

XÂY DỰNG GIAO DIỆN TƯƠNG TÁC BACKEND

BÀI 4: TỔNG QUAN VỀ FRAMEWORK VUEJS

- ◎ Kết thúc bài học này bạn có khả năng
 - Hiểu VueJS, cách cài đặt môi trường
 - Viết mã VueJS đơn giản
 - Hiểu được khái niệm nội suy
 - Nắm được cú pháp Template
 - Kết hợp Bootstrap và VueJS



Phần I: Tổng quan về framework VueJS và cài đặt môi trường

- ❖ Giới thiệu VueJS
- ❖ Các thành phần của VueJS
- ❖ Ưu và nhược điểm VueJS
- ❖ Cài đặt môi trường

Phần II: Template syntax và Bootstrap

- ❖ Interpolation (Nội suy)
- ❖ Cú pháp template
- ❖ Kết hợp Bootstrap và VueJS



BÀI 4:
TỔNG QUAN VỀ FRAMEWORK
VUEJS

PHẦN I: TỔNG QUAN VỀ FRAMEWORK
VUEJS VÀ CÀI ĐẶT MÔI TRƯỜNG

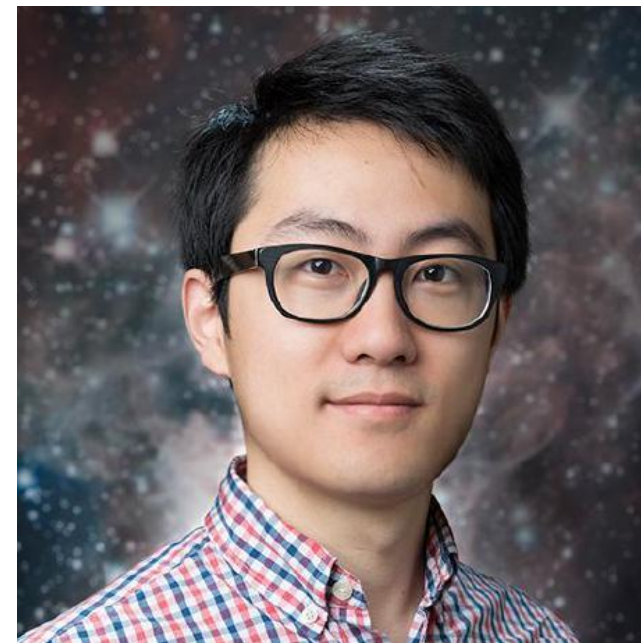
VueJS là gì?



- ☐ VueJS là một Framework JavaScript linh động được sử dụng để xây dựng giao diện người dùng (UI).
- ☐ VueJS được xây dựng dựa trên HTML, CSS và JavaScript tiêu chuẩn
- ☐ VueJS dễ dàng đáp ứng được nhu cầu tổ chức ứng dụng một trang SPA (Single-Page Applications) với độ phức tạp cao.
- ☐ VueJS hoạt động theo cơ chế tái sử dụng các component và kiến trúc linh hoạt, đơn giản hóa việc phát triển web

Lịch sử hình thành

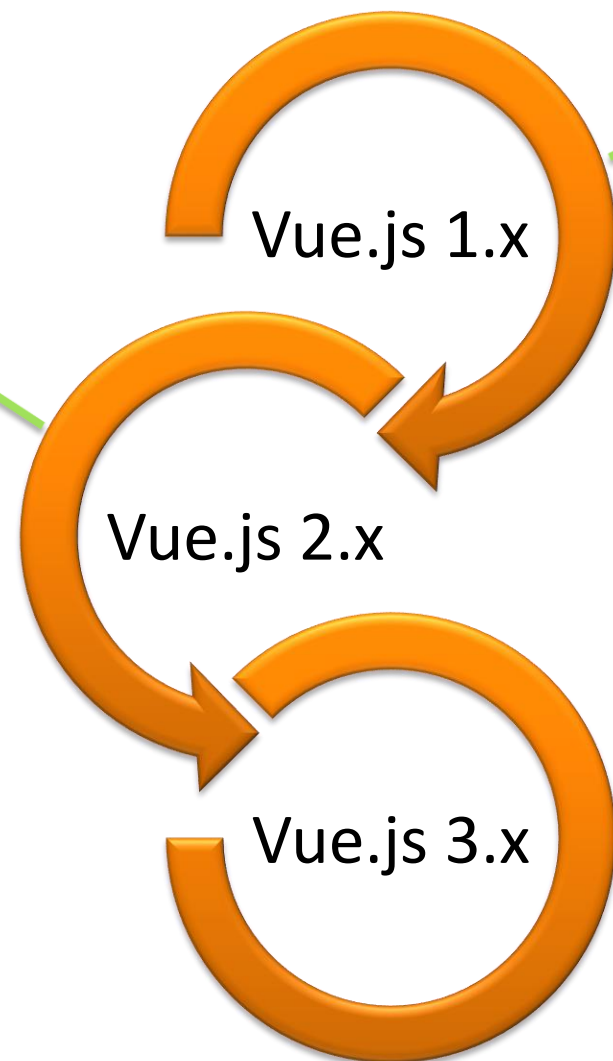
- ❑ VueJS được tạo ra bởi **Evan You**, một cựu nhân viên và lập trình viên của Google. Phiên bản đầu tiên của VueJS được phát hành vào tháng 2 năm 2014.
- ❑ Với ý tưởng tạo ra một framework nhẹ và dễ sử dụng hơn, Evan đã phát triển Vue.js bằng cách kết hợp các tính năng tốt nhất từ các framework khác.



Evan You

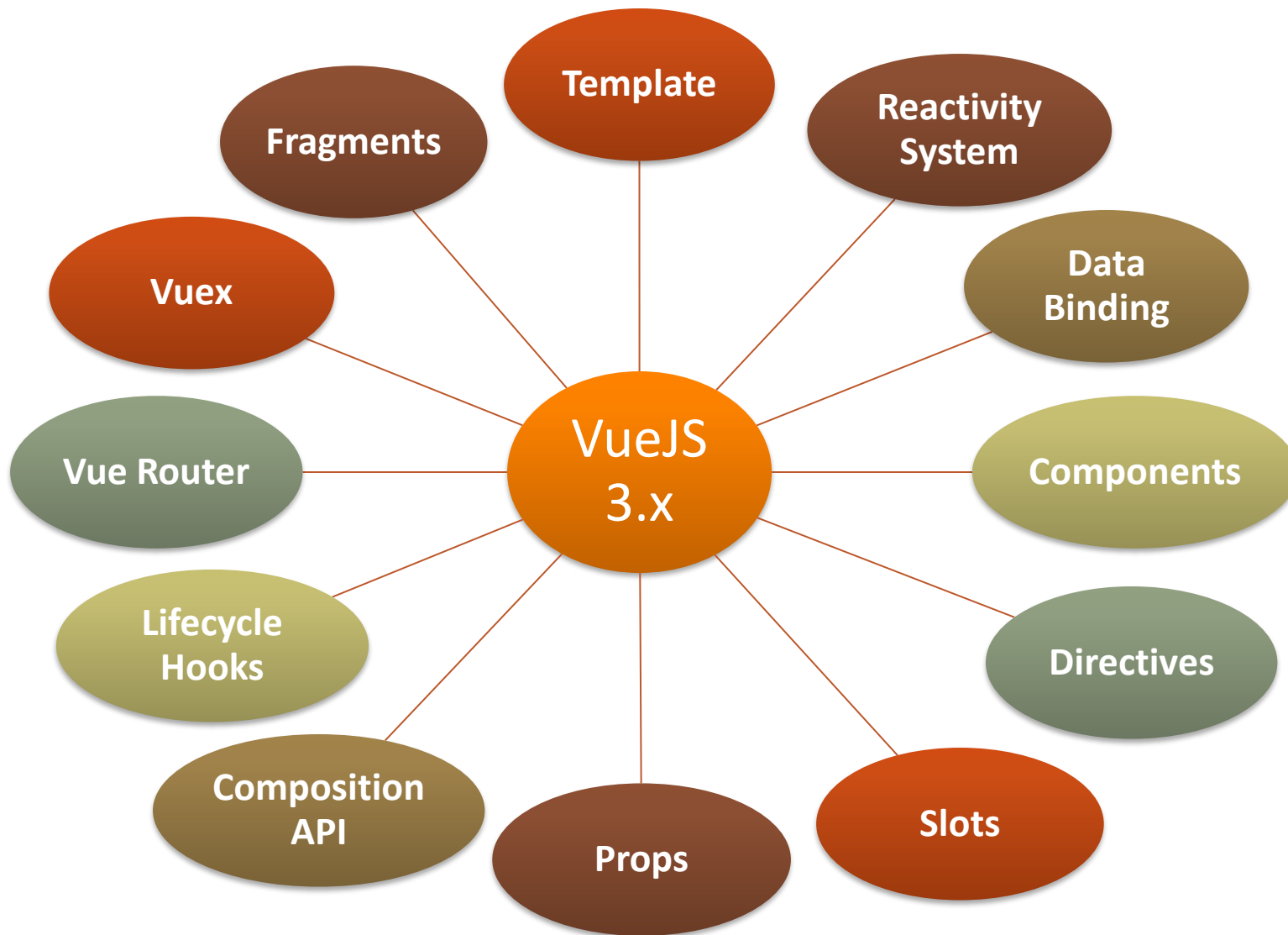
Các phiên bản VueJS

- **Ra mắt:** Tháng 10 năm 2016
- **Đặc điểm chính:**
 - Virtual DOM
 - Nâng cấp hệ thống Components
 - Transition/Animation
 - Hệ sinh thái mở rộng với Vue Router, Vuex...



- **Ra mắt:** Tháng 2 năm 2014 – **Phiên bản VueJS đầu tiên**
- **Đặc điểm chính:** Chủ yếu tập trung vào khả năng dễ học, dễ tích hợp vào các dự án hiện có và cung cấp một API đơn giản cho việc quản lý trạng thái và tạo ra các thành phần tái sử dụng.

- **Ra mắt:** Tháng 9 năm 2020
- **Đặc điểm chính:**
 - Tổ chức mã lệnh theo chức năng
 - Hiệu suất cao hơn bản 2.x
 - Hỗ trợ TypeScript
 - Kích thước gói nhỏ hơn
 - Cho phép tạo các trình kết xuất tùy chỉnh



Ưu điểm của VueJS



Nhược điểm của VueJS

Tính linh hoạt là ưu điểm lớn nhất cũng gắn liền với **nhược điểm** đáng chú ý nhất của VueJS



- Trong một dự án lớn, VueJS quá linh hoạt với nhiều cách tiếp cận khác nhau, sử dụng nhiều file component đơn lẻ, dễ dẫn đến việc mất tính thống nhất trong quy trình phát triển, đôi khi là khó kiểm soát khi phát sinh lỗi.

Chuẩn bị trước khi học

➤ Trước khi bắt đầu học VueJS, cần phải trang bị những kiến

thức cơ bản sau:

✓ HTML

✓ CSS

✓ JavaScript

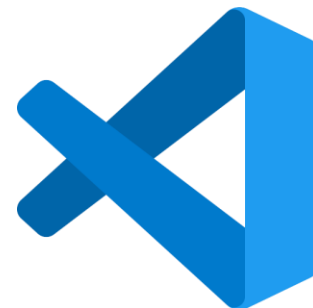
➤ Cài đặt môi trường

➤ Cài đặt công cụ lập trình



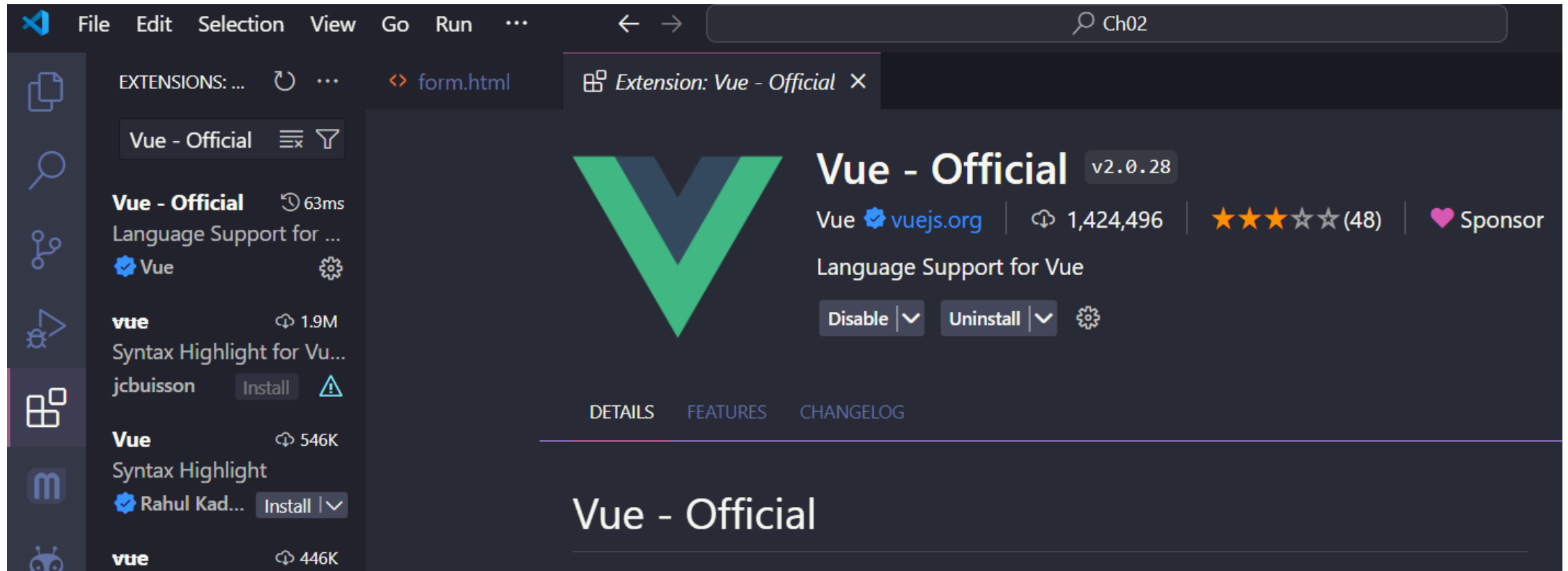
❑ Ngoài **Visual Studio Code** (VSCode), còn nhiều công cụ khác để lập trình VueJS, bao gồm:

- ❖ **WebStorm**
- ❖ **Sublime Text**
- ❖ **Atom**
- ❖ **Brackets**
- ❖ **Eclipse**
- ❖ **NetBeans**
- ❖ **IntelliJ IDEA**



Visual Studio Code

- ❑ Tìm kiếm và cài đặt Extension **Vue-Official** trên công cụ Visual Studio Code



Để cài đặt VueJS, trước tiên phải cài đặt môi trường NodeJS. Download tại trang chủ NodeJS: <https://nodejs.org/en>



Mở CMD, gõ: **node -v** và **npm -v** để kiểm tra phiên bản

The screenshot shows a Windows Command Prompt window. The title bar says "Command Prompt". The text inside the window shows the output of two commands: "node -v" returns "v20.16.0" and "npm -v" returns "10.8.1".

Có 2 cách để cài đặt VueJS

❑ Cách 1: Tiến hành cài đặt VueJS bằng NPM

- ✓ Gõ lệnh sau: **npm create vue@latest**
- ✓ Nhấn Enter để tiến hành cài đặt,
tạo tên project

Tên dự án

```
PS D:\Vuejs\Demo> npm create vue@latest

> npx
> create-vue

Vue.js - The Progressive JavaScript Framework

✓ Project name: my-project
✓ 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
✓ Add ESLint for code quality? ... No / Yes
✓ Add Vue DevTools 7 extension for debugging? (experimental) ... No / Yes

Scaffolding project in D:\Vuejs\Demo\my-project...

Done. Now run:

  cd my-project
  npm install
  npm run dev
```

❑ Cách 2: Tạo dự án Vue với Vite

✓ Gõ: **npm create vite@latest**

```
PS D:\Vuejs\Demo> npm create vite@latest  
  
> npx  
> create-vite  
  
✓ Project name: ... vue  
✓ Select a framework: » Vue  
✓ Select a variant: » JavaScript  
  
Scaffolding project in D:\Vuejs\Demo\vue...  
  
Done. Now run:  
  
  cd vue  
  npm install  
  npm run dev
```

Tên dự án



Vite

Giới thiệu **Vite** – Người chơi hệ tốc độ:



Vite là một công cụ xây dựng front-end thế hệ tiếp theo, được tạo ra bởi **Evan You**, người đã tạo ra Vue. Ban đầu Vite dành riêng cho VueJS, về sau mở rộng hỗ trợ cả React và một số thư viện front-end khác. Nó cung cấp trải nghiệm phát triển nhanh hơn và gọn gàng hơn cho các dự án web hiện tại. Đặc biệt Vite có hỗ trợ các biến thể của JS và CSS như:

- Typescript, JSX (JavaScript XML)
- SCSS, LESS,...

Khởi chạy dự án Vue + Vite

- ✓ Di chuyển vào thư mục vừa tạo bằng lệnh: **cd project-name**

```
PS D:\Vuejs\Demo> cd vue
PS D:\Vuejs\Demo\vue> |
```

- ✓ Tiếp theo, gõ lệnh sau: **npm install**
- ✓ Nhấn Enter để tiến hành cài đặt npm

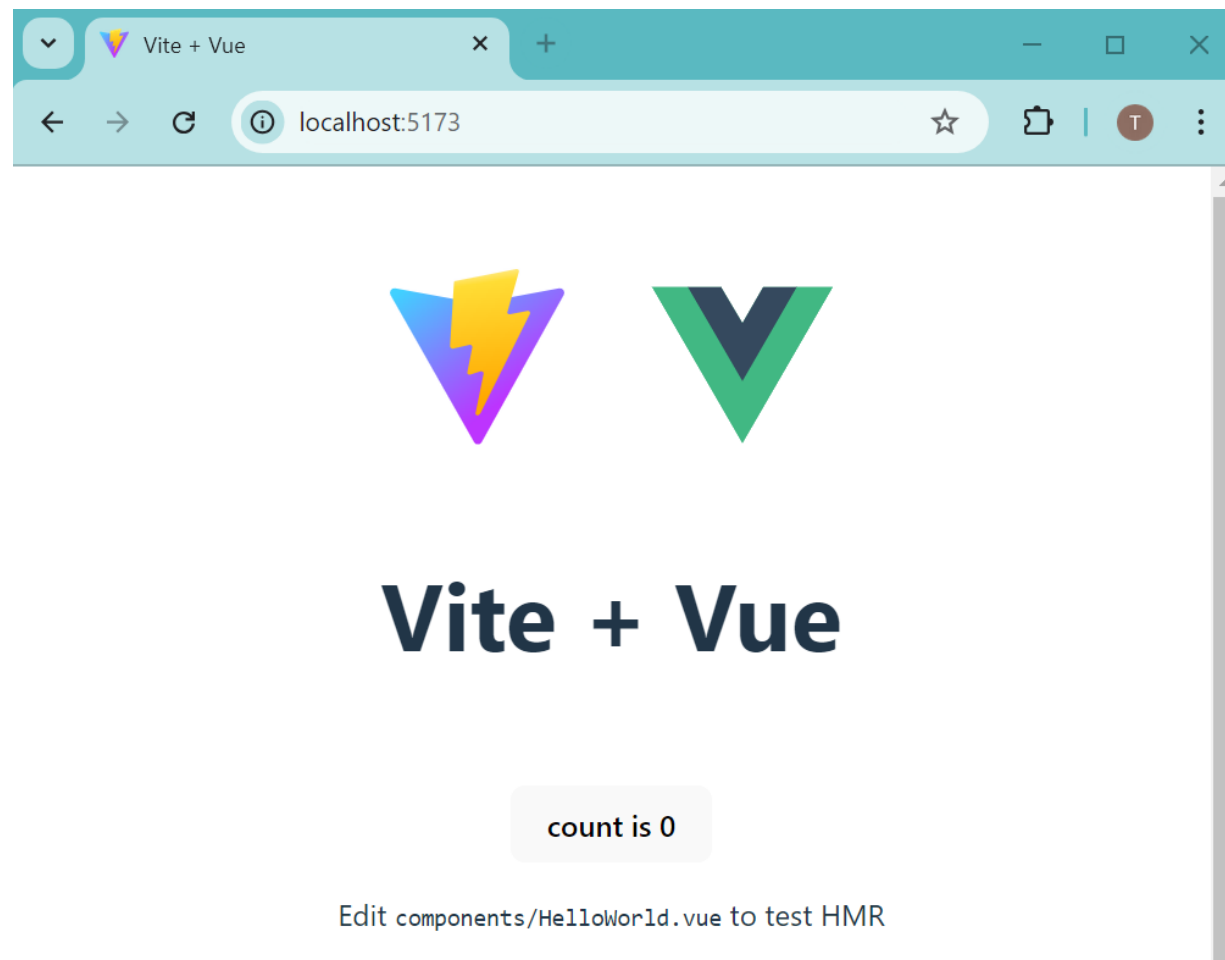
```
PS D:\Vuejs\Demo\vue> npm install

added 31 packages, and audited 32 packages in 11s

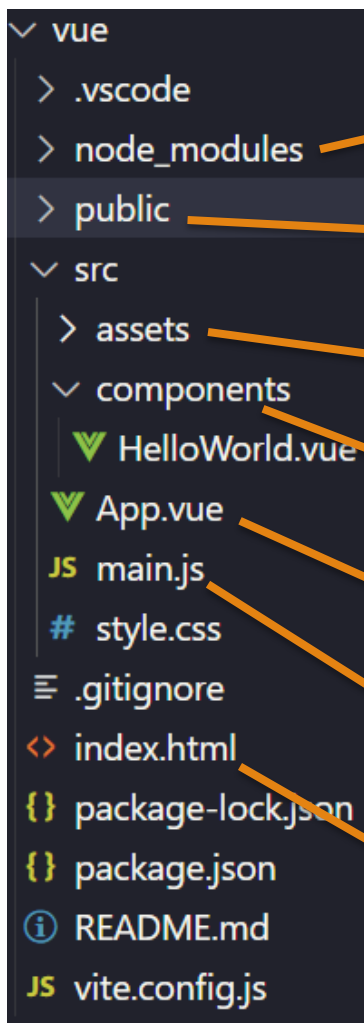
4 packages are looking for funding
  run `npm fund` for details
```

Khởi chạy dự án Vue + Vite

- ✓ Gõ lệnh: **npm run dev**
- ✓ Local: <http://localhost:5173/>



Project mặc định chạy trên cổng **5173** và cấu trúc thư mục sẽ như sau:



Chứa tất cả các thư viện cần để xây dựng Vue.

Chứa các static assets nếu không muốn chạy thông qua webpack

Chứa tài nguyên được nhập vào trong các components

Tất cả UI components của project.

Tập chính của dự án, nơi khởi tạo các component khác

Đây chính là file render ra App.vue component

Dùng cho ứng dụng SPA với một trang index duy nhất

Chương trình VueJS đầu tiên

B1: Tạo mới một component với file **.vue** có cấu trúc như sau

HelloWorld.vue

```
<template>
  <div>
    <h1>Welcome to VueJS</h1>
  </div>
</template>

<script setup>
</script>

<style scoped>
</style>
```

HTML

JS

CSS

B2: Khai báo component vừa tạo vào file **App.vue**

HelloWorld.vue

```
<template>
  <div>
    <h1>Welcome to VueJS</h1>
  </div>
</template>
<script setup></script>
<style scoped></style>
```

App.vue

```
<script setup>
import HelloWorld from
'./components/HelloWorld.vue'
</script>

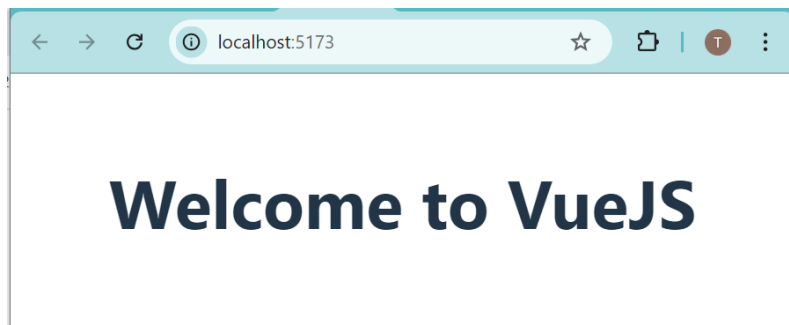
<template>
  <HelloWorld></HelloWorld>
</template>

<style scoped>
</style>
```

JS

HTML

CSS





Tái hiện demo cài đặt môi trường và tạo dự án Vue đầu tiên

BÀI 4:
TỔNG QUAN VỀ FRAMEWORK
VUEJS

PHẦN II: TEMPLATE SYNTAX VÀ
BOOTSTRAP

- ❑ Trong Vue.js 3, có thể khai báo biến và phương thức trong component chủ yếu thông qua **Composition API** hoặc **Option API**.
 - ❖ **Option API** là cách truyền thống để khai báo biến và phương thức trong Vue.js.
 - ❖ **Composition API** cho phép tổ chức logic theo các chức năng, tạo nên code dễ quản lý hơn, đặc biệt là trong các component lớn. (*Phương pháp Composition API sẽ được giới thiệu kỹ ở phần học **Reactivity** sau*)

Khai báo biến và phương thức trong component thông qua Option API.

1. Khai báo biến (data)

Khai báo các biến trong một phương thức data và trả về một object chứa các biến này.

```
export default {  
  data() {  
    return {  
      count: 0,    // Khai báo biến trong data()  
      user: {  
        name: 'Nam',  
        age: 25  
      }  
    };  
  }  
};
```

Khai báo biến và phương thức trong component thông qua Option API.

- 2. Khai báo phương thức (methods):** Các phương thức được khai báo trong object methods.

```
export default {  
  data() {  
    return {  
      count: 0  
    };  
  },  
  methods: {  
    increment() {  
      this.count++; // Dùng this để truy cập biến trong data  
    },  
    reset() {  
      this.count = 0; //đặt lại count về 0  
    }  
  }  
};
```

- ❑ Vue.js sử dụng các cú pháp mẫu trong HTML để bind và render DOM thành các tag HTML và tất cả các vue.js template này đều phải tuân thủ theo các tag HTML
- ❑ Bên dưới, Vue.js biên dịch template thành các hàm render Virtual DOM (DOM ảo). Kết hợp với hệ thống reactivity (phản ứng), Vue.js có thể xác định một cách thông minh số lượng tối thiểu các component cần phải render lại, và áp dụng số lượng tối thiểu các hiệu chỉnh về DOM khi trạng thái của ứng dụng thay đổi.

Interpolation (Phép nội suy):

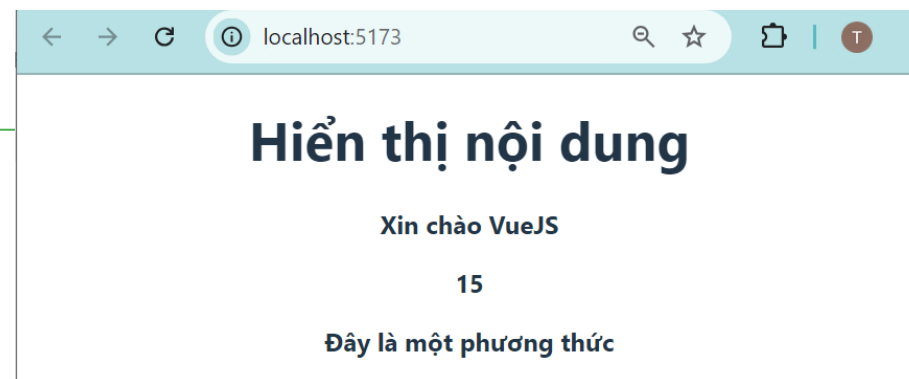
- ❖ Là quá trình thêm một văn bản, nội dung, attribute ,... vào các thẻ HTML bằng Vue.js.
- ❖ Cú pháp: '{{ }}

Ví dụ: `Thông điệp: {{ message }}`

Interpolation.vue

```
<template>
  <div>
    <h1>Hiển thị nội dung</h1>
    <h2>{{ message }}</h2>
    <h2>{{ number + 5 }}</h2>
    <h2>{{ sayHi() }}</h2>
  </div>
</template>
```

```
<script>
export default {
  data() {
    return {
      message: 'Xin chào VueJS',
      number: 10
    }
  },
  methods: {
    sayHi() {
      return 'Đây là một phương thức'
    }
  }
}
</script>
```



Raw HTML:

- ❖ Dùng để hiển thị dữ liệu ra dưới dạng HTML code (tương tự như innerHTML trong Javascript)
- ❖ Cú pháp: `<tag v-html="data"></tag>`

Trong đó:

- ❖ **tag** là các tag trong HTML.
- ❖ **data** là dữ liệu muốn gắn vào tag đó (dữ liệu này thường được khai báo trong data scope của vue.js).

```
<template>
  <div>
    <h2 v-html="content"></h2>
  </div>
</template>

<script>
  export default {
    data() {
      return {
        content: 'xin chào các bạn'
      }
    }
  }
</script>
```



Attributes: Để thêm các attribute vào tag HTML sử dụng cú pháp sau:

```
<tag v-bind:attributeName="data"></tag>
```

❑ Trong đó:

- ❖ **attributeName**: là tên của attribute khi muốn thực hiện binding.
- ❖ **data**: là data đã thiết lập trong vue.js.

```
<template>
  <div>
    <h2 v-bind:class="className" v-bind:style="styleData" v-html="content"></h2>
  </div>
</template>
<script>
  export default {
    data() {
      return {
        content: 'xin chào các bạn',
        className: 'text-red',
        styleData: 'font-size: 5rem'
      }
    }
  }
</script>
<style scoped>
  .text-red{color: red;}
</style>
```



JavaScript Expressions: Vue hỗ trợ các biểu thức JavaScript bên trong tất cả các liên kết dữ liệu. Ví dụ:

```
{{ number + 1 }}
```

```
{{ ok ? 'YES' : 'NO' }}
```

```
{{ message.split('').reverse().join('') }}
```

```
<div :id="`list-${id}`"></div>
```

SỬ DỤNG JAVASCRIPT EXPRESSIONS

```
<template>
  <p>Phương thức sayHi trả về: {{ sayHi() }}</p>
  <p>Số này là số {{ number%2==0?'Chẵn':'Lẻ' }}</p>
</template>
<script>
  export default {
    data() {
      return {
        number : 22
      }
    },
    methods: {
      sayHi : function () {
        return 'vuejs.org'
      }
    }
  }
</script>
```

Phương thức sayHi trả về: vuejs.org

Số này là số Chẵn

Khái niệm **Directives** (Chỉ thị)

- ❑ **Directives** là các thuộc tính đặc biệt bắt đầu bằng tiền tố **v-**. Nhiệm vụ của directive là áp dụng các cập nhật vào DOM khi giá trị biểu thức thay đổi.
- ❑ **Directives** làm việc với view và thực hiện các nhiệm vụ lặp đi lặp lại một cách dễ dàng.

Một số directive thông dụng:

Thuộc tính	Chức năng
v-bind	Dùng để lấy dữ liệu và hiển thị ra template
v-on	Lắng nghe các sự kiện DOM và thực thi JavaScript khi những sự kiện này được kích hoạt
v-if	Kiểm tra điều kiện trước khi hiển thị ra
v-show	Tương tự như v-if , nhưng thay vì thỏa mãn điều kiện mới render ra thì v-show sẽ render ra hết, nhưng chỉ hiển thị phần thỏa mãn điều kiện, những phần còn lại sẽ được đặt thuộc tính display: none.
v-model	Tạo ra ràng buộc hai chiều (two-way binding) giữa form input và trạng thái sử dụng
v-for	Thường dùng dùng để render một danh sách các item dựa trên một mảng

Phần này chỉ giới thiệu **v-bind** và **v-on**, các thuộc tính khác sẽ giới thiệu chi tiết hơn ở các bài sau

1. v-bind: Lấy dữ liệu và hiển thị ra template

❑ Cú pháp: **v-bind:<tên thuộc tính>="giá trị"**

Hoặc ngắn gọn hơn là **:<tên thuộc tính>="giá trị"**

❑ Ví dụ: `<input type="text" v-bind:value="name">`

Hoặc: `<input type="text" :value="name">`

2. v-on: Lắng nghe các sự kiện DOM và thực thi JavaScript khi những sự kiện này được kích hoạt

❑ Cú pháp: ***v-on:<tên sự kiện>="doSomething"***

Hoặc ngắn gọn hơn là ***@<tên sự kiện>=" doSomething"***

❑ Ví dụ: `<a v-on:click="showInfo">...`

Hoặc: `<a @click="showInfo">...`

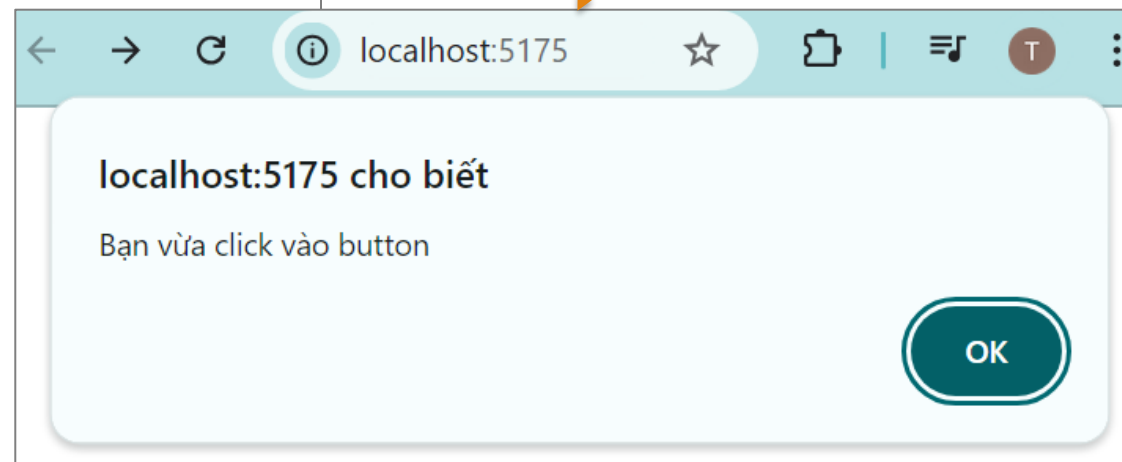
Demo v-on:

```
<template>
  <p>Click vào button để xem kết quả</p>
  <button class="btn btn-info" v-on:click="showInfo">
    Click
  </button>
</template>
<script>

export default {
  methods: {
    showInfo: function(){
      alert('Bạn vừa click vào button')
    }
  }
}
</script>
```

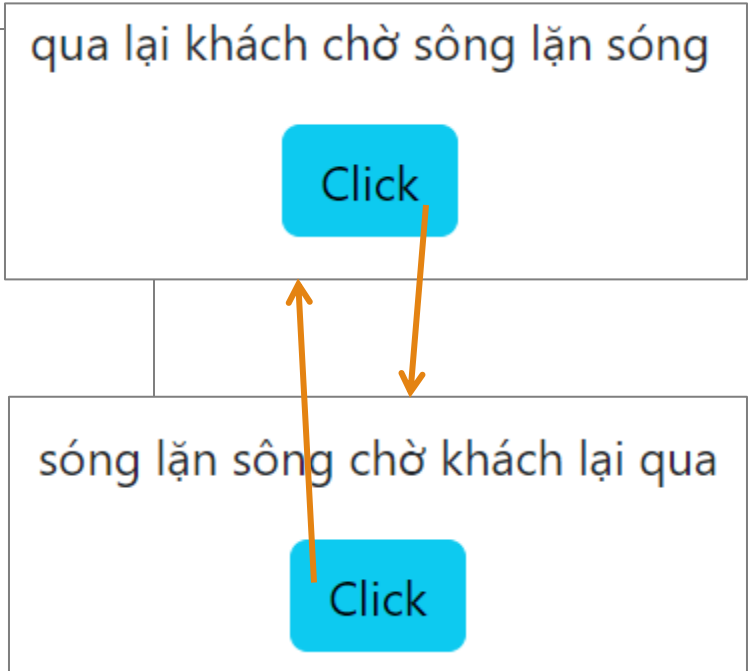
Click vào button để xem kết quả

Click



Demo v-on kết hợp một số phương thức:

```
<template>
  <p>{{ message }}</p>
  <button class="btn btn-info" @click="showInfo">
    Click</button>
</template>
<script>
export default {
  data() {
    return {
      message: 'qua lại khách chờ sông lặn sóng',
    }
  },
  methods: {
    showInfo: function(){
      this.message = this.message.split(' ').reverse().join(' ');
    }
  }
}
</script>
```

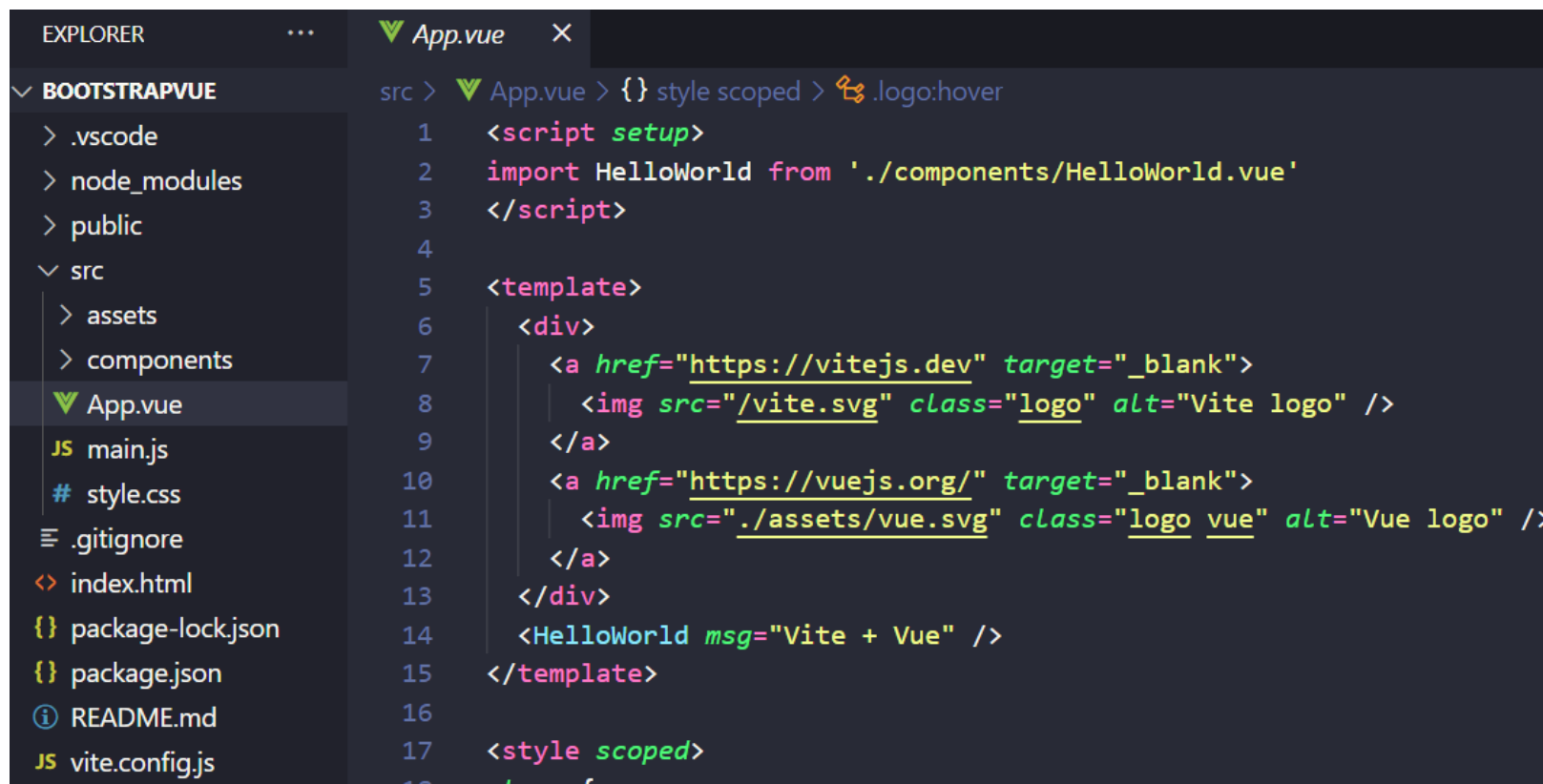


Kết hợp Bootstrap và Vue.js: hai framework nổi tiếng trong giới lập trình.



B1: Tạo dự án Vue.js

- ✓ Gõ: **npm create vite@latest**, đặt tên cho dự án
- ✓ Gõ: **npm install**



The screenshot shows a VS Code editor with a project named 'BOOTSTRAPVUE'. The Explorer sidebar on the left shows the file structure: .vscode, node_modules, public, src (containing assets, components, App.vue, main.js, style.css), .gitignore, index.html, package-lock.json, package.json, README.md, and vite.config.js. The App.vue file is open in the editor, showing the following code:

```
src > App.vue > {} style scoped > .logo:hover
1  <script setup>
2  import HelloWorld from './components/HelloWorld.vue'
3  </script>
4
5  <template>
6    <div>
7      <a href="https://vitejs.dev" target="_blank">
8        
9      </a>
10     <a href="https://vuejs.org/" target="_blank">
11       
12     </a>
13   </div>
14   <HelloWorld msg="Vite + Vue" />
15 </template>
16
17 <style scoped>
18   .logo {
```

B2: Cài đặt Bootstrap trong dự án VueJS vừa tạo

- ✓ Gõ: **npm install bootstrap**
- ✓ Gõ: **npm install bootstrap-vue**

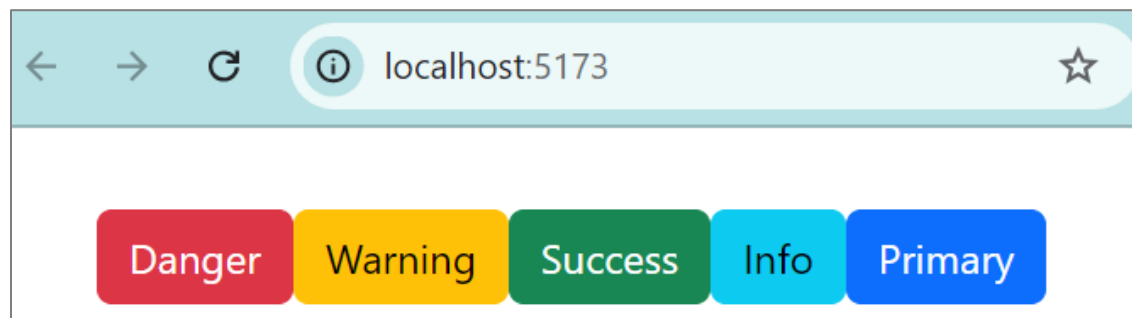
B3: Mở file **main.js** và thực hiện import Bootstrap bằng lệnh như sau:

```
import 'bootstrap/dist/css/bootstrap.css'  
import 'bootstrap-vue/dist/bootstrap-vue.css'
```

```
JS main.js  
import { createApp } from 'vue'  
import './style.css'  
import App from './App.vue'  
import 'bootstrap/dist/css/bootstrap.css'  
import 'bootstrap-vue/dist/bootstrap-vue.css'  
  
createApp(App).mount('#app')
```

B4: Mở thành phần HelloWorld.vue và thêm code sau:

```
<template>
  <div class="container">
    <button class="btn btn-danger">Danger</button>
    <button class="btn btn-warning">Warning</button>
    <button class="btn btn-success">Success</button>
    <button class="btn btn-info">Info</button>
    <button class="btn btn-primary">Primary</button>
  </div>
</template>
```





Tái hiện demo kết hợp Bootstrap + Vue

- ☑ Hiểu về VueJS, ưu nhược điểm
- ☑ Cài đặt môi trường
 - ☑ Công cụ cài đặt
 - ☑ Môi trường
 - ☑ Khởi tạo dự án
- ☑ Hiểu cơ bản về nội suy, cú pháp template
- ☑ Kết hợp được Bootstrap và VueJS



Thank
You

