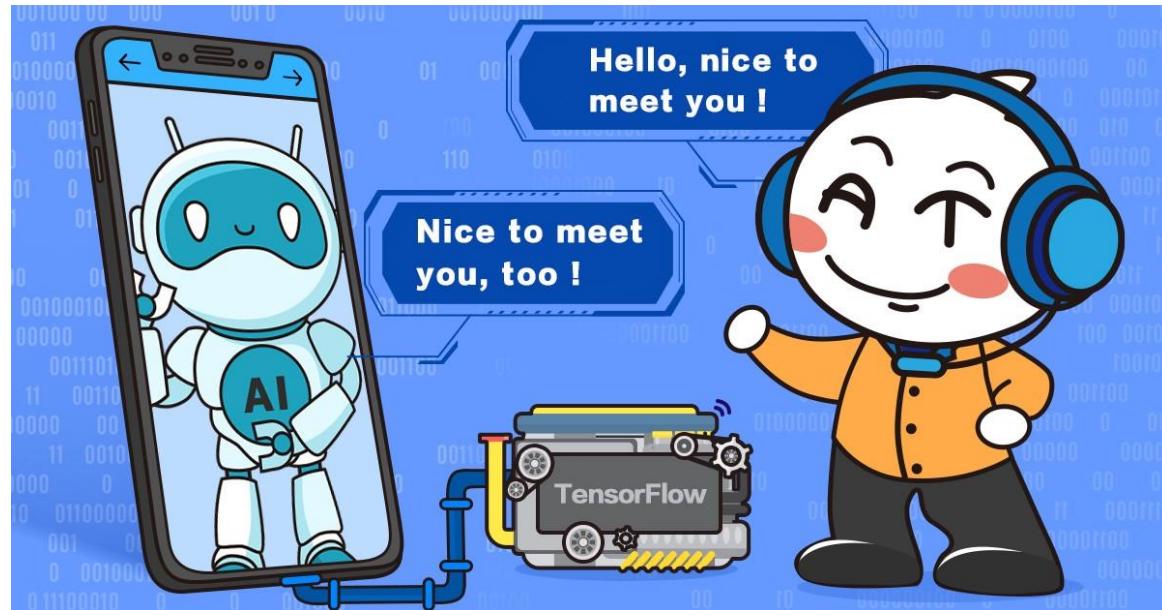


Các bước chính:

3. Cài đặt (Deploy).

Sử dụng các dữ liệu đã
có sẵn.

Duy trì và cập nhật mô
hình (model)



Key steps of a machine learning project

Echo / Alexa

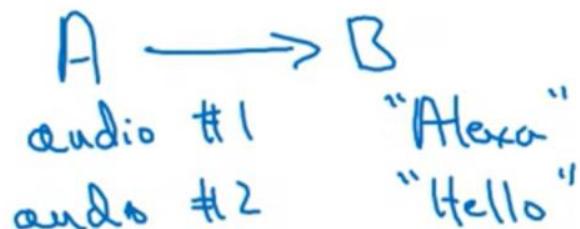
→ 1. Collect data

→ 2. Train model

Iterate many times until
good enough

→ 3. Deploy model

Get data back
Maintain / update model



Key steps of a machine learning project

Self-driving car

1. Collect data

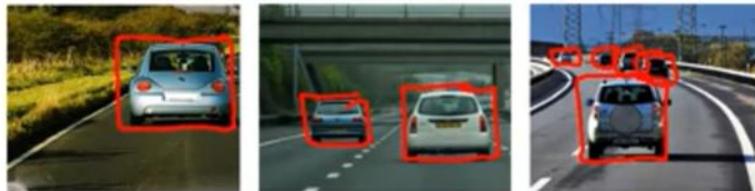


image → position of other cars

2. Train model

Iterate many times until
good enough



3. Deploy model

Get data back
Maintain / update model



THỰC HIỆN 1 DỰ ÁN A.I

➤ **Các bước thực hiện dự án Khoa học dữ liệu**

VISIT
WEBSITE

Ghé Trang web



TIKNOW Ly Sứ Cao Cấp Minh Tiến A11 (8cm) - Hồng Phấn 59.000 ₫ -25%
29.000 ₫



TIKNOW Ly Sứ Moritalia B106-WB (400ml) 40.000 ₫ -57%
18.000 ₫
★★★★★ (13 nhận xét)



Cốc Sứ Elmich Smart Cook SM8864 - 2026864 52.000 ₫
★★★★★ (9 nhận xét)



Ly Sứ Cao Cấp Dong Hwa Hình Viên Đầu Chân... 39.000 ₫ -41%
66.000 ₫
★★★★★ (1 nhận xét)



Q Rê chuột lên hình để phóng to



Bộ 3 Ly Sứ Cao Cấp Sọc Ngang Dong Hwa C1408... 94.000 ₫ -48%
48.000 ₫
★★★★★ (4 nhận xét)



Cốc Vuông Sứ Sương GSSBC4A6 (0.18L) - Xanh... 72.500 ₫



Cốc Vuông Sứ Sương GSSBC4A6 (0.18L) - Xanh... 72.500 ₫
Ly Sứ Mờ Mờ 12 Dung Hoàng Đạo 154.000 ₫ -39%
99.000 ₫
★★★★★ (4 nhận xét)



Ly Sứ Sola Nắp Trượt (Màu Ngẫu Nhiên) 149.000 ₫ -25%
109.000 ₫

Cốc Vuông Sứ Sương GSSBC4A6 (0.18L) -

Thương hiệu: Sứ Sương SKU: 2131143255182

72.500 ₫

- Cốc (tách) dáng vuông
- Có kích cỡ hoàn hảo để uống cà phê
- Chất liệu đất sét trắng

Số lượng:

- 1 +

CHỌN MUA



THỜI GIAN DỰ KIẾN GIAO HÀNG

Giao hàng tới Phường Bến Nghé, Quận 1, Hồ Chí Minh - Chọn địa chỉ khác

Q Giao hàng tiêu chuẩn Dự kiến giao Thứ năm - Thứ sáu, 15/08 - 16/08 chỉ phí 12.000 ₫, miễn phí giao hàng tiêu chuẩn cho đơn hàng từ 250.000 ₫

Chọn mua sản phẩm

GIỎ HÀNG (1 sản phẩm)



Cốc Vuông Sứ Sương GSSBC4A6 (0.18L) - Xanh Ngọc

Cung cấp bởi [Gốm Sứ Sương](#)

Xóa

Để dành mua sau

72.500 ₫

- 1 +

Tạm tính:

72.500 ₫

Thành tiền:

72.500 ₫

(Đã bao gồm VAT)

Tiến hành đặt hàng

Mã giảm giá / Quà tặng

Nhập ở đây..

Đồng ý

THANH TOÁN

1. Chọn hình thức giao hàng

Thứ sáu, 16/08/2019
12.000 ₫ Giao hàng tiêu chuẩn

2. Thanh toán bằng Tiki Xu

Sử dụng 2.689 Xu (ứng với 2.689 ₫) để thanh toán.
Bạn có Phiếu quà tặng Tiki/Got It/UBox muốn đổi thành Tiki Xu? Nhập tại đây

3. Chọn hình thức thanh toán

Thanh toán tiền mặt khi nhận hàng

Thanh toán bằng thẻ quốc tế Visa, Master, JCB

Thẻ ATM nội địa/Internet Banking (Miễn phí thanh toán)

Thanh toán bằng ví MoMo - [Chi tiết](#)

Vui lòng cài đặt app MoMo để thanh toán!

ưu đãi Thẻ OCB giảm 150K cho đơn hàng từ 1 triệu đồng - [Chi tiết](#)

Chưa đến thời gian tham gia chương trình hoặc số lượng đơn hàng ưu đãi đã hết!

ưu đãi Thẻ tín dụng JCB giảm thêm 25% tối đa 300K - [Chi tiết](#)

Chưa đến thời gian tham gia chương trình hoặc số lượng đơn hàng ưu đãi đã hết!

ưu đãi Thẻ Visa giảm 50K cho đơn hàng từ 500K - [Chú ý](#)

Chưa đến thời gian tham gia chương trình hoặc số lượng đơn hàng ưu đãi đã hết!

Gửi quà tặng đến bạn bè, người thân

Thông tin người mua:

Sử dụng Họ tên & Số điện thoại của địa chỉ giao hàng

ĐẶT MUA

Địa chỉ giao hàng

Nguyễn Ngọc TÚ

Landmark 3, Vinhomes TÂN CĂNG, Số 208
NGUYỄN HỮU CÀNH, Phường 22, Quận
Bình Thạnh, Hồ Chí Minh
Việt Nam
Điện thoại: 0903686860

Đơn Hàng (1 sản phẩm)

1x Cốc Vuông Sứ Sương
GSSBC4A6 (0.18L) - Xanh Ngọc
Cung cấp bởi Gốm Sứ Sương

Tạm tính 72.500 ₫
Phí vận chuyển 12.000 ₫

Thành tiền: 84.500 ₫
(Đã bao gồm VAT)

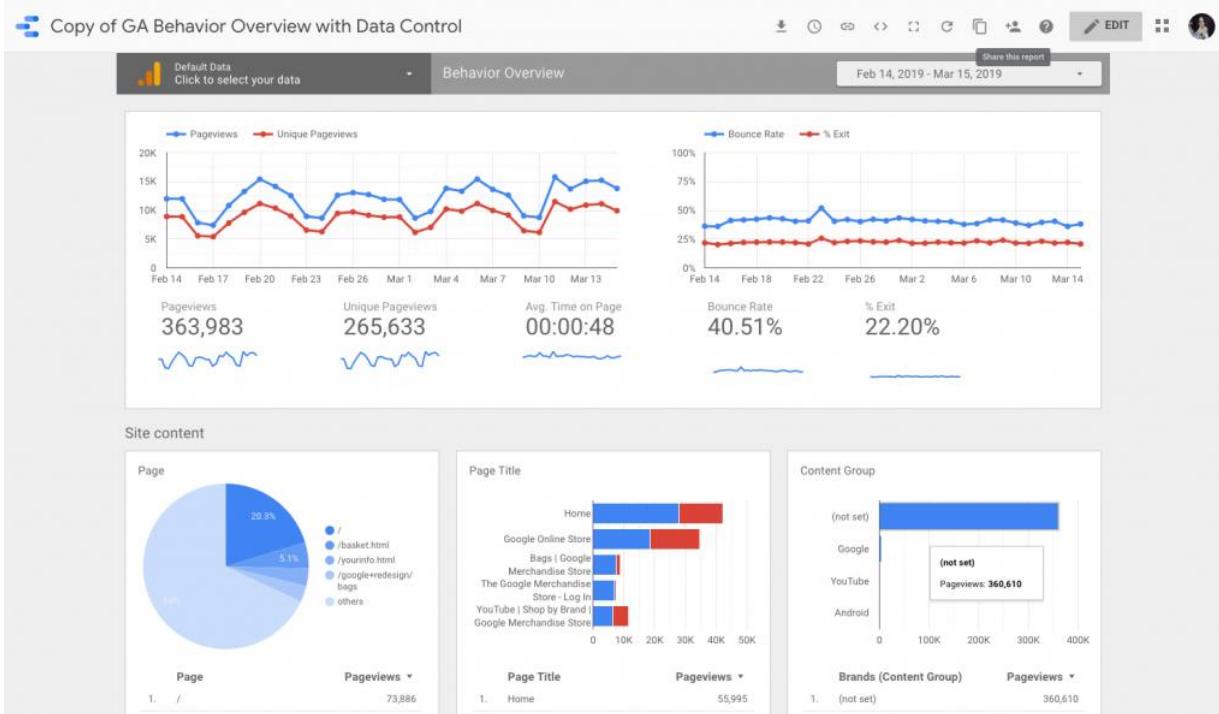
Các bước chính:

1. Thu thập dữ liệu.

User ID	Khu vực	Thời gian	Website
2009	HCM	08:03:15 Jan 19	Tiki.vn
2017	HN	14:05:00 Dec 18	Shoppe.vn
4963	Singapore	09:00:15 Mar 19	Amazon.com

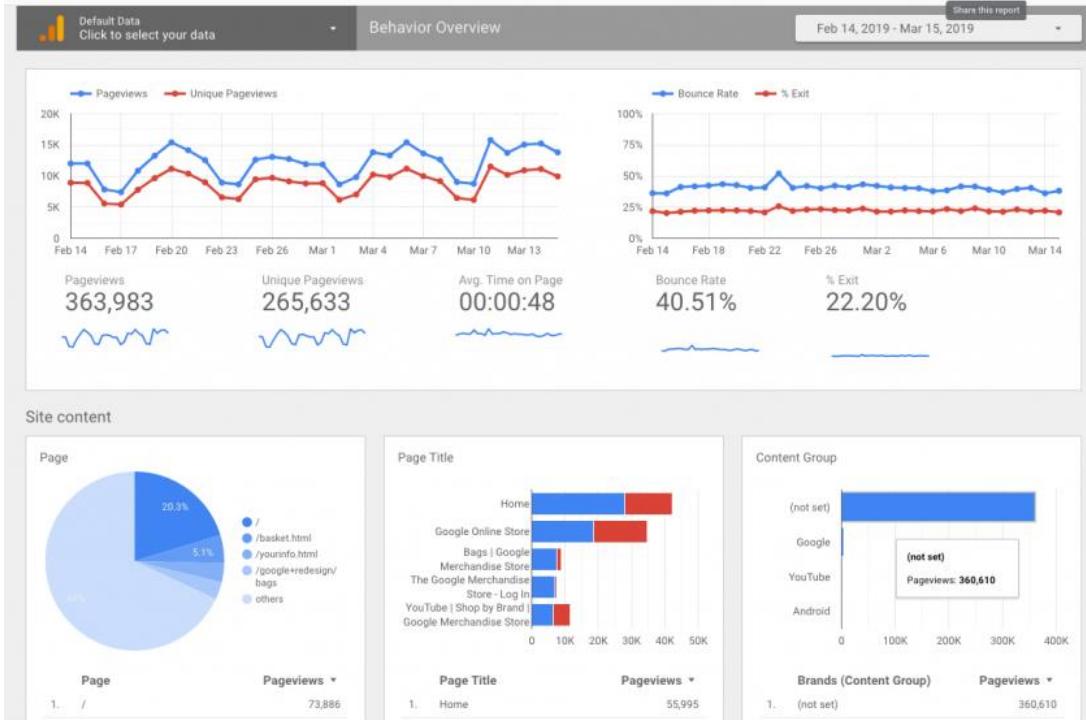
Các bước chính:

2. Phân tích dữ liệu. (Thử đi, thử lại)



Các bước chính:

2. Phân tích dữ liệu. (Thử đi, thử lại)

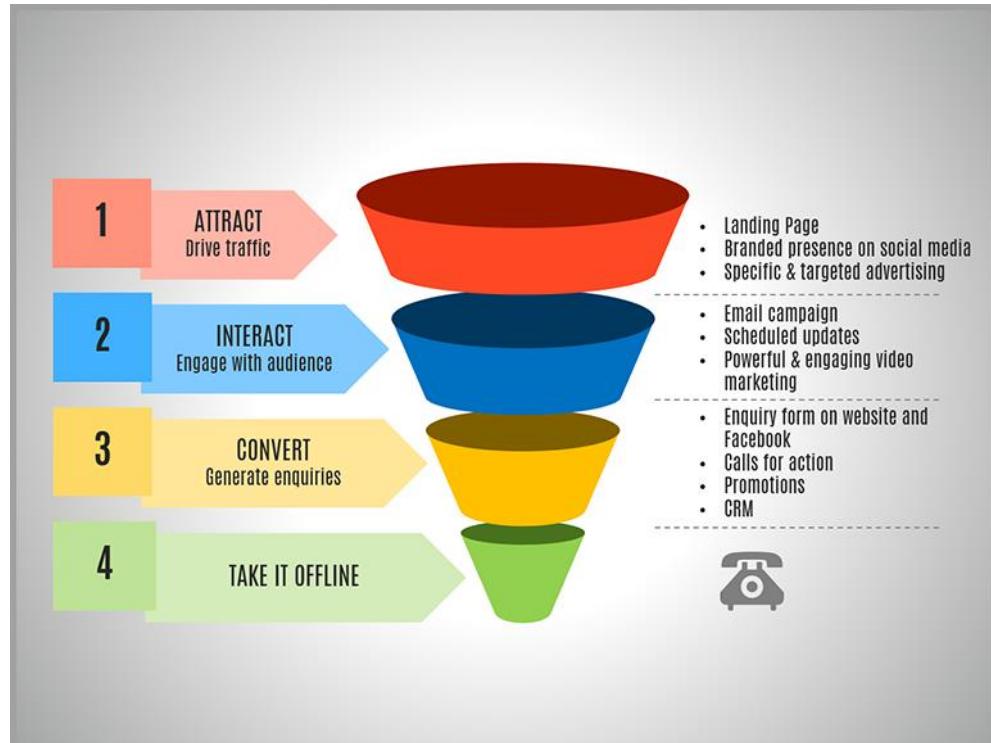


Các bước chính:

2. Đề xuất các hành động

Tích hợp các thay đổi
(Xây mới trang web,
thêm video quảng cáo,
Khuyến mãi...)

Tiếp tục phân tích với
dữ liệu mới.



Key steps of a data science project

Manufacturing line

Mix clay Shape mug Add glaze Fire kiln Final inspection



Key steps of a data science project

Manufacturing line



Clay Batch #	Supplier	Mixing time (minutes)
001	ClayCo	35
034	GooClay	22
109	BrownStuff	28

1. Collect data
2. Analyze data
 - Iterate many times to get good insights
3. Suggest hypotheses/actions
 - Deploy changes
 - Re-analyze new data periodically

Mug Batch #	Humidity	Temperature in kiln (F)	Duration in kiln (hours)
301	0.002%	1410°	22
302	0.003%	1520°	24
303	0.002%	1420°	22

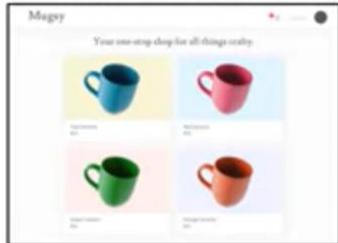
THỰC HIỆN 1 DỰ ÁN A.I

➤ ***Mọi vị trí công việc cần
biết cách sử dụng Dữ liệu***

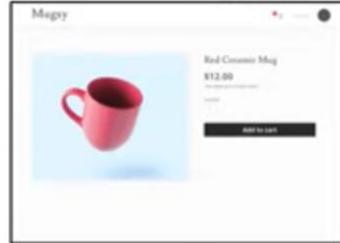
BỘ PHẬN BÁN HÀNG

Data science

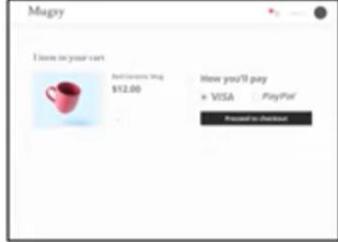
Visit website



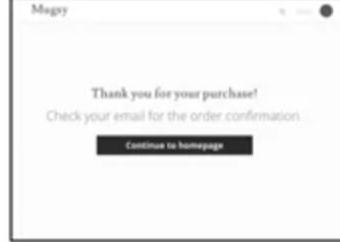
Product page



Shopping cart



Checkout



Optimize sales funnel

Machine learning

Name	Title	Company size	Email	Priority
Taylor	CEO	3050	tay@a..	high
Janet	Manager	230	jan@b..	medium
David	Intern	30	dave@c..	low

Automated lead sorting

BỘ PHẬN QUẢN LÝ SẢN XUẤT Ở NHÀ MÁY

Data science



Machine learning



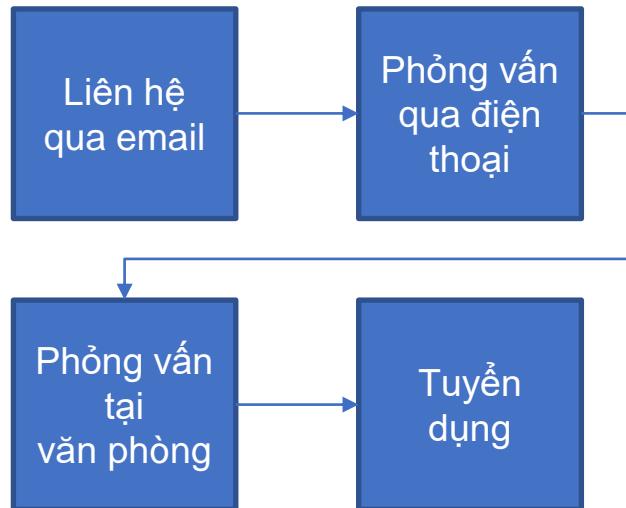
ok

ok

defect

Automated visual inspection

BỘ PHẬN TUYỂN DỤNG NHÂN SỰ



THÚY KIỀU	
Thông tin cá nhân	
Học vấn	ĐẠT
Chuyên môn	
Kinh nghiệm	

HOẠN THƯ	
Thông tin cá nhân	
Học vấn	CHƯA ĐẠT
Chuyên môn	
Kinh nghiệm	

Tối ưu hóa quy trình tuyển dụng

Tự động hóa quá trình đọc SYLL

BỘ PHẬN MARKETING

Data science

Break into AI

Whether you want to build algorithms or build a company, deeplearning.ai's courses will teach you key concepts and applications of AI.

[Take the Deep Learning Specialization](#)

A

Break into AI

Whether you want to build algorithms or build a company, deeplearning.ai's courses will teach you key concepts and applications of AI.

[Take the Deep Learning Specialization](#)

B

A/B testing

Machine learning

Recommended for you



Button Shirt
\$60



Button Shirt
\$60



Button Shirt
\$60



Button Shirt
\$60

Customized product recommendation

NGÀNH NÔNG NGHIỆP

Data science



Crop analytics

Machine learning



Precision weed killing

AI technical tools

Machine learning frameworks:

- TensorFlow
- PyTorch
- Keras
- MXNet
- CNTK
- Caffe
- PaddlePaddle
- Scikit-learn
- R
- Weka

AI technical tools

Machine learning frameworks:

- TensorFlow
- PyTorch
- Keras
- MXNet
- CNTK
- Caffe
- PaddlePaddle
- Scikit-learn
- R
- Weka

Research publications:

- Arxiv

Open source repositories:

- GitHub

CPU vs. GPU

CPU: Computer processor (Central Processing Unit)



GPU: Graphics Processing Unit



Cloud vs. On-premises

Edge

XÂY DỰNG MỘT CÔNG TY A.I

➤ ***Trường hợp: Loa thông minh***

LOA THÔNG MINH



Amazon
Echo / Alexa



Google
Home



Apple
Siri



Baidu
DuerOS

“Hey Google, Kể 1 câu truyện cười”

“Hey Google, Kể 1 câu truyện cười”

Các bước để thực hiện lệnh:

1. Xác nhận “Khâu lệnh khởi động”

“*Audio*” → “*Hey Google*” (0/1)

2. Nhận dạng giọng nói.

“*Audio*” → “*Kể một câu chuyện cười*” (0/1)

3. Nhận dạng câu lệnh/ý định.

“*Kể một câu chuyện cười*”

A

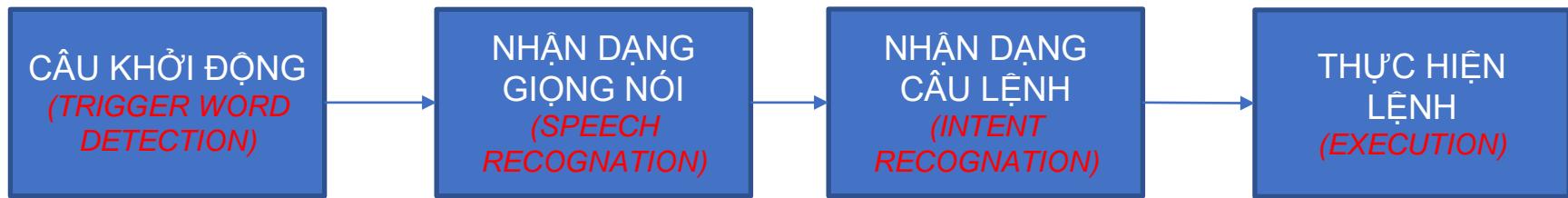
“*Thời tiết*”
“*Âm nhạc*”
“*Thực hiện cuộc gọi*”

B

“Hey Google, Kể 1 câu truyện cười”

Các bước để thực hiện lệnh:

4. Thực hiện kể chuyện



**QUÁ TRÌNH THỰC HIỆN CÔNG VIỆC CỦA A.I
(A.I PIPELINE)**

XÂY DỰNG MỘT CÔNG TY A.I

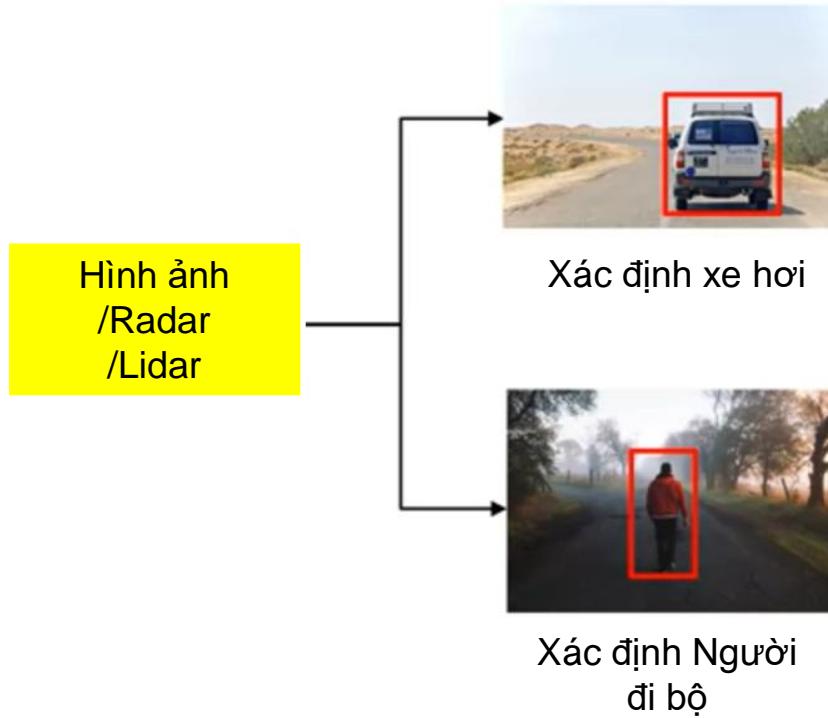
- ***Trường hợp: Loa thông minh
– Thêm ví dụ***

XÂY DỰNG MỘT CÔNG TY A.I

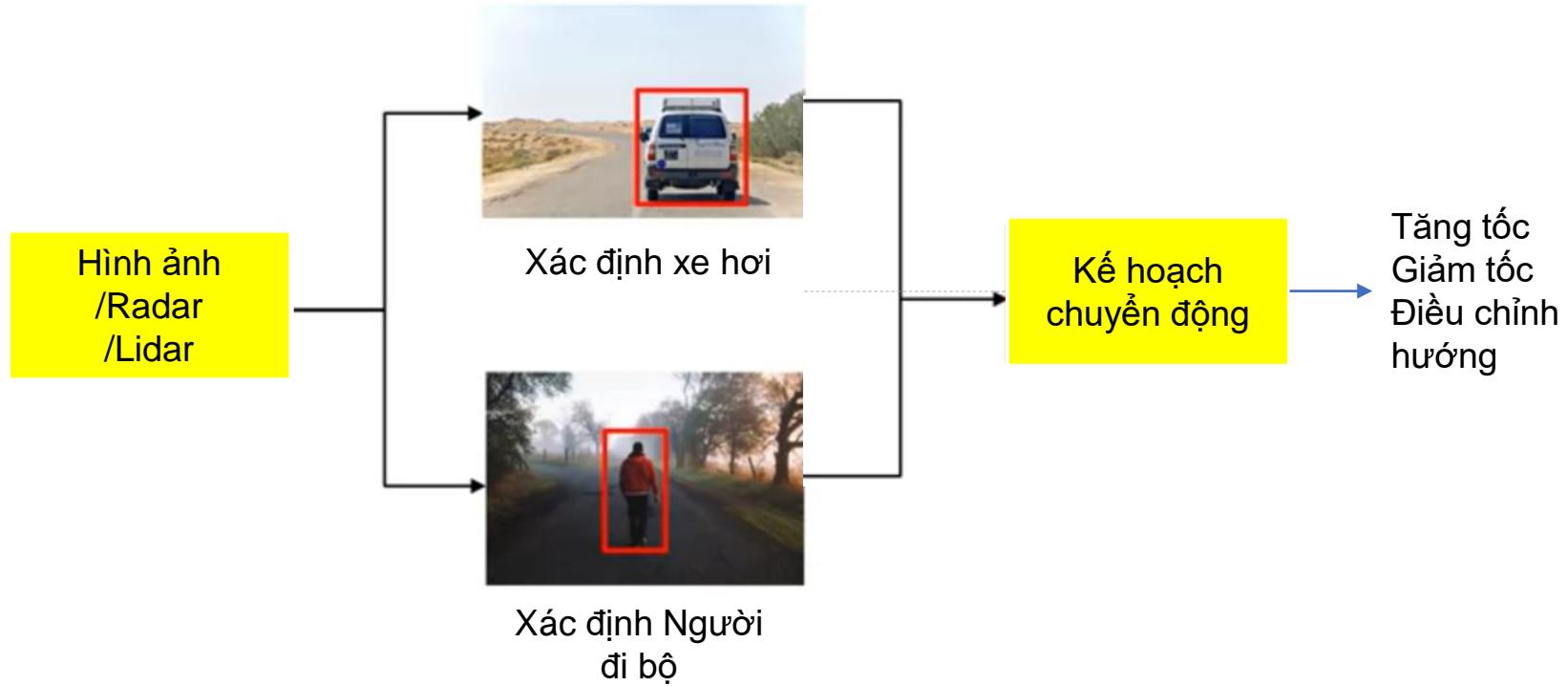
➤ ***Trường hợp: Xe tự lái***



“Các bước ra quyết định lái xe”

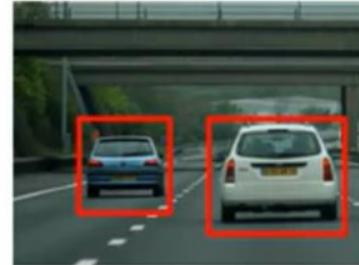


“Các bước ra quyết định lái xe”



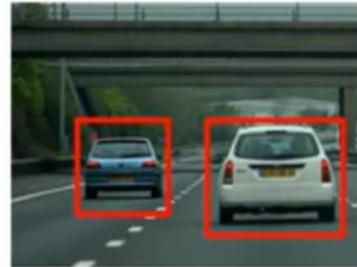
Key steps:

1. Car detection



Key steps:

1. Car detection

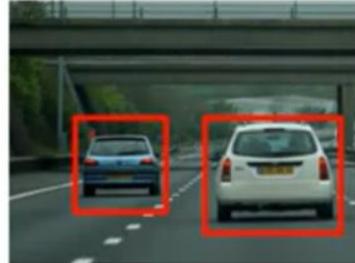


2. Pedestrian detection

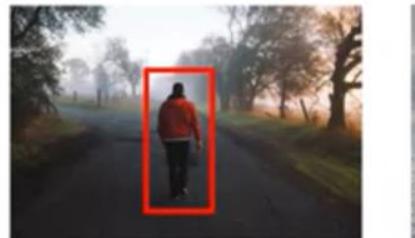


Key steps:

1. Car detection



2. Pedestrian detection

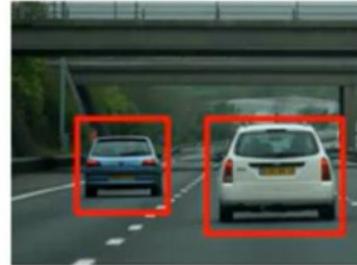


3. Motion planning

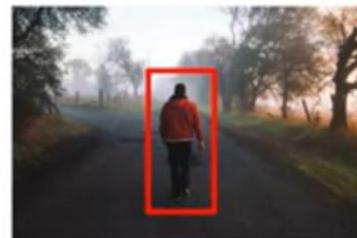


Key steps:

1. Car detection



2. Pedestrian detection



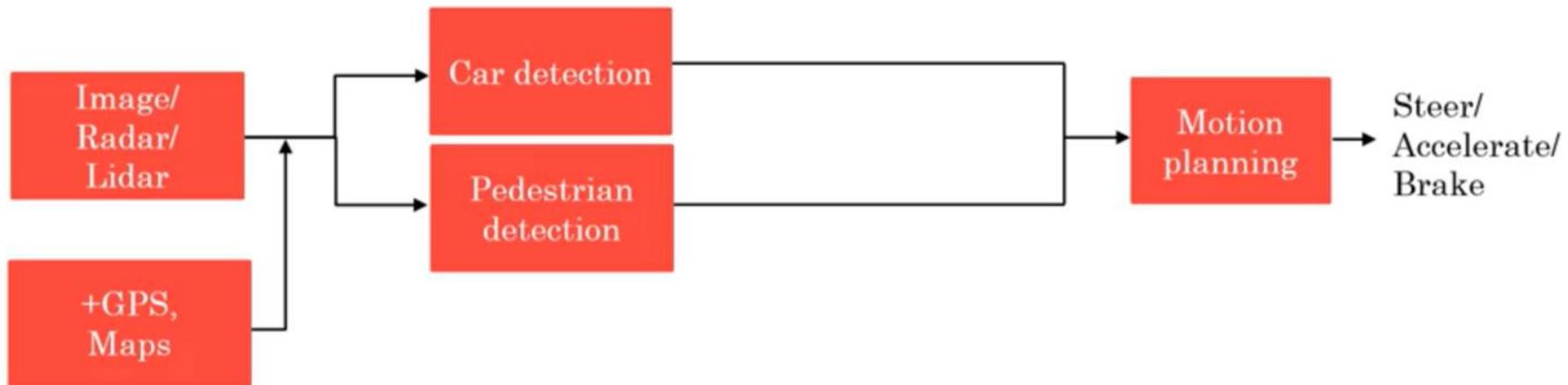
3. Motion planning



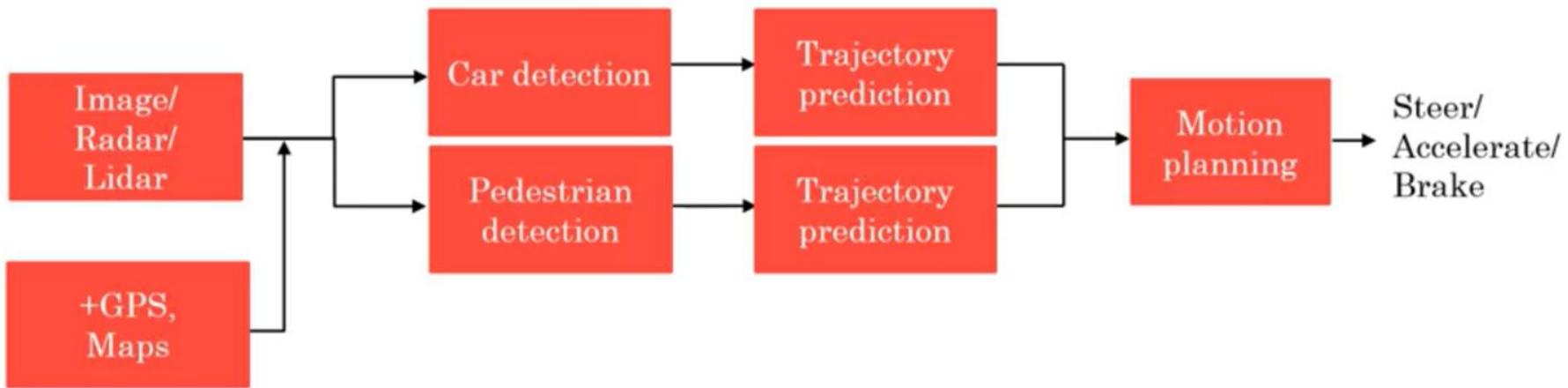
Steps for deciding how to drive



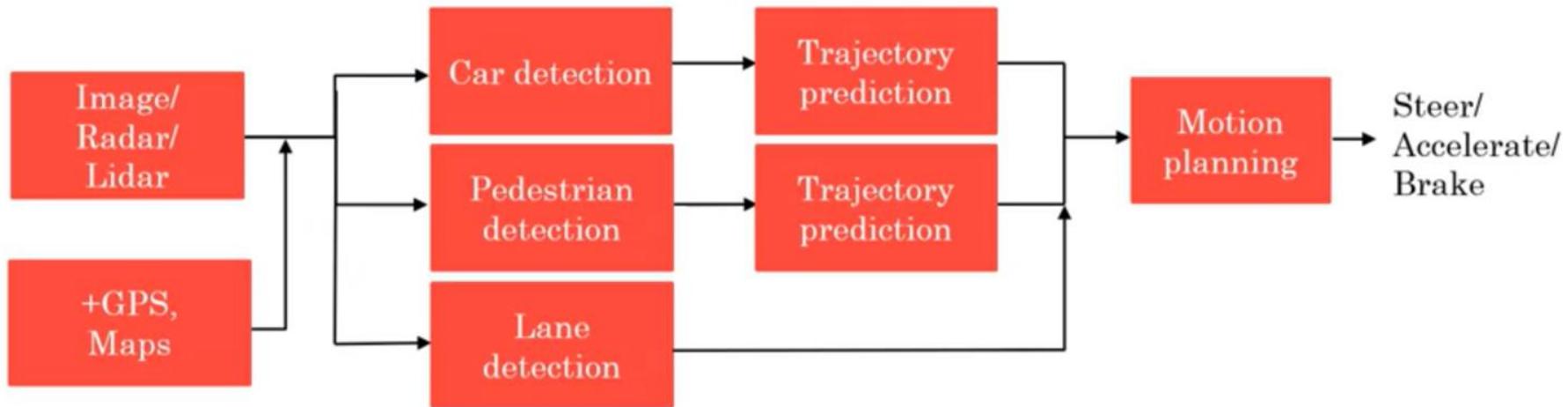
Steps for deciding how to drive



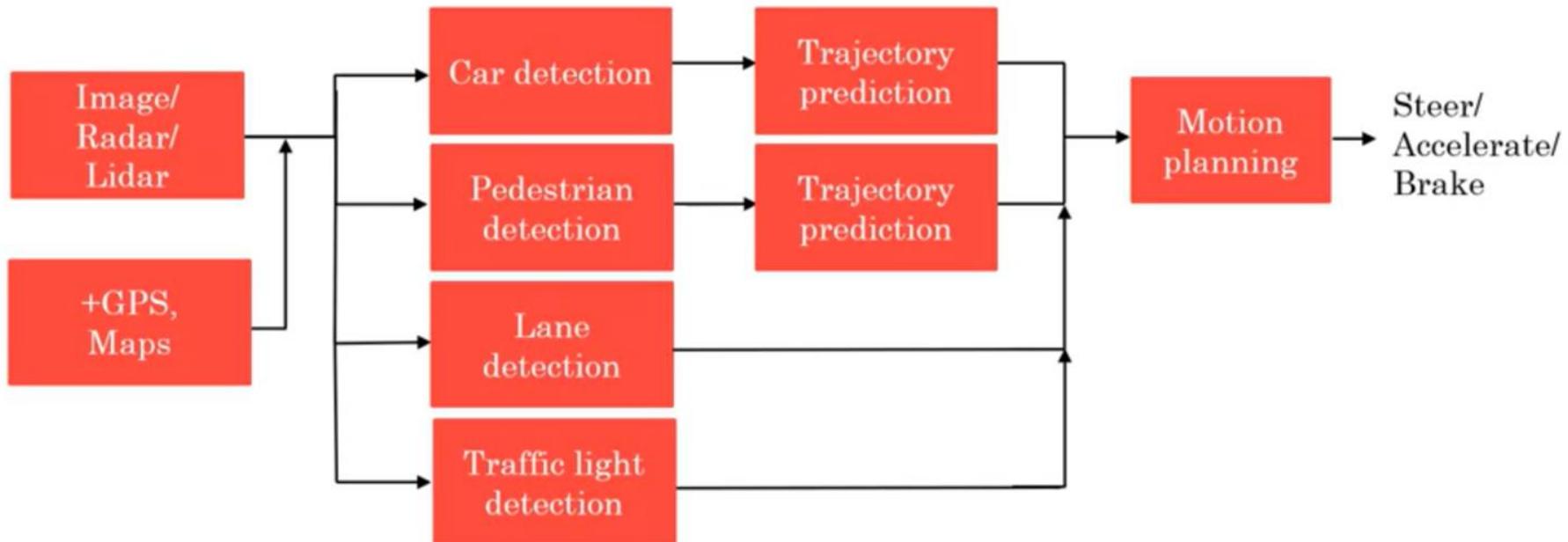
Steps for deciding how to drive



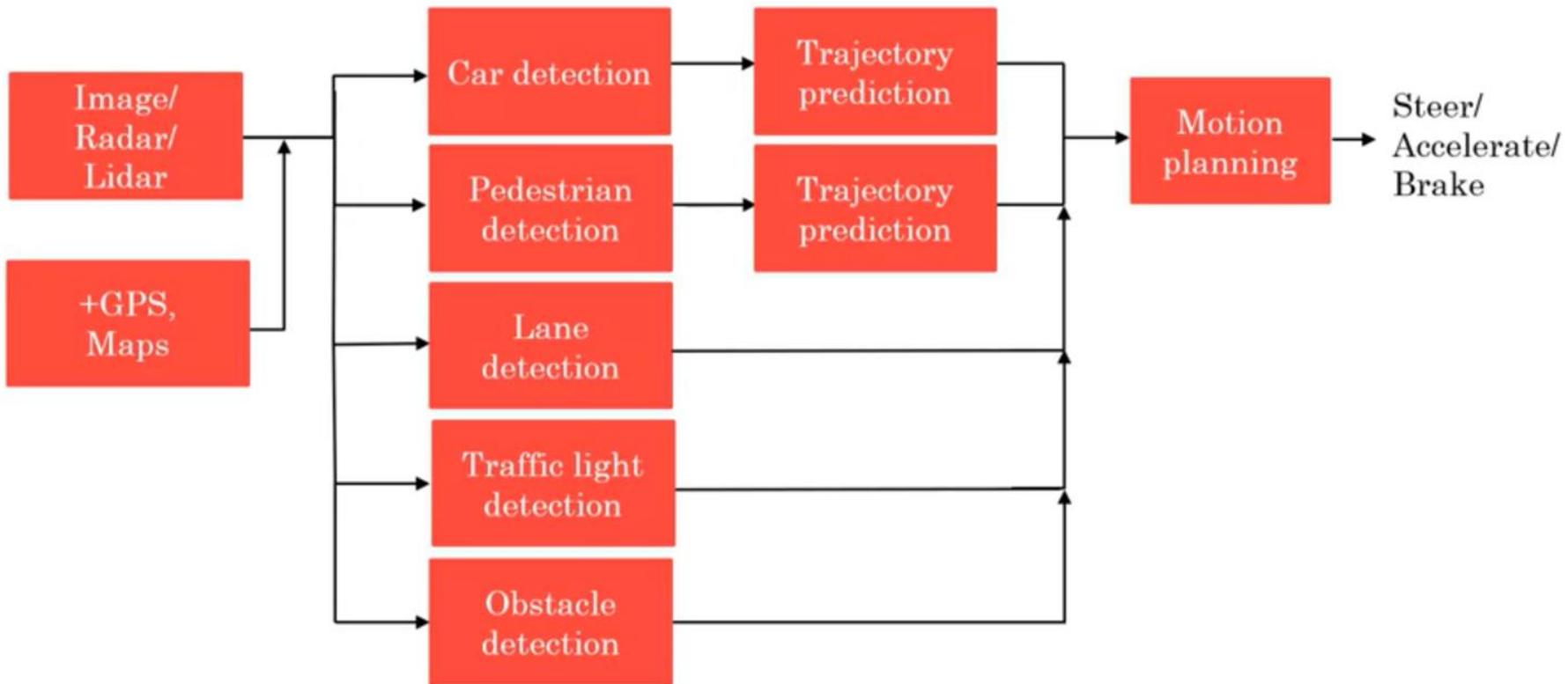
Steps for deciding how to drive



Steps for deciding how to drive



Steps for deciding how to drive



XÂY DỰNG MỘT CÔNG TY A.I

➤ ***Vai trò trong Nhóm AI***

Các vai trò điển hình

1. Kỹ sư Phần mềm – Software Engineer

“VD: Thực hiện lệnh kể chuyện, đảm bảo độ chính xác khi xe tự lái.”

2. Kỹ sư Học máy (Machine Learning Engineer)

“A → B”

3. Nhà nghiên cứu Học Máy – ML Researcher

“Tăng cường khả năng của ML, tối ưu các thuật toán”

Vị trí (2) & (3) có thể gọi là: Nhà khoa học Ứng dụng ML

Các vai trò điển hình

4. Chuyên gia Dữ liệu (Data Scientist)

Phân tích dữ liệu và cung cấp các tri thức (insights).

Tạo các báo cáo cho Nhóm

5. Kỹ sư Dữ liệu (Data Engineer)

Tổ chức dữ liệu

Thực hiện công việc để cho Dữ liệu được lưu trữ đúng cách, an toàn, dễ dàng thực hiện công việc.

6. Quản lý/Giám đốc sản phẩm AI – AI Product Manager

Hỗ trợ, đưa ra quyết định nên làm gì, xác định khả năng, giá trị sp.

BẮT ĐẦU VỚI MỘT ĐỘI A.I NHỎ

1 Kỹ sư Dữ liệu, hoặc

1 Kỹ sư Học máy (Machine Learning Engineer), 1 Chuyên gia dữ liệu
hoặc

Chẳng có ai**ngoài bạn** !

XÂY DỰNG MỘT CÔNG TY A.I

➤ ***Những thứ cần tránh***

CẦN TRÁNH TRONG A.I

ĐỪNG

- 1. A.I sẽ giải quyết hết được mọi việc/ vấn đề**
- 2. Thuê một vài (2-3) Kỹ sư ML và giao phó hết trách nhiệm cho họ, với mong muốn có được các ứng dụng tốt.**

NÊN

- 1. Thực tế AI có thể làm và không thể làm gì. Các giới hạn của Công nghệ, dữ liệu và cả tài nguyên kỹ thuật.**
- 2. Xem xét các nguồn lực về nhân sự thấu đáo. Giữa Kỹ thuật và Kinh tế và tìm ra các lựa chọn “khả thi, giá trị”.**

CẦN TRÁNH TRONG A.I

ĐỪNG

- 3. A.I sẽ hoạt động tốt ngay và luôn.**
- 4. Duy trì các kế hoạch cũ, ứng dụng công nghệ mới mà không cần đến sự thay đổi.**
- 5. Cần ngay 1 siêu sao AI trước khi bạn có thể thực hiện mọi thứ.**

NÊN

- 3. Dành thời gian cho Đội AI thực hiện các thử nghiệm, điều chỉnh cho đến khi thành công.**
- 4. Làm việc với đội AI để lên kế hoạch thời gian, phân bổ lại nguồn lực, xây dựng các thước đo mới, những mục tiêu mới.**

XÂY DỰNG MỘT CÔNG TY A.I

➤ ***Thực hiện bước đầu tiên***

A.I VÀ CUỘC SỐNG XÃ HỘI

➤ **Giới thiệu**

A.I VÀ CUỘC SỐNG XÃ HỘI

AI VÀ NHỮNG CÁCH HIỂU CHUNG

GIỚI HẠN CỦA AI

AI VÀ PHÁT TRIỂN KINH TẾ, CÔNG VIỆC

A.I VÀ CUỘC SỐNG XÃ HỘI

➤ **Nhìn nhận thực tế về AI**

NHỮNG CÁCH HIỂU CHUNG VỀ AI

KHÔNG QUÁ LẠC QUAN VỀ AI: TRÍ THÔNG MINH NHÂN TẠO SIÊU VIỆT SẼ XUẤT HIỆN. ROBOT SÁT THỦ SẼ SỚM RA ĐỜI

KHÔNG QUÁ BI QUAN VỀ AI: TRÍ THÔNG MINH NHÂN TẠO KHÔNG LÀM ĐƯỢC GÌ. CHỈ LÀ TRÀO LƯU. SẼ SỚM BỊ TÀN

CHỈ CẦN HIỂU ĐÚNG: TRÍ TUỆ NHÂN TẠO KHÔNG THỂ LÀM ĐƯỢC MỌI THỨ, NHƯNG NÓ CÓ KHẢ NĂNG GIÚP THAY ĐỔI NHIỀU LĨNH VỰC, NHIỀU NGÀNH CÔNG NGHIỆP.

GIỚI HẠN CỦA AI

AI CÒN NHIỀU HẠN CHẾ. NĂNG LỰC THỰC HIỆN CÓ GIỚI HẠN

GIẢI THÍCH TẠI SAO AI LẠI ĐƯA RA 1 KẾT QUẢ LÀ KHÓ KHĂN



Right-sided
Pneumothorax
(collapsed lung)



GIỚI HẠN CỦA AI

BIASED AI (ĐỊNH KIẾN AI) VÌ NHỮNG ĐỊNH KIẾN VỀ DỮ LIỆU

GIẢI THÍCH TẠI SAO AI LẠI ĐƯA RA 1 KẾT QUẢ LÀ KHÓ KHĂN

A.I VÀ CUỘC SỐNG XÃ HỘI

➤ ***Những cách dùng AI gây hại***

NHỮNG CÁCH DÙNG AI GÂY HẠI

DeepFakes

Cắt ghép hình ảnh, phim và gán cho người khác.

PHÁ HỎNG QUYỀN RIÊNG TƯ VÀ DÂN CHỦ

Theo dõi, kiểm soát bằng các ứng dụng AI

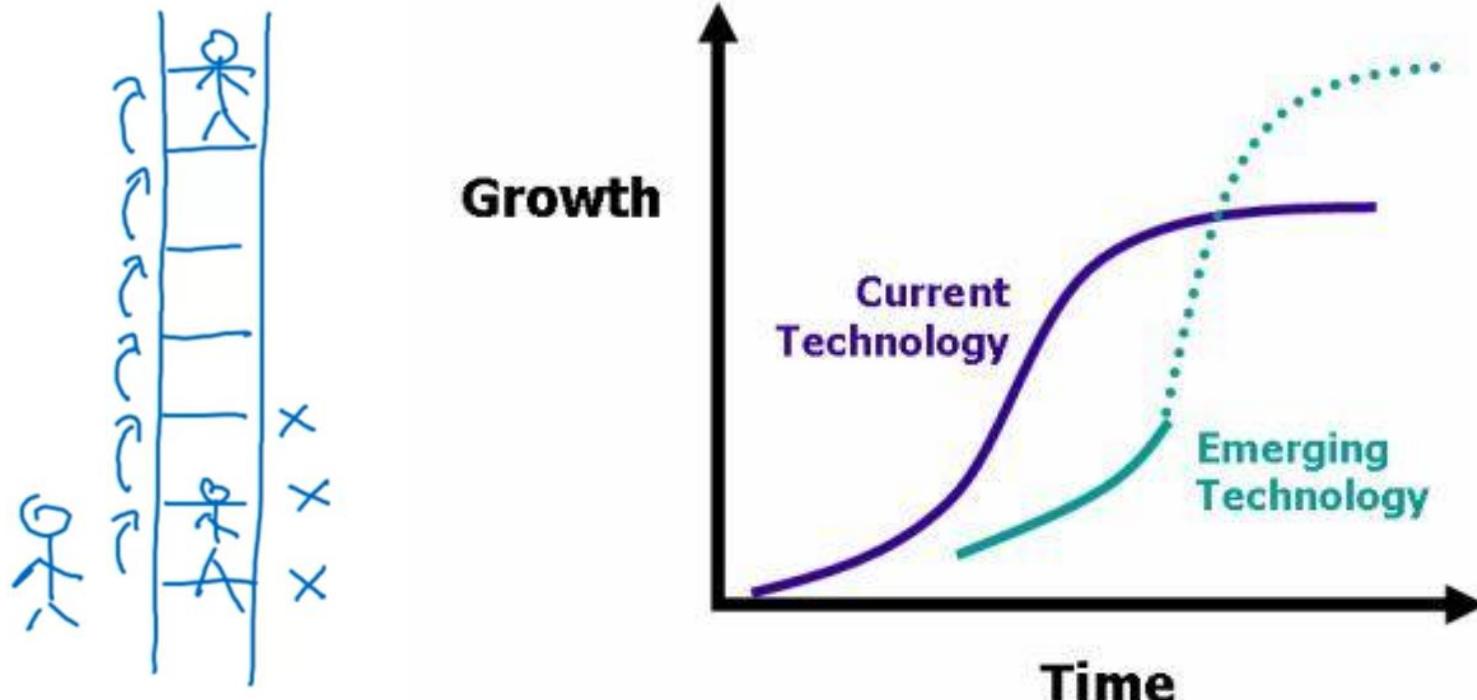
TẠO RA THÔNG TIN GIẢ TẠO

GIẢ MẠO, TẠO RÁC THÔNG TIN, ĐÁNH CẮP

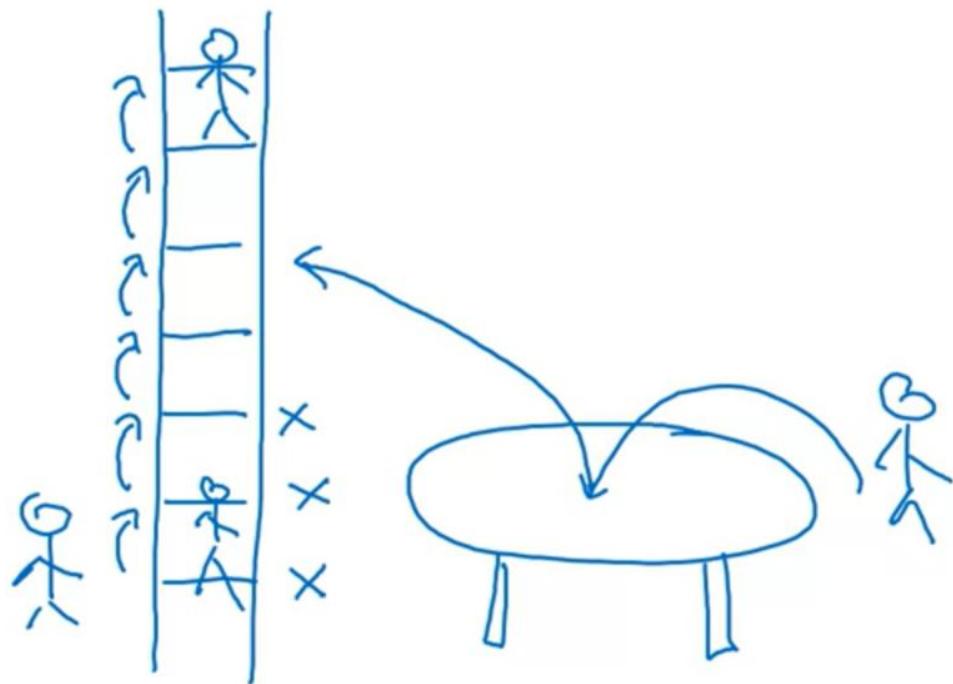
A.I VÀ CUỘC SỐNG XÃ HỘI

➤ **AI và KINH TẾ**

Developing economies

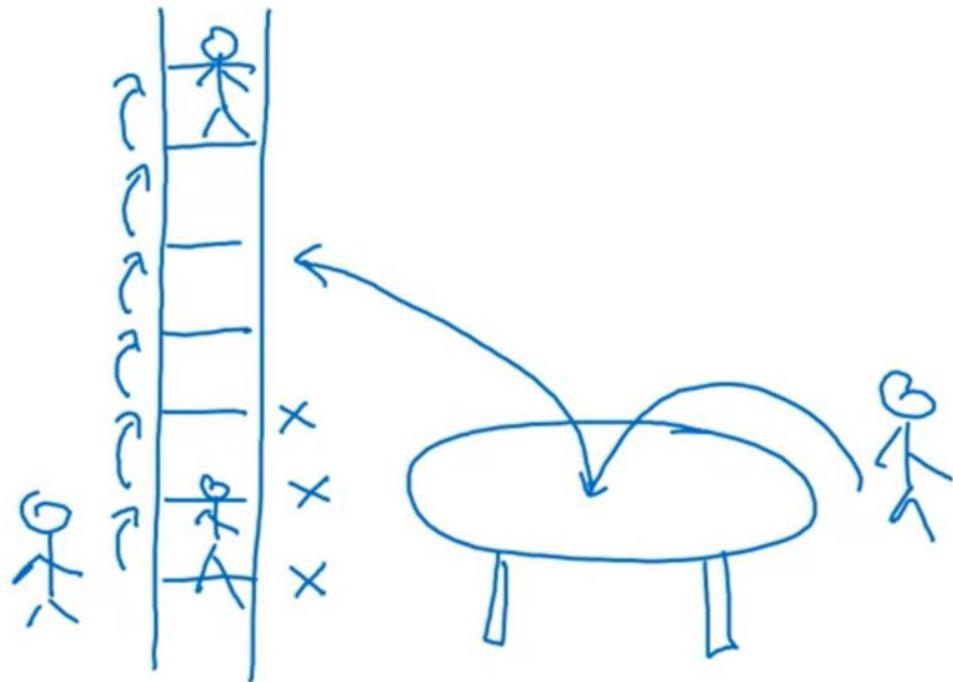


Developing economies



“Leapfrog”

Developing economies



“Leapfrog”

- Mobile phones
- Mobile payments
- Online education

CÁC QUỐC GIA ĐANG PHÁT TRIỂN ĐỀU CÓ THỂ XÂY DỰNG AI

**MẶC DÙ MỸ, TRUNG QUỐC ĐANG DẪN ĐẦU VỀ NGHIÊN CỨU VÀ
ỨNG DỤNG AI. TUY NHIÊN, MỌI THỦ VĂN CHƯA CHÍNH MUỒI.**

TẬP TRUNG PHÁT TRIỂN AI TRONG NHỮNG THỂ MẠNH QUỐC GIA

HỢP TÁC CÔNG-TƯ TRONG VIỆC TĂNG TỐC PHÁT TRIỂN AI

ĐẦU TƯ VÀO GIÁO DỤC

AI VÀ CUỘC SỐNG XÃ HỘI

➤ AI và VIỆC LÀM

TÁC ĐỘNG CỦA AI LÊN VIỆC LÀM TOÀN CẦU

Jobs displaced
by 2030

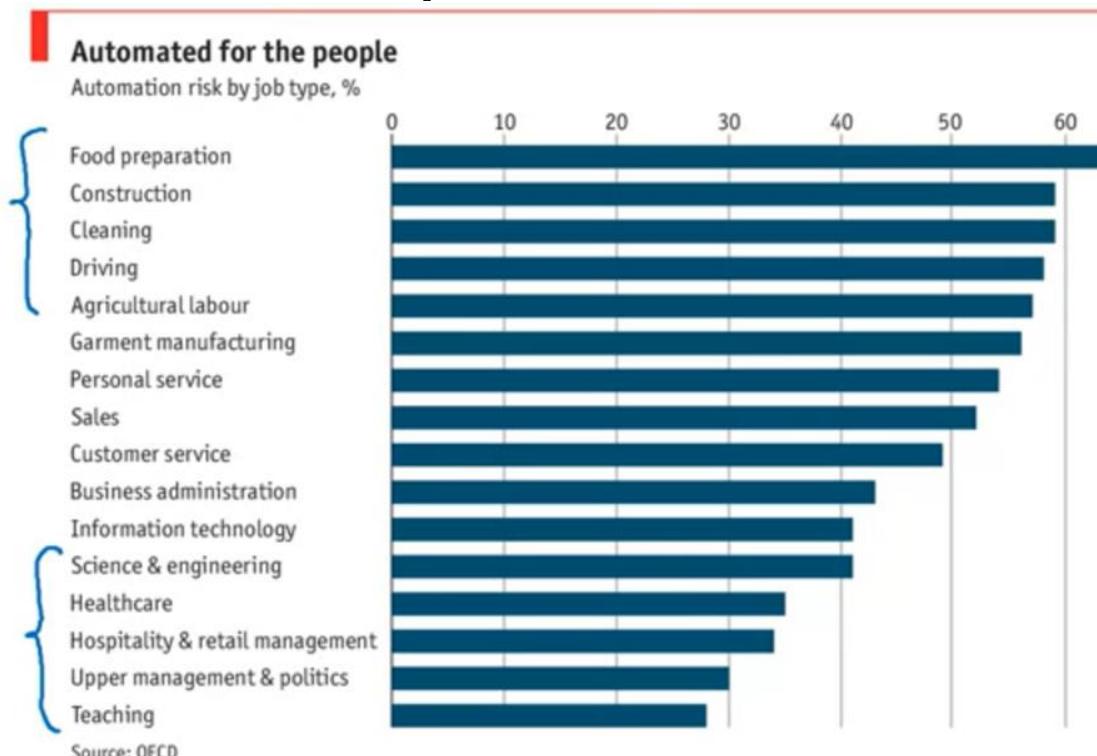
400-800 mil

Jobs created
by 2030

555-890 mil

[Source: McKinsey Global Institute.]

TÁC ĐỘNG CỦA AI LÊN VIỆC LÀM TOÀN CẦU



[Image credit: Economist.com]
[Nedelkoska, L. and G. Quintini. (2018). Automation, skills use and training. *OECD Social, Employment and Migration Working Papers*, No. 202.]

CÁM ƠN!