**ỦY BAN NHÂN DÂN TP.HCM**

**TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC**



**BÁO CÁO ĐỒ ÁN CHUYÊN ĐỀ WEB 2**

**NĂM HỌC: 2020**

**GV: Phan Thanh Nhuần**

**Thành viên nhóm: C**

* **Nguyễn Thu Tuyết**
* **Ngô Thị Ngọc Diễm**
* **Nguyễn Thị Tuyết Nhung**
* **Lê Thanh Thiện**
* **Trần Văn Thành**

BẢNG PHÂN CÔNG VIỆC NHÓM C

Báo cáo 1: Tìm Hiểu Về Vuejs

|  |  |  |
| --- | --- | --- |
| STT | Tên | Vuejs |
| 1 | Nguyễn Thu Tuyết (nhóm trưởng)  0332058811  [Nthutuyet98@gmail.com](mailto:Nthutuyet98@gmail.com) | -Tìm hiểu vê Vue Instance  -Tìm hiểu về Template systax  -Cho ví dụ demo cho mỗi phần  -Người review - Thành |
| 2 | Ngô Thị Ngọc Diễm  0987919502  Ngocdiem98.tdc@gmail.com | - Tìm hiểu về [Computed Properties and Watchers](https://vuejs.org/v2/guide/computed.html)  -Tìm hiểu về [Class and Style Bindings](https://vuejs.org/v2/guide/class-and-style.html)  -Cho ví dụ demo  -Người review – Thiện |
| 3 | Nguyễn Thị Tuyết Nhung  0904263815  Ntnhung238@gmail.com | -Tìm hiểu về [Conditional Rendering](https://vuejs.org/v2/guide/conditional.html)  -Tìm hiểu về [List Rendering](https://vuejs.org/v2/guide/list.html)  -Cho ví dụ demo  -Người review – Diễm |
| 4 | Lê Thanh Thiện  0349477734  Lethanhthien414@gmail.com | -Tìm hiểu về [Event Handling](https://vuejs.org/v2/guide/events.html)  -Tìm hiểu về [Form Input Bindings](https://vuejs.org/v2/guide/forms.html)  -Cho ví dụ demo  -Người review - Nhung |
| 5 | Trần Văn Thành  0339778775  [Thanhssntc2@gmail.com](mailto:Thanhssntc2@gmail.com) | -Tìm hiểu về [Components Registration](https://vuejs.org/v2/guide/components.html)  - Tìm hiểu về [Component Props](https://vuejs.org/v2/guide/components-registration.html)  -Cho ví dụ demo  người review – Tuyết |

Chủ Đề

VUEJS



Hình 1: Vuejs

**MỤC LỤC**

[**1.** **Giới thiệu VueJS, NodeJS** 9](#_Toc45852097)

[**1.1.** **Sự ra đời của NodeJS** 9](#_Toc45852098)

[**1.2.** **Cài đặt NodeJs** 10](#_Toc45852099)

[**1.3.** **VueJS** 15](#_Toc45852100)

[**1.4.** **Cách cài đặt Vuejs** 15](#_Toc45852101)

[**2.** **Vue Instance** 18](#_Toc45852102)

[**2.1.** **Creating a Vue Instance** 18](#_Toc45852103)

[**2.2.** **Data and Method** 19](#_Toc45852104)

[**2.3.** **Lifecyle Diagram, Intance Lifecyle Hook** 20](#_Toc45852105)

[**3.** **Template Syntax** 23](#_Toc45852106)

[**3.1.** **Interpolation** 24](#_Toc45852107)

[**3.1.1.** **Text** 24](#_Toc45852108)

[**3.1.2.** **Raw HTML** 25](#_Toc45852109)

[**3.1.3.** **Attributes** 26](#_Toc45852110)

[**3.1.4.** **Using JavaScript Expressions** 27](#_Toc45852111)

[**3.2.** **Directives** 27](#_Toc45852112)

[**3.2.1.** **Arguments** 27](#_Toc45852113)

[**3.2.2.** **Dynamic Arguments** 28](#_Toc45852114)

[**3.2.3.** **Modifiers** 30](#_Toc45852115)

[**3.3.** **Shorthands** 32](#_Toc45852116)

[**3.3.1.** **V-bind Shorthands** 32](#_Toc45852117)

[**3.3.2.** **V-on Shorthands** 32](#_Toc45852118)

[**4.** **Tìm hiểu về Computed Properties and Watchers** 32](#_Toc45852119)

[**4.1.** **Computed Properties (thuộc tính):** 32](#_Toc45852120)

[**4.1.1.** **So sánh Computed Caching và Methods:** 33](#_Toc45852121)

[**4.1.2.** **So sánh Computed với Watched Property** 38](#_Toc45852122)

[**4.1.3.** **Computed Setter** 40](#_Toc45852123)

[**4.2.** **Watchers:** 42](#_Toc45852124)

[**5.** **Tìm hiểu Class and Style Bindings** 44](#_Toc45852125)

[**5.1.** **Binding HTML Classes** 44](#_Toc45852126)

[**5.1.1.** **Object Syntax** 44](#_Toc45852127)

[**5.1.2.** **Array Syntax** 46](#_Toc45852128)

[**5.1.3.** **With Components:** 46](#_Toc45852129)

[**5.2.** **Binding Inline Styles** 47](#_Toc45852130)

[**5.2.1.** **Object Syntax** 47](#_Toc45852131)

[**5.2.2.** **Array Syntax** 48](#_Toc45852132)

[**6.** **Conditional Rendering: Điều kiện** 48](#_Toc45852133)

[**7.** **List Rendering: Vòng lặp** 50](#_Toc45852134)

[**8.** **Event Handling** 53](#_Toc45852135)

[**8.1.** **Listening to Event** 53](#_Toc45852136)

[**8.2.** **Method Event Handlers** 53](#_Toc45852137)

[**8.3.** **Methods in Inline Handlers** 54](#_Toc45852138)

[**8.4.** **Event Modifiers** 55](#_Toc45852139)

[**8.5.** **Key Modifiers** 55](#_Toc45852140)

[**8.6.** **System Modifier Keys** 56](#_Toc45852141)

[**8.7.** **Why Listeners in HTML?** 56](#_Toc45852142)

[**9.** **Form Input Bindings** 56](#_Toc45852143)

[**9.1.** **Basic Usage** 56](#_Toc45852144)

[**9.2.** **Value Bindings** 62](#_Toc45852145)

[**9.3.** **Modifiers** 63](#_Toc45852146)

[**9.4.** **V-model with Components** 65](#_Toc45852147)

[**10.** **Component** 65](#_Toc45852148)

[**10.1.** **Cách tạo một component trong vue-cli** 65](#_Toc45852149)

[**10.2.** **Đăng ký component** 66](#_Toc45852150)

[**10.3.** **Props trong component** 67](#_Toc45852151)

**MỤC LỤC HÌNH ẢNH**

[Hình 1: Vuejs 3](#_Toc45852152)

[Hình 2: JavaScript kết hợp với PHPxây dựng các ứng dụng Web có thể truy cập qua trình duyệt mà không cần lãng phí một phần mềm nào. 9](#_Toc45852153)

[Hình 3: Logo ngôn ngữ "V8". 9](#_Toc45852154)

[Hình 4: Logo NodeJS. 10](#_Toc45852155)

[Hình 5: Sau khi nhấn vào link sẽ hiện ra bảng như trên. 10](#_Toc45852156)

[Hình 6: Thư mục chứa NodeJs vừa download. 11](#_Toc45852157)

[Hình 7: Nhấn Next bắt đầu quá trinh cài đặt. 11](#_Toc45852158)

[Hình 8: Tick chọn I Accept… rồi nhấn Next. 12](#_Toc45852159)

[Hình 9: Tiếp tục nhấn Next. 12](#_Toc45852160)

[Hình 10: Tiếp tục Next. 13](#_Toc45852161)

[Hình 11: Nhấn Install để cài đặt. 13](#_Toc45852162)

[Hình 12: Bấm Finish để kết thúc cài đặt. 14](#_Toc45852163)

[Hình 13: Click vào icon Node ở màn hình Desktop để khởi động Node. 14](#_Toc45852164)

[Hình 14: Sau khi click chuột giao diện khởi động Node hiện ra và ta chỉ cần tắt nó là đã thành công bước cài đặt Nodejs. 14](#_Toc45852165)

[Hình 15: Ảnh minh họa. 15](#_Toc45852166)

[Hình 16: Version hiện lên sau khi kiểm tra. 15](#_Toc45852167)

[Hình 17: Màn hình sau khi chạy dòng lệnh npm i -g vue-cli. 16](#_Toc45852168)

[Hình 18: Màn hình sau khi chạy dòng lệnh vue init webpack-simple sẽ xuất hiện các câu hỏi. 16](#_Toc45852169)

[Hình 19: Ảnh minh họa. 16](#_Toc45852170)

[Hình 20: Hai câu lệnh xuất hiện khi nhập các câu lệnh trên. 16](#_Toc45852171)

[Hình 21: Màn hình sau khi chạy npm install. 17](#_Toc45852172)

[Hình 22: Màn hình sau khi chạy npm run dev. 17](#_Toc45852173)

[Hình 23: Cấu trúc VueJs sau khi cài đặt. 18](#_Toc45852174)

[Hình 24 Hình ảnh tạo Vue instance. 18](#_Toc45852175)

[Hình 25: VD demo tạo đoạn text script cơ bản của Vue. 19](#_Toc45852176)

[Hình 26 Code html tính tổng. 19](#_Toc45852177)

[Hình 27 Xử lý phép cộng. 20](#_Toc45852178)

[Hình 28 Kết quả xử lý phép cộng. 20](#_Toc45852179)

[Hình 29 Vòng đời Vue. 21](#_Toc45852180)

[Hình 30: Màn hình Consolog: Nơi chứa code ở trình duyệt. 22](#_Toc45852181)

[Hình 31: Code demo ở màn hình Script. 22](#_Toc45852182)

[Hình 32: Màn hình Console.log - nơi chứa code tại trình duyệt. 22](#_Toc45852183)

[Hình 33: Code demo đơn giản tại file Script. 23](#_Toc45852184)

[Hình 34: Sửa color của text thành “yellow”. 23](#_Toc45852185)

[Hình 35: Kết quả sau khi sửa màu sắc ở trình duyệt. 23](#_Toc45852186)

[Hình 36: Ví dụ text của phần Interpolation. 24](#_Toc45852187)

[Hình 37: Đoạn code thêm v-once vào <span>. 25](#_Toc45852188)

[Hình 38: Bắt sự kiện click nhưng không thay đổi. 25](#_Toc45852189)

[Hình 39: Ví dụ raw html của phần Interpolation. 26](#_Toc45852190)

[Hình 40: Ví dụ Attributes của phần Interpolation. 26](#_Toc45852191)

[Hình 41: Ví dụ về Using JavaScript Expressions của phần Interpolation. 27](#_Toc45852192)

[Hình 42: Demo đơn giản về v-bind:href. 27](#_Toc45852193)

[Hình 43: Lỗi thường gặp khi không dùng v-bind. 28](#_Toc45852194)

[Hình 44: Sau khi dùng v-bind và kết quà thành công. 28](#_Toc45852195)

[Hình 45: Demo đơn giản ở màn hình Script. 29](#_Toc45852196)

[Hình 46: Click nào nút “Thay đổi hình ảnh” để thay đổi ảnh. 29](#_Toc45852197)

[Hình 47: Kết quả sau khi đã click vào nút "Thay đổi hình ảnh". 30](#_Toc45852198)

[Hình 48: Demo tạo 1 nút input để load trang. 30](#_Toc45852199)

[Hình 49: Demo trường hợp không dùng .prevent. 31](#_Toc45852200)

[Hình 50: Demo tạo nút load trang có dùng .prevent. 31](#_Toc45852201)

[Hình 51: Kết quả hiển thị trên trình duyệt khi dùng .prevent. 32](#_Toc45852202)

[Hình 52: Demo computed-properties ở màn hình index. 32](#_Toc45852203)

[Hình 53: Demo computed-properties ở màn hình script.js. 33](#_Toc45852204)

[Hình 54: Kết quả khi chạy trên màn hình trình duyệt. 33](#_Toc45852205)

[Hình 55: Demo Method ở màn hình index. 34](#_Toc45852206)

[Hình 56: Demo method ở script. 35](#_Toc45852207)

[Hình 57: Kết quả sau khi demo method trên trình duyệt. 35](#_Toc45852208)

[Hình 58: Màn hình consolog – nơi ghi lại nhật kí hoạt động của method. 36](#_Toc45852209)

[Hình 59: Demo computed ở màn hình index. 36](#_Toc45852210)

[Hình 60: Demo Computed ở màn hình script. 37](#_Toc45852211)

[Hình 61: Kết quả sau khi chạy demo Computed trên trình duyệt. 37](#_Toc45852212)

[Hình 62: Màn hình consolog – nơi ghi lại nhật kí hoạt động của computed. 38](#_Toc45852213)

[Hình 63: Demo Watcher ở màn hình Index. 39](#_Toc45852214)

[Hình 64: Demo Watcher ở màn hình Script. 39](#_Toc45852215)

[Hình 65: Demo Watcher trên trình duyệt. 39](#_Toc45852216)

[Hình 66: Demo Computed màn hình index. 40](#_Toc45852217)

[Hình 67: Demo Computed màn hình Script. 40](#_Toc45852218)

[Hình 68: Demo Computed khi chạy ở trình duyệt. 40](#_Toc45852219)

[Hình 69: Demo code dùng Setter ở màn hình Index. 41](#_Toc45852220)

[Hình 70: Demo Code dùng Setter ở màn hình Script. 41](#_Toc45852221)

[Hình 71: Demo Setter ở trình duyệt. 42](#_Toc45852222)

[Hình 72: Màn hình Condole.log – nơi kiểm tra giá trị a và b sau khi thay đổi. 42](#_Toc45852223)

[Hình 73: Demo Watcher ở màn hình Index. 43](#_Toc45852224)

[Hình 74: Demo Watcher ở màn hình Script. 43](#_Toc45852225)

[Hình 75: Demo Watcher khi chạy trên trình duyệt. 43](#_Toc45852226)

[Hình 76: Màn hình Console.log – nơi kiểm tra khi thay đổi giá trị cho 1 đối tượng trong Watcher. 44](#_Toc45852227)

[Hình 77: Cú pháp truyền Object vào Class Bindings. 44](#_Toc45852228)

[Hình 78: v-bind:class và class thông thường có thể được gọi chung với nhau mà không bị lỗi. 45](#_Toc45852229)

[Hình 79: Demo truyền Object vào Class Binding ở màn hình Index. 45](#_Toc45852230)

[Hình 80: Demo truyền Object vào Class Binding ở màn hình Script. 45](#_Toc45852231)

[Hình 81: Cú pháp truyền Array vào Class Binding. 46](#_Toc45852232)

[Hình 82: Cú pháp truyền Array vào Class Binding dùng toán tử 3 ngôi. 46](#_Toc45852233)

[Hình 83: Cú pháp kết hợp truyền Object và Array vào Class Binding. 46](#_Toc45852234)

[Hình 84: Khai báo 1 Component với Class có sẵn ở màn hình Script. 46](#_Toc45852235)

[Hình 85: Dùng Component để tạo thêm các Class mới ở màn hình Index. 46](#_Toc45852236)

[Hình 86: Class mới đã xuất hiện ở màn hình Elements khi chạy trình duyệt. 47](#_Toc45852237)

[Hình 87: Khi Binding Class mới dùng Component ở màn hình Index. 47](#_Toc45852238)

[Hình 88: Kết quả khi chạy trình duyệt ở màn hình Elements. 47](#_Toc45852239)

[Hình 89: Cú pháp truyền Object vào Style Binding ờ màn hình index. 47](#_Toc45852240)

[Hình 90: Cú pháp truyền Object vào Style Binding ở màn hình Script. 47](#_Toc45852241)

[Hình 91: Cú pháp gọi bind dành riêng cho Style. 47](#_Toc45852242)

[Hình 92: Cú pháp gọi bind dành riêng cho Style ở màn hình Script. 48](#_Toc45852243)

[Hình 93: Cú pháp truyền Array vào Style Binding ở màn hình Index. 48](#_Toc45852244)

[Hình 94: Tạo form đăng nhập, đăng ký trên <template> chưa có v-if. 48](#_Toc45852245)

[Hình 95:Màn hình localhost sau khi chạy chương trình khi chưa có v-if. 49](#_Toc45852246)

[Hình 96: Dữ liệu khi đã có v-if. 49](#_Toc45852247)

[Hình 97:Màn hình localhost sau khi đã sử dụng v-if để ẩn Form Sign Up. 50](#_Toc45852248)

[Hình 98:Màn hình localhost sau khi đã sử dụng v-if để ẩn Form Sign In… 50](#_Toc45852249)

[Hình 99: Một dữ liệu đơn giản về các dữ liệu bị trùng nhau. 51](#_Toc45852250)

[Hình 100:Dữ liệu sau khi đã thêm phần v-for. 51](#_Toc45852251)

[Hình 101:Trong main.js ta tạo một mảng danh sách tương tự. 52](#_Toc45852252)

[Hình 102: Sau khi chạy chương trình sẽ xuất hiện một danh sách có 3 hàng Nhung-chỉ số index-title-body. 52](#_Toc45852253)

[Hình 103: Code demo listenning to even. 53](#_Toc45852254)

[Hình 104: Giao diện của demo dựa trên code ở hình phía trên. 53](#_Toc45852255)

[Hình 105: Code demo Method Event Handlers. 54](#_Toc45852256)

[Hình 106: Giao diện của demo dựa trên code ở hình phía trên. 54](#_Toc45852257)

[Hình 107: Code demo Methods in inline Handlers. 54](#_Toc45852258)

[Hình 108: Giao diện của code phía trên. 55](#_Toc45852259)

[Hình 109: Code demo v-model đơn giản 57](#_Toc45852260)

[Hình 110: Giao diện của code demo phía trên 57](#_Toc45852261)

[Hình 111 Code demo v-model nhiều dòng. 58](#_Toc45852262)

[Hình 112: Giao diện của code demo phía trên 58](#_Toc45852263)

[Hình 113: Code demo v-model checkbox đơn. 58](#_Toc45852264)

[Hình 114: Giao diện của code demo phía trên. 59](#_Toc45852265)

[Hình 115: Code demo v-model nhiều checkbox. 59](#_Toc45852266)

[Hình 116: Giao diện của code demo phía trên. 59](#_Toc45852267)

[Hình 117: Code demo v-model Radio. 60](#_Toc45852268)

[Hình 118: Giao diện của code demo phía trên. 60](#_Toc45852269)

[Hình 119: Code demo v-model Select. 60](#_Toc45852270)

[Hình 120: Giao diện của code demo phía trên. 61](#_Toc45852271)

[Hình 121: Code demo v-model Select nhìu lựa chọn. 61](#_Toc45852272)

[Hình 122: Giao diện của code demo phía trên. 61](#_Toc45852273)

[Hình 123: Code demo v-model Select render. 62](#_Toc45852274)

[Hình 124: Giao diện của code demo phía trên. 62](#_Toc45852275)

[Hình 125: Code demo Value Bindings checkbox. 62](#_Toc45852276)

[Hình 126: Giao diện của code demo phía trên. 62](#_Toc45852277)

[Hình 127: Code demo Value Bindings radio. 63](#_Toc45852278)

[Hình 128: Giao diện của code demo phía trên. 63](#_Toc45852279)

[Hình 129: Code demo Value Bindings Select. 63](#_Toc45852280)

[Hình 130: Giao diện của code demo phía trên. 63](#_Toc45852281)

[Hình 131: Code demo Modifiers lazy. 64](#_Toc45852282)

[Hình 132: Giao diện của code demo phía trên. 64](#_Toc45852283)

[Hình 133: Code demo Modifiers Number 64](#_Toc45852284)

[Hình 134: Giao diện của code demo phía trên. 64](#_Toc45852285)

[Hình 135: Code demo Modifiers Trim. 65](#_Toc45852286)

[Hình 136: Giao diện của code demo phía trên. 65](#_Toc45852287)

[Hình 137: Cách tải vue-cli. 65](#_Toc45852288)

[Hình 138: Tạo component. 66](#_Toc45852289)

[Hình 139: Đăng ký component toàn cục. 66](#_Toc45852290)

[Hình 140: Sử dụng component toàn cục. 67](#_Toc45852291)

[Hình 141: Đăng ký component cục bộ. 67](#_Toc45852292)

[Hình 142 Gửi dữ liệu cho component con. 68](#_Toc45852293)

[Hình 143: Nhận dữ liệu Props. 69](#_Toc45852294)

[Hình 144: Show dữ liệu. 69](#_Toc45852295)

1. **Giới thiệu VueJS, NodeJS**
   1. **Sự ra đời của NodeJS**

Internet đã ra đời🡪Trình duyệt web đã ra đời ngay sau đó🡪LiveScript được phát minh🡪Đổi tên thành JavaScript🡪Kết hợp với PHP phía máy chủ 🡪Các ứng dụng web đã chính thức ra đời🡪Các trình duyệt không thể theo kịp tốc độ🡪Công nghệ trình duyệt web đang lỗi thời.



Hình 2: JavaScript kết hợp với PHP xây dựng các ứng dụng Web có thể truy cập qua trình duyệt mà không cần lãng phí một phần mềm nào.

Google đã đến giải cứu: Họ đã phát minh lại trình duyệt🡪 Xây dựng trình thông dịch JavaScript thực tế như một dự án độc lập🡪Gọi công cụ JavaScript mới này là “V8”.



Hình 3: Logo ngôn ngữ "V8".

Năm 2009🡪Google Chrome là một trình duyệt web tuyệt vời chủ yếu là do động cơ V8🡪Một nhân viên của Google đã đưaV8 đến máy chủ🡪Gọi nó là NodeJs🡪 Cùng một JavaScript ở cả phía máy chủ và máy khách không gì khác hơn là một phép màu.



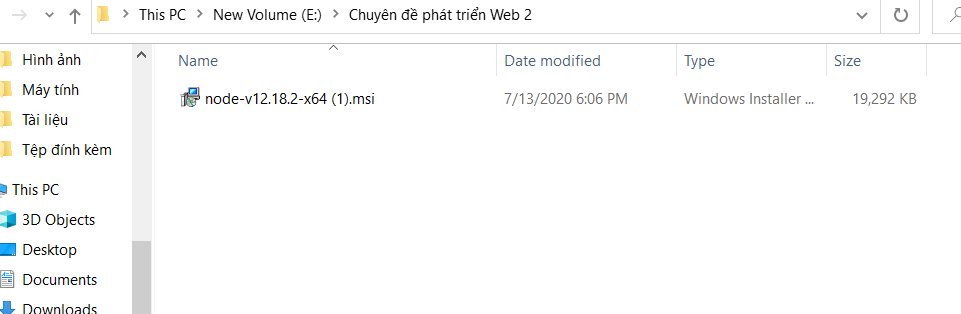
Hình 4: Logo NodeJS.

* 1. **Cài đặt NodeJs**
* Bước 1**:** Nhấn vào đường link<https://nodejs.org/en/download/> để tải Nodejs.



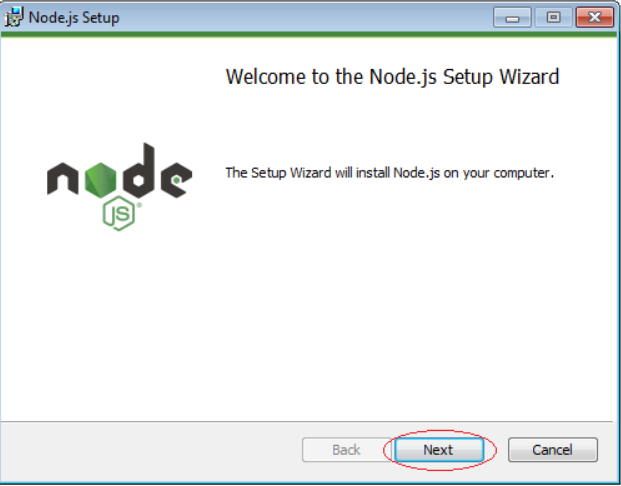
Hình 5: Sau khi nhấn vào link sẽ hiện ra bảng như trên.

* Bước 2: Sau khi download xong sẽ hiện lên bảng sau:



Hình 6: Thư mục chứa NodeJs vừa download.

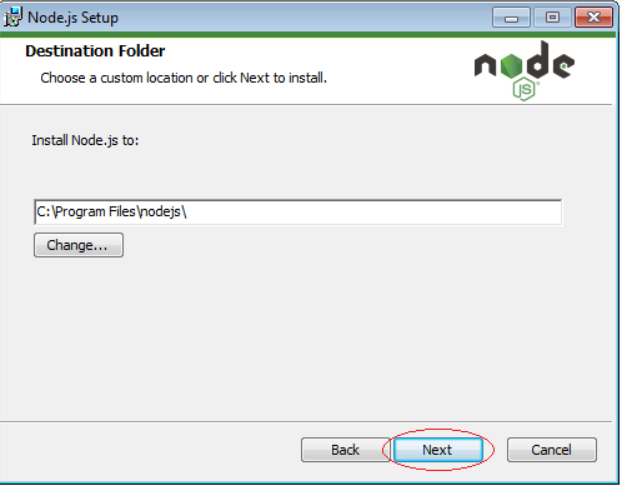
* Bước 3: Chấp nhận các tùy chọn mặc định và nhấn "Next … Next" cho tới bước cuối cùng.



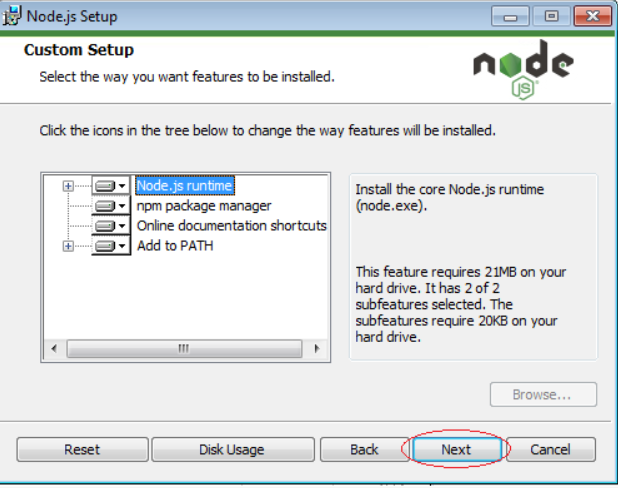
Hình 7: Nhấn Next bắt đầu quá trinh cài đặt.



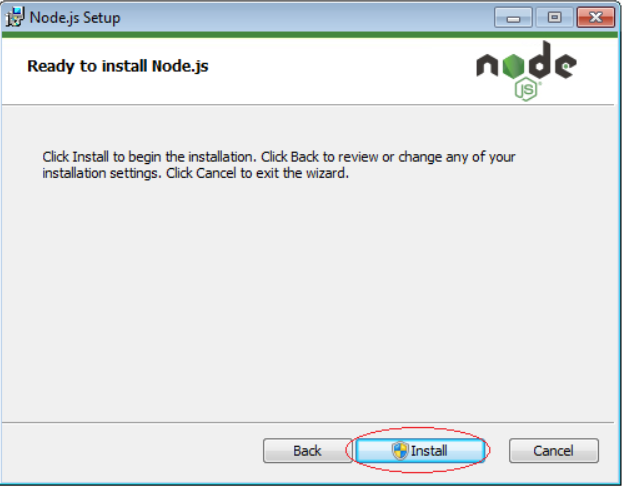
Hình 8: Tick chọn I Accept… rồi nhấn Next.



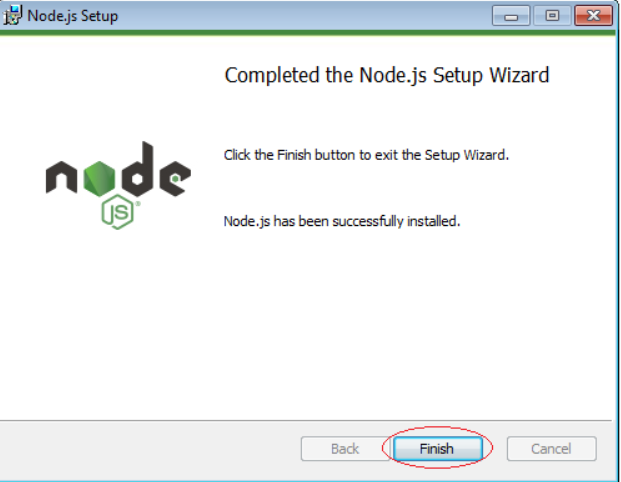
Hình 9: Tiếp tục nhấn Next.



Hình 10: Tiếp tục Next.



Hình 11: Nhấn Install để cài đặt.

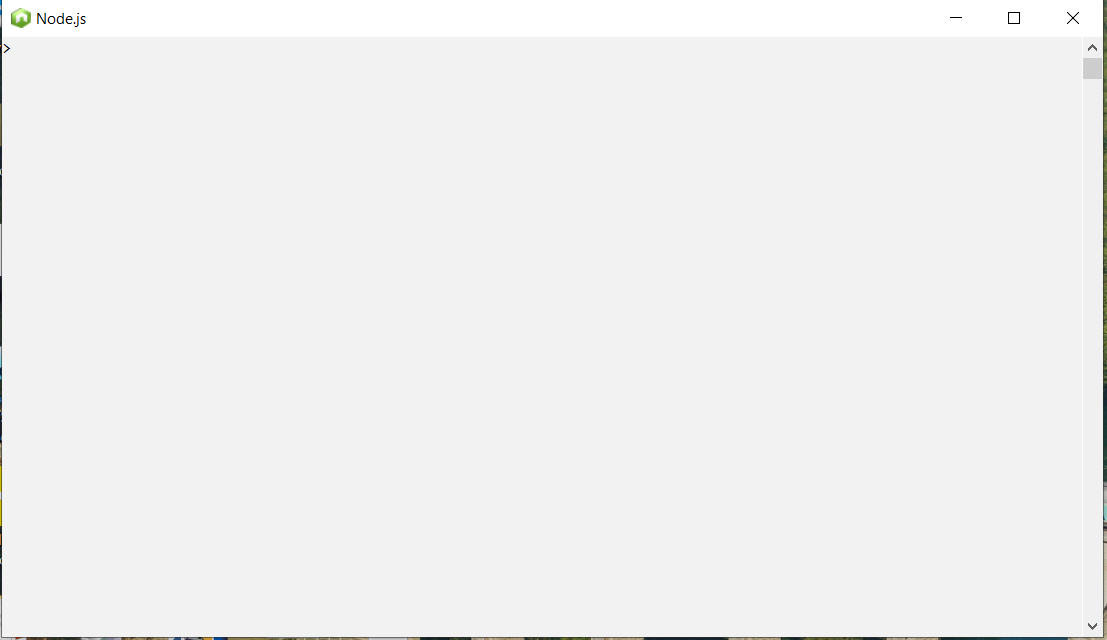


Hình 12: Bấm Finish để kết thúc cài đặt.

* Bước 4: Vào Icon Node ở màn hình Desktop, double click để chạy thành công Node.



Hình 13: Click vào icon Node ở màn hình Desktop để khởi động Node.



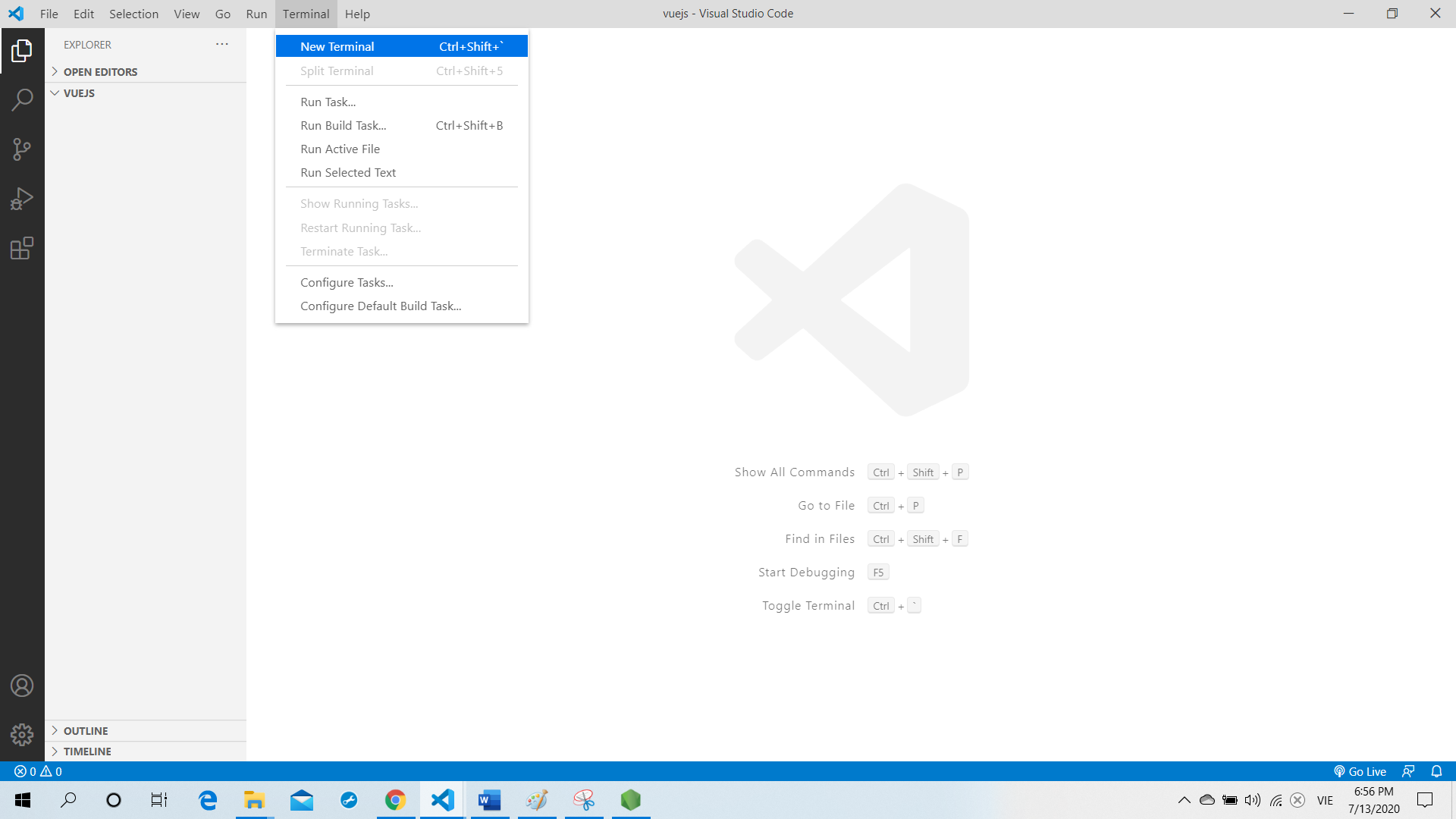
Hình 14: Sau khi click chuột giao diện khởi động Node hiện ra và ta chỉ cần tắt nó là đã thành công bước cài đặt Nodejs.

* 1. **VueJS**
* VueJS là một Framework được viết hoàn toàn bằng JavaScript ra đời sau Angular và ReactJS.
* VueJS chỉ tập trung vào phần giao diện nên làm một Project rất dễ dàng.
  1. **Cách cài đặt Vuejs**
* **Cách 1: Cài đặt Vue empty:**
* Bước 1: Chạy cmd gõ lệnh: **npm install vue**
* Bước 2: Tạo file index.html và thêm vào đoạn script



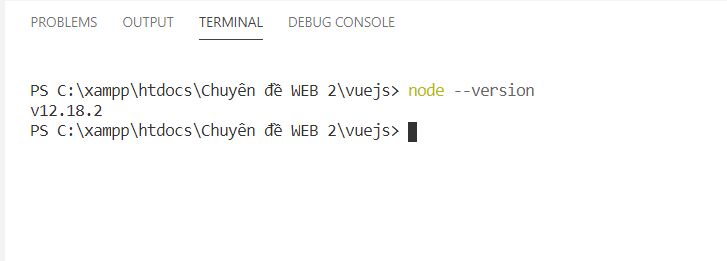
Hình 15: Đường link hỗ trợ hàm của vue.

* Bước 3: Kết quả
* **Cách 2: Cài đặt Vue-Cli version 2.9(tham khảo)**
* Bước 1: Trước khi cài đặt VueJs ta mở ứng dụng NodeJS và Visual Studio Code tạo một thư mục và nhấn vào Terminal.



Hình 16: Ảnh minh họa.

* Bước 2: Kiểm tra version của NodeJs bằng câu lệnh **node –version.**



Hình 17: Version hiện lên sau khi kiểm tra.

* Bước 3: Dùng các tổ hợp câu lệnh NPM(thư viện của NodeJs) để cài đặtVueJs.
* Chạy dòng lệnh: **npm i -g vue-cli.**

Giải Thích: npm i: là npm install dùng để cài đặt các module trong nodejs

-g: là global dùng để thông báo vue-cli là đối tượng toàn cục cho các module

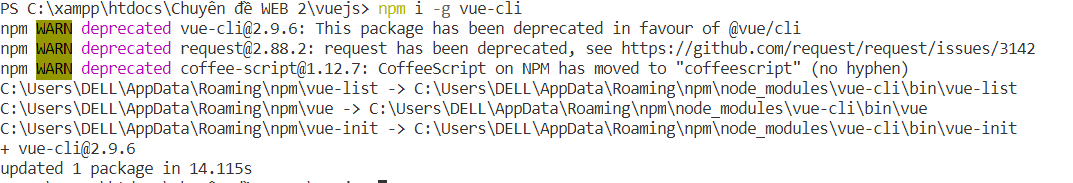
vue-cli: là cài đặt thư viện vue-cli vào project của chúng ta

Update: package **vue-cli** đã thay đổi thành **@vue/cli**

Để cài đặt vue mới nhất ta chạy các câu lệnh sau:

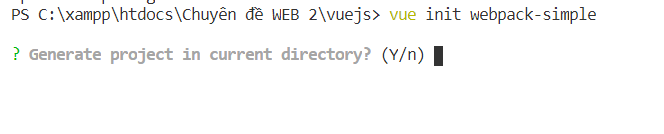
1: **npm uninstall vue-cli -g**

2: **npm install -g @vue/cli**



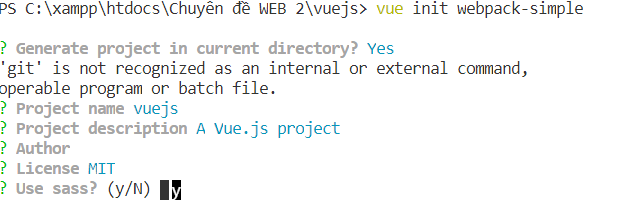
Hình 18: Màn hình sau khi chạy dòng lệnh npm i -g vue-cli.

* Sau đó chạy dòng lệnh: **vue init webpack-simple**.



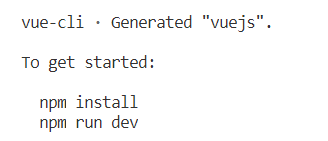
Hình 19: Màn hình sau khi chạy dòng lệnh vue init webpack-simple sẽ xuất hiện các câu hỏi.

* Enter enter cho đến khi hiện ra dòng Use sass --> chọn Yes.



Hình 20: Ảnh minh họa.

* Sau đó sẽ hiện lên các câu lệnh này



Hình 21: Hai câu lệnh xuất hiện khi nhập các câu lệnh trên.

* Bước 4: Chạy câu lệnh **npm install**.



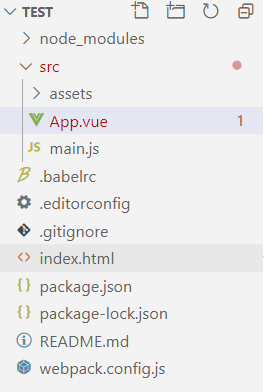
Hình 22: Màn hình sau khi chạy npm install.

* Bước 5: Chạy câu lệnh **npm run dev**.

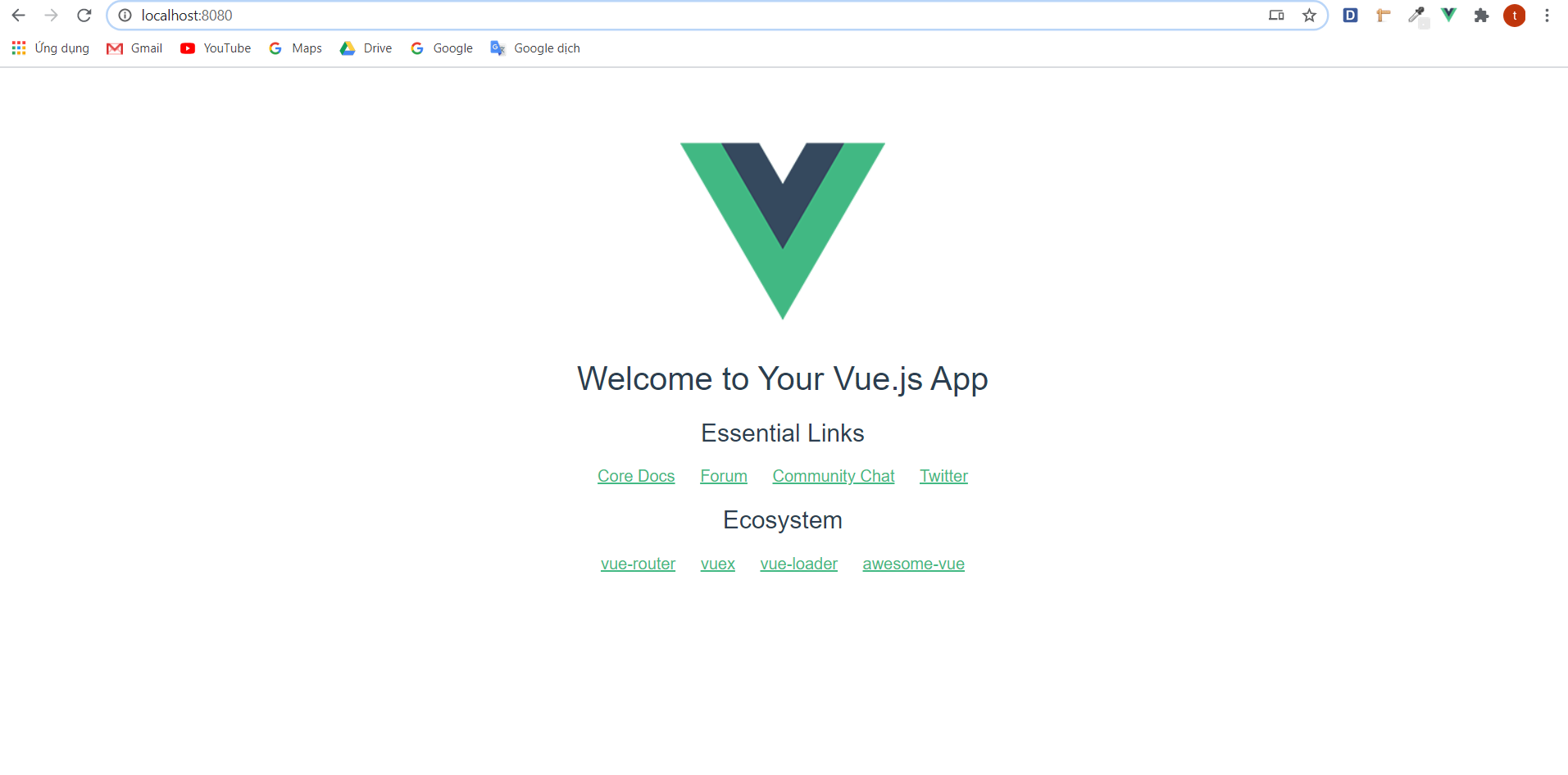


Hình 23: Màn hình sau khi chạy npm run dev.

Sau khi chạy xong sẽ xuất hiện một source chứa một cấu trúc module.



Hình 24: Cấu trúc VueJs sau khi cài đặt.



Hình 25: Trang chủ khi chạy

1. **Vue Instance**
   1. **Creating a Vue Instance**

**Cách 1:**



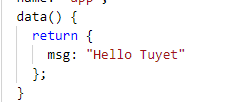
Hình 26: Hình ảnh tạo Vue Instance.

Để tạo ra “Hello Tuyet” ta cần file script:

Name: đặt tên gì cũng được nhưng không được có dấu cách.

Name: “app”.

Sau đó là object.



Hình 27: VD demo tạo đoạn text script cơ bản của Vue.

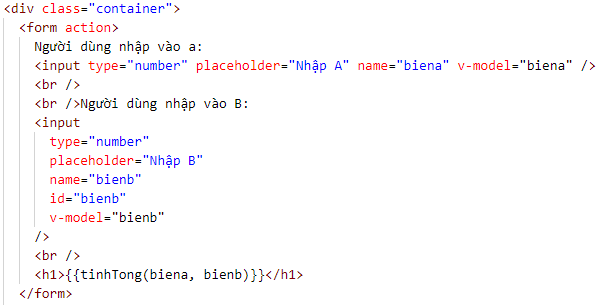
Trong data() tạo 1 đoạn test

Sau đó vào <template>

Trong thẻ <h1> {{nơi gọi tên đoạn text đó}} </h1>

**Cách 2:**

* 1. **Data and Method**



Hình 28: Code HTML tính tổng.

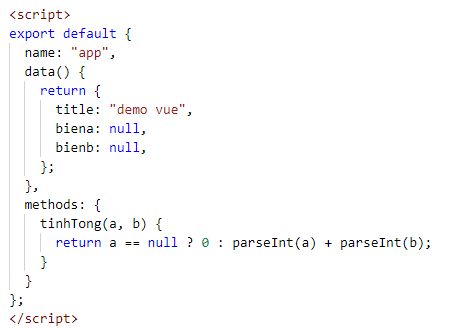
**Ý tưởng:**

Cho 2 thẻ input:

+ input thứ 1: Cho người dùng nhập vào số tự nhiên A

+ input thứ 2: Cho người dùng nhập vào số tự nhiên B

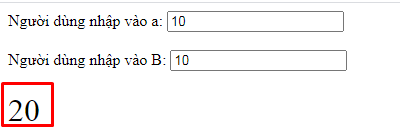
Sau đó cho A + B = kết quả.



Hình 29: Xử lý phép cộng.

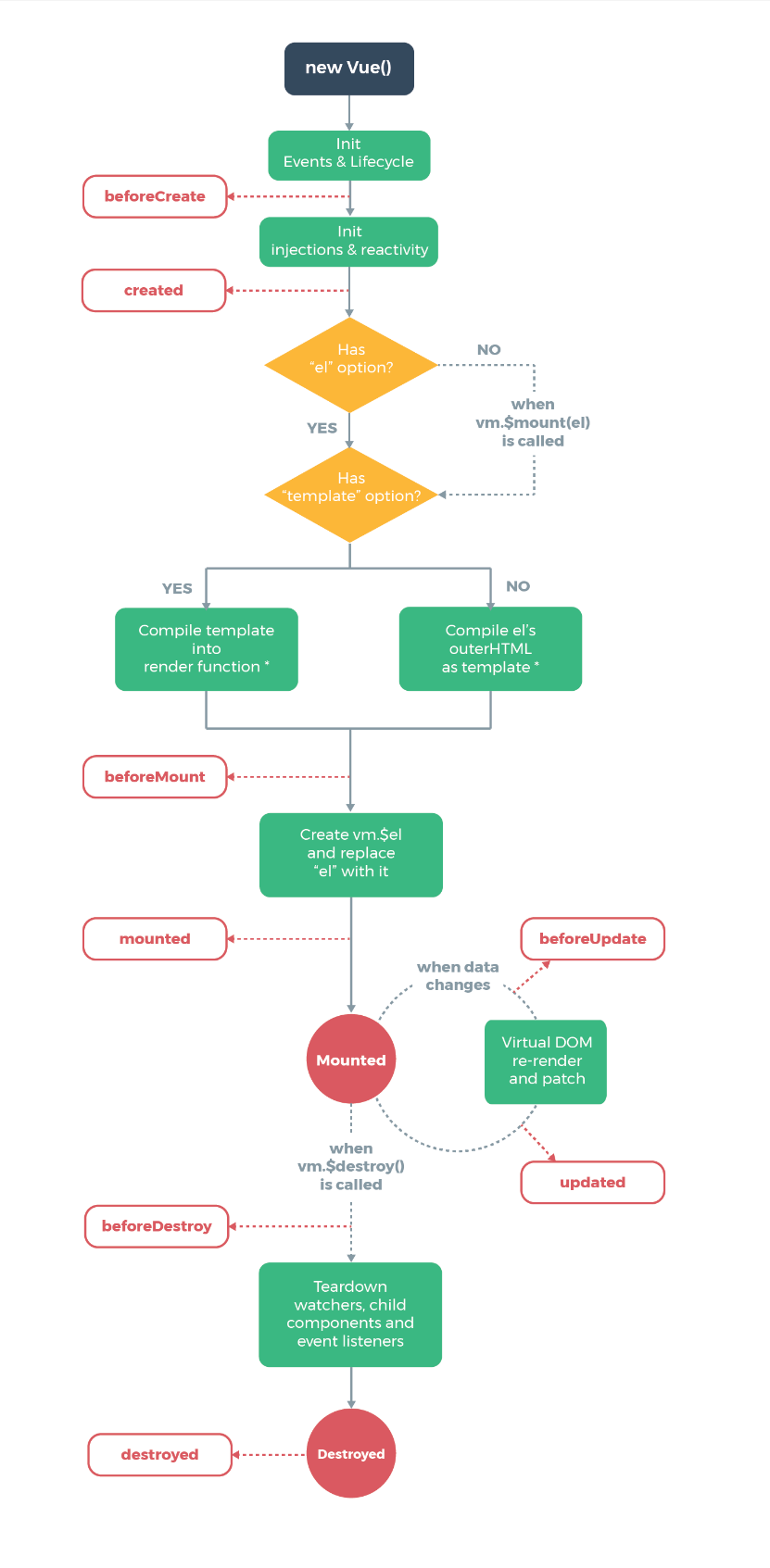
* Khai báo biến a và b đều null. Sau đó tạo 1 hàm method tính tổng 2 số, lưu ý: sử dụng parseInt để chuyển 1 chuổi thành số.

Kết quả:

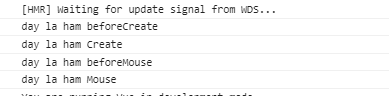


Hình 30: Kết quả xử lý phép cộng.

* 1. **Lifecyle Diagram, Intance Lifecyle Hook**



Hình 31: Vòng đời Vue.



Hình 32: Màn hình Consolog: Nơi chứa code ở trình duyệt.

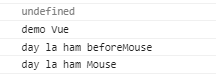


Hình 33: Code demo ở màn hình Script.

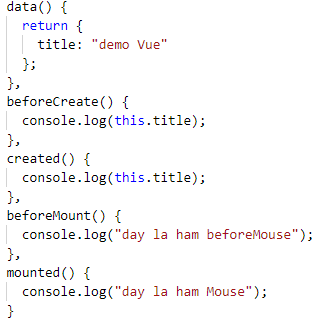
**Mô tả:**

Chạy console.log để hiển thị luồng xử lý của Vue.

* **Hàm Created:** dùng để nhận API, dữ liệu chỉ được nhận trong hàm created.



Hình 34: Màn hình Console.log - nơi chứa code tại trình duyệt.



Hình 35: Code demo đơn giản tại file Script.

* **Hàm Mounted:** là để hiển thị cấu trúc HTML, DOM ra được cấu trúc HTML.
* **Hàm beforeMouted:** không DOM ra được cấu trúc HTML.

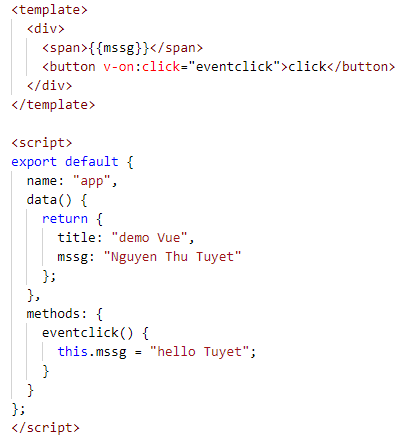


Hình 36: Sửa color của text thành “yellow”.



Hình 37: Kết quả sau khi sửa màu sắc ở trình duyệt.

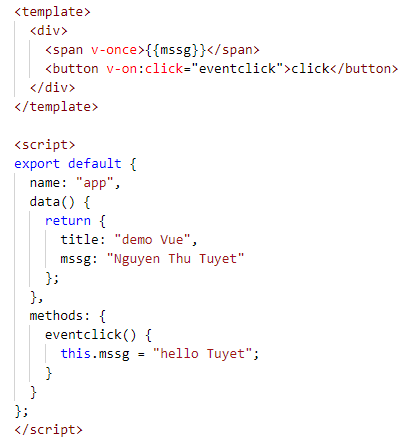
1. **Template Syntax**
   1. **Interpolation**
      1. **Text**



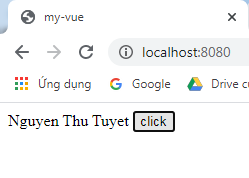
Hình 38: Ví dụ text của phần Interpolation.

* Cho 1 đoạn code có nút button khi nhấn vào thì chuyển thành tên khác. Không có v-once thì sẽ thực hiện sự kiện click.

Khi thêm v-once vào đoạn <span> khi click vào sẽ không có thay đổi.



Hình 39: Đoạn code thêm v-once vào <span>.



Hình 40: Bắt sự kiện click nhưng không thay đổi.

* + 1. **Raw HTML**



Hình 41: Ví dụ raw html của phần Interpolation.

* V-html: dùng để xuất 1 đoạn html.
* Đầu tiên khai báo v-html trong thẻ <h1>:

v-html= “rawhtml”

Sau đó xử lý trong script:

Rawhtml: là phần code nội dung và kèm với style cho đoạn code.

* + 1. **Attributes**



Hình 42: Ví dụ Attributes của phần Interpolation.

* v-bind: truyền giá trị cho các thuộc tính trong thẻ.
* Cho 1 đoạn button:

<button v-bind:className="className" v-html="rawhtml"></button>

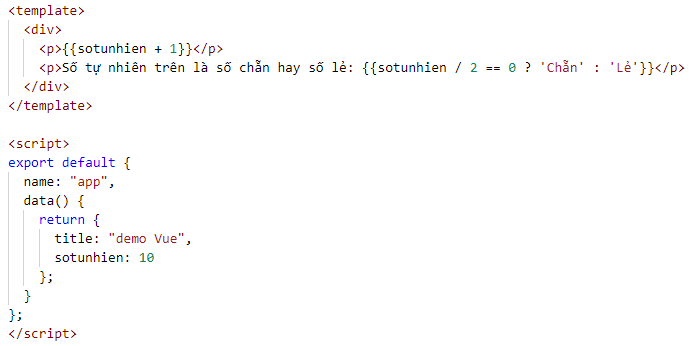
Phần script xử lý:

Ta khai báo:

Classname: “text-red”: chữ trong button chuyển sáng màu đỏ.

Rawhtml: đoạn text “Hoa Tuyet”.

* + 1. **Using JavaScript Expressions**



Hình 43: Ví dụ về Using JavaScript Expressions của phần Interpolation.

* Áp dụng phép tình toán vào trong code.
* Cho 1 số tự nhiên và dùng phép cộng thêm 1. Kết quả của số tự nhiên chia cho 2 để phân biệt số tự nhiên chẵn hay lẽ.
  1. **Directives**
     1. **Arguments**



Hình 44: Demo đơn giản về v-bind:href.

* Cho 1 thẻ a khi click vào chữ “Click vao day”, sẽ đi tới trang Vuejs.org.
  + 1. **Dynamic Arguments**



Hình 45: Lỗi thường gặp khi không dùng v-bind.

Nếu không dùng v-bind thì sẽ xuất hiện lỗi như hình.



Hình 46: Sau khi dùng v-bind và kết quà thành công.

Nếu thêm v-bind vào thì sẽ hiện thị được hình ảnh.

Ví dụ:



Hình 47: Demo đơn giản ở màn hình Script.

Nhấn vào nút thay đổi hình ảnh, hình ảnh ban đầu sẽ được thay đổi sang một hình ảnh khác:

Hình ảnh trước khi thay đổi:



Hình 48: Click nào nút “Thay đổi hình ảnh” để thay đổi ảnh.

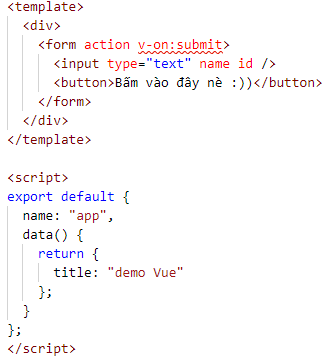
Hình ảnh sau khi thay đổi:



Hình 49: Kết quả sau khi đã click vào nút "Thay đổi hình ảnh".

* + 1. **Modifiers**

VD: Cho 1 thẻ input cho người dùng nhập vào và có một nút button để xem load trang:



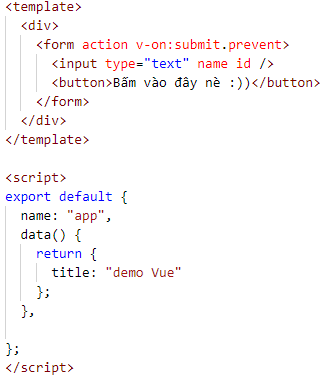
Hình 50: Demo tạo 1 nút input để load trang.

Trường hợp nếu không dùng .prevent:



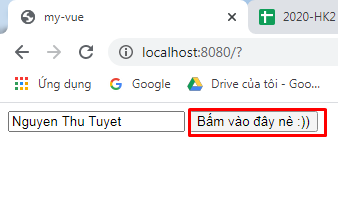
Hình 51: Demo trường hợp không dùng .prevent.

Trường hợp nếu thêm .prevent:



Hình 52: Demo tạo nút load trang có dùng .prevent.

Khi dùng prevent sẽ không còn load lại trang nữa:



Hình 53: Kết quả hiển thị trên trình duyệt khi dùng .prevent.

* 1. **Shorthands**
     1. **V-bind Shorthands**

Kí hiệu viết tắt của v-bind: là dấu “**:**”

* + 1. **V-on Shorthands**

Kí hiệu viết tắt của v-on: là dấu “**@**”

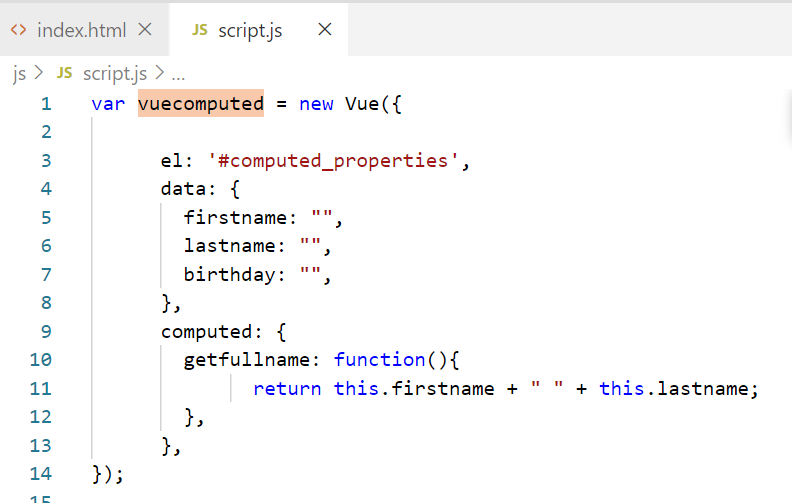
1. **Tìm hiểu về Computed Properties and Watchers**
   1. **Computed Properties (thuộc tính):**

Được sinh ra sau khi thực hiện 1 hàm (Function). Thường thì ta có thể viết trực tiếp tại template bằng cách gọi trực tiếp trong cặp dấu .

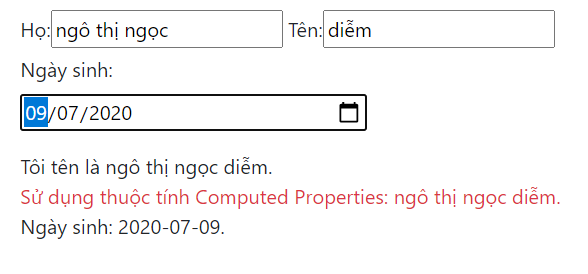
* Rất bất tiện cho việc bảo trì nếu đó là phép tính phức tạp 🡪 Có 1 giải pháp tối ưu là dùng Computed Properties.
* Demo:



Hình 54: Demo computed-properties ở màn hình index.



Hình 55: Demo computed-properties ở màn hình script.js.

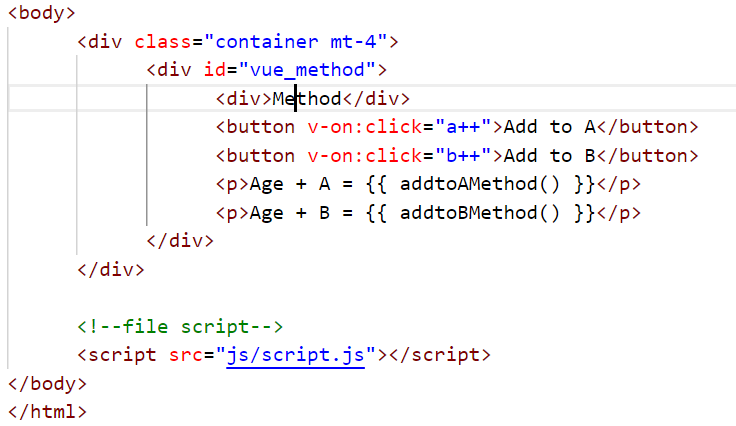


Hình 56: Kết quả khi chạy trên màn hình trình duyệt.

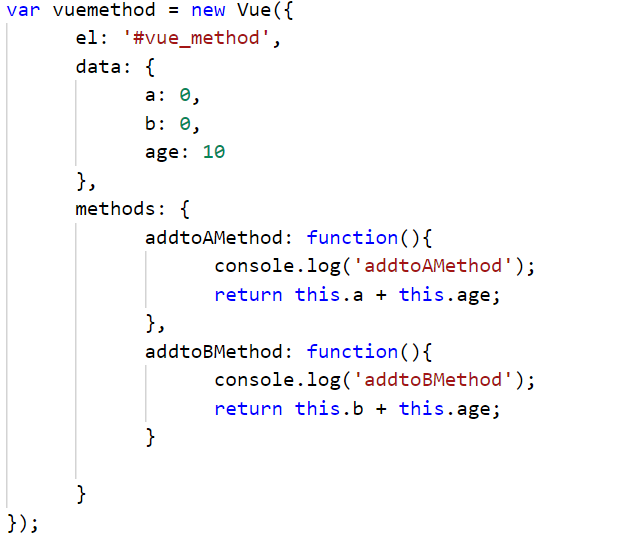
* Computed Properties thuận tiện cho việc sửa chữa và bảo trì code.
  + 1. **So sánh Computed Caching và Methods:**
  + Khác nhau:

|  |  |
| --- | --- |
| Computed | Method |
| Vì Computed không nhận tham số đầu vào nên nó chỉ nhận dữ liệu có sẵn trong data của component 🡪 Vì vậy không thể đặt cặp dấu  đằng sau thuộc tính Computed. | Nhận tham số đầu vào🡪 Đặt cặp dấu  đằng sau thuộc tính Method. |
| Chỉ được cập nhật lại khi các biến phụ thuộc của nó thay đổi. | Được cập nhật bất cứ lúc nào khi được gọi. |
| Chứa bộ nhớ Cached nên các kết quả sẽ được ghi lại cho lần dùng sau và sẽ gọi ra khi cần thiết🡪 Tiết kiệm bộ nhớ hệ thống. | Mỗi lần gọi là sinh ra một kết quả mới 🡪 gây tiêu tốn tài nguyên. |

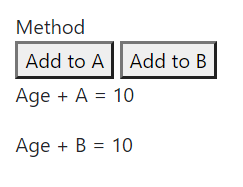
* Demo về sự khác nhau giữa Computed và Method:
* Method:



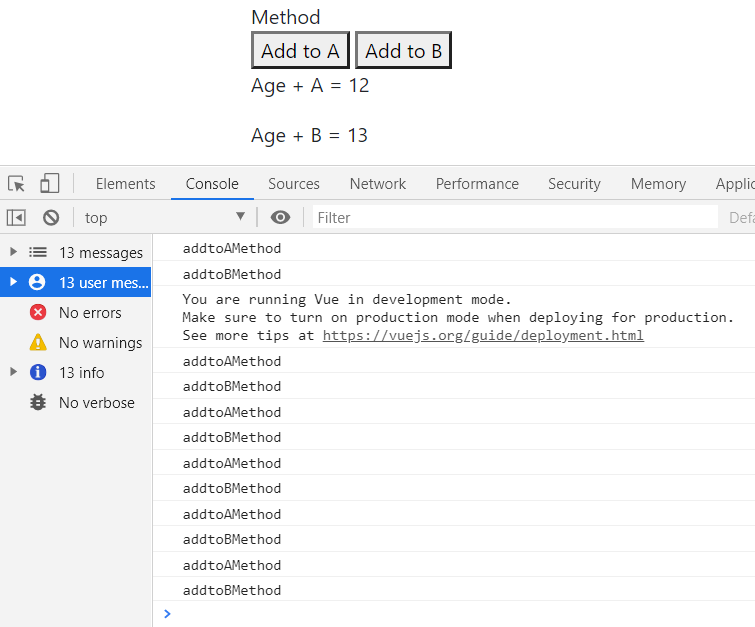
Hình 57: Demo Method ở màn hình index.



Hình 58: Demo method ở script.

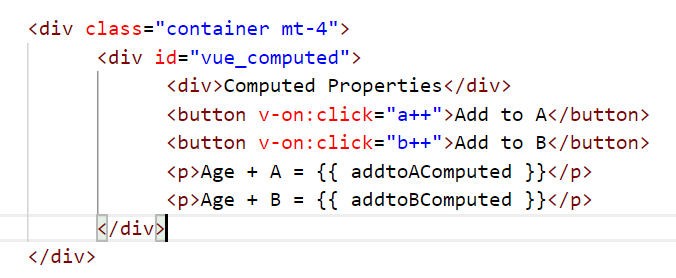


Hình 59: Kết quả sau khi demo method trên trình duyệt.

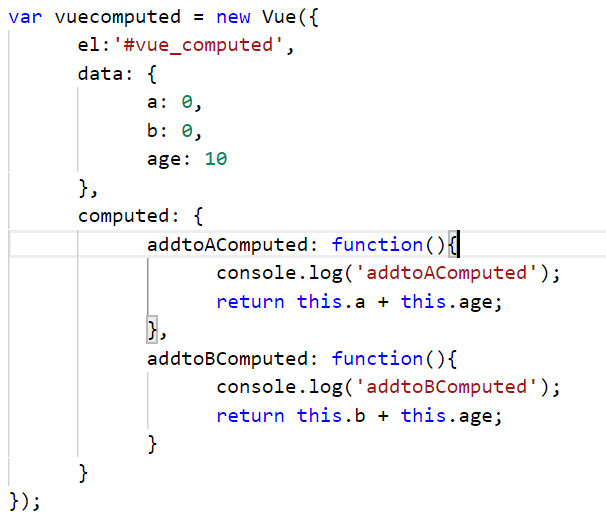


Hình 60: Màn hình consolog – nơi ghi lại nhật kí hoạt động của method.

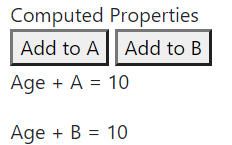
* Khi ta nhấp nút addtoAMethod: tất cả các phương thức được gọi (addtoAMethod và addtoBMethod) cũng được thực thi trong khi đó, ta không nhấp vào nó, nhưng giá trị addtoBMethod vẫn không thay đổi. Tương tự như việc gọi addtoBMethod cũng mang kết quả tương tự 🡪 Gây tiêu tốn tài nguyên và chậm tốc độ load trang web.
* Computed:



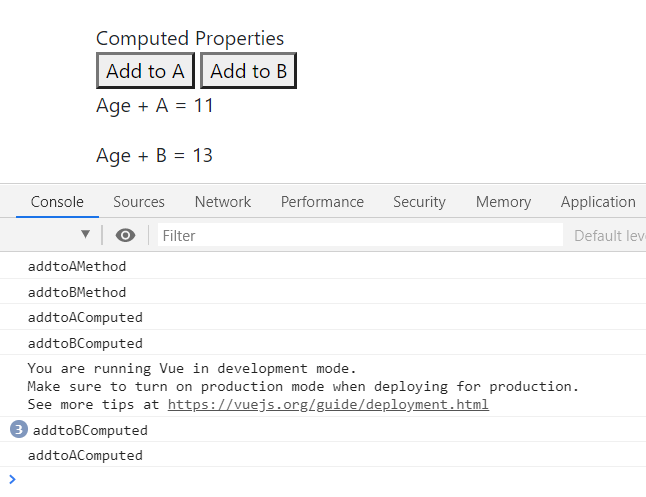
Hình 61: Demo computed ở màn hình index.



Hình 62: Demo Computed ở màn hình script.



Hình 63: Kết quả sau khi chạy demo Computed trên trình duyệt.

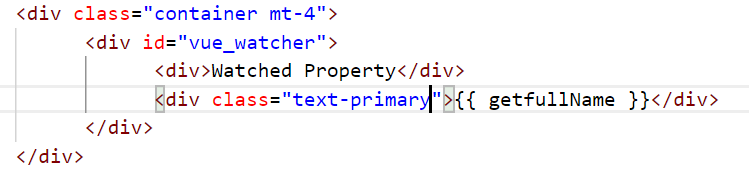


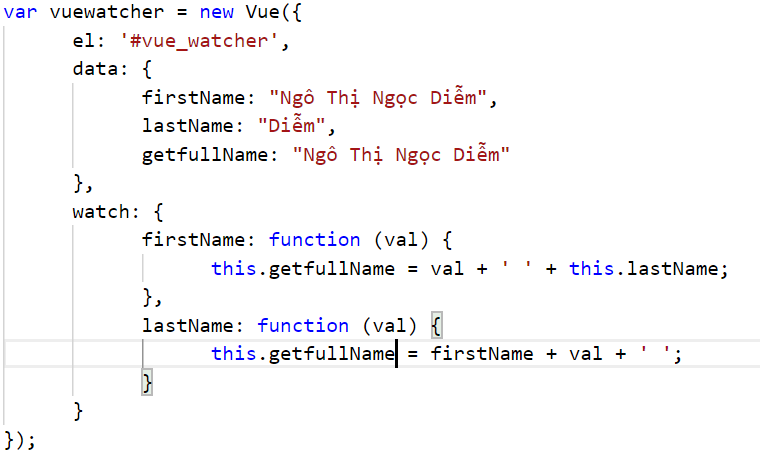
Hình 64: Màn hình consolog – nơi ghi lại nhật kí hoạt động của computed.

* Khi ta nhấp vào nút addtoBComputed, chỉ có phương thức tương ứng được gọi, vì vậy phương thức nào không được tác động sẽ không thay đổi giá trị của nó 🡪 Tiết kiệm tài nguyên, và có thể gọi các phương thức cũ khi cần sử dụng lại.
  + 1. **So sánh Computed với Watched Property**
  + Khác nhau: Dùng Computed / Watcher khi:

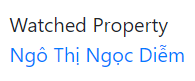
|  |  |
| --- | --- |
| Computed | Watcher |
| Muốn giảm tên một thuộc tính phức tạp, lồng nhau thành 1 tên dễ đọc và dễ sử dụng  Được cập nhật khi thuộc tính dữ liệu ban đầu bị thay đổi  Thuộc tính dữ liệu đạt đến 1 giá trị cụ thể, sau đó được sử dụng vào một việc cụ thể  🡪Cần tham chiếu 1 giá trị từ template | Tạo dữ liệu mới từ dữ liệu hiện có  Sắp xếp hoặc thao tác 1 nhóm dữ liệu lớn 🡪Sử dụng các giá trị trực tiếp trong template |

* Demo về sự khác nhau giữa Watcher và Computed:
* Watcher:

Hình 65: Demo Watcher ở màn hình Index.



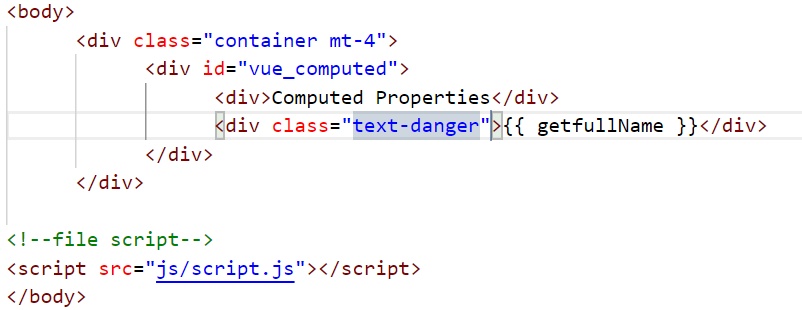
Hình 66: Demo Watcher ở màn hình Script.



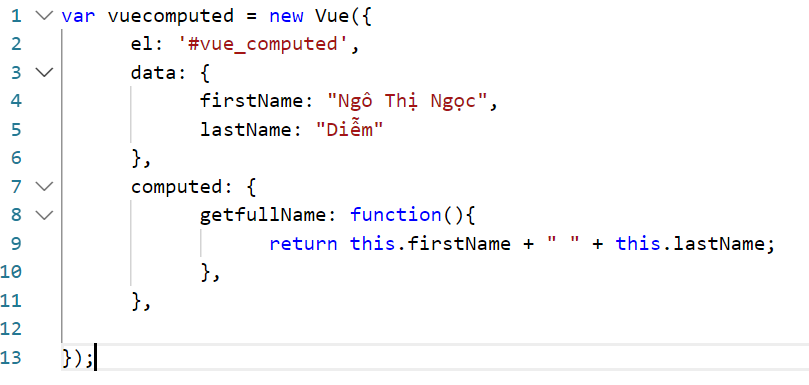
Hình 67: Demo Watcher trên trình duyệt.

🡪Cũng là yêu cầu in ra màn hình FullName nhưng sử dụng Computed lại thật sự khác biệt:

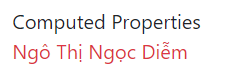
* Computed:



Hình 68: Demo Computed màn hình index.



Hình 69: Demo Computed màn hình Script.

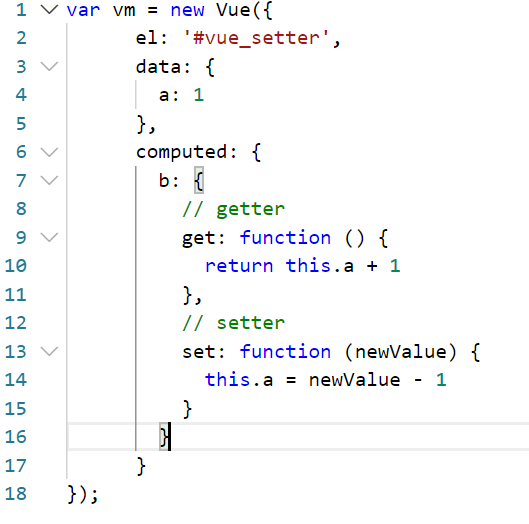


Hình 70: Demo Computed khi chạy ở trình duyệt.

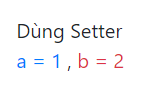
* Mặc dù 2 demo cho ra cùng một kết quả nhưng ta sẽ thấy code của Computed ngắn gọn và dễ hiểu hơn so với Watcher 🡪 Vì vậy, khi code, tránh lạm dụng Watcher quá nhiều dẫn đến tình trạng thừa code không mong muốn.
  + 1. **Computed Setter**
* Demo code khi sử dụng Setter:



Hình 71: Demo code dùng Setter ở màn hình Index.



Hình 72: Demo Code dùng Setter ở màn hình Script.



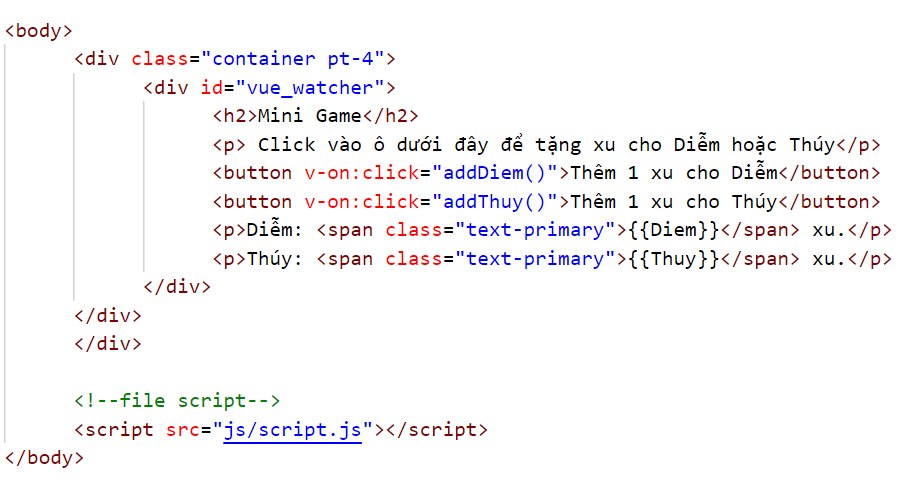
Hình 73: Demo Setter ở trình duyệt.

* Ta thực hiện kiểm tra xem a và b có thay đổi giá trị khi ta thay đổi ở màn hình kiểm tra Console.log:

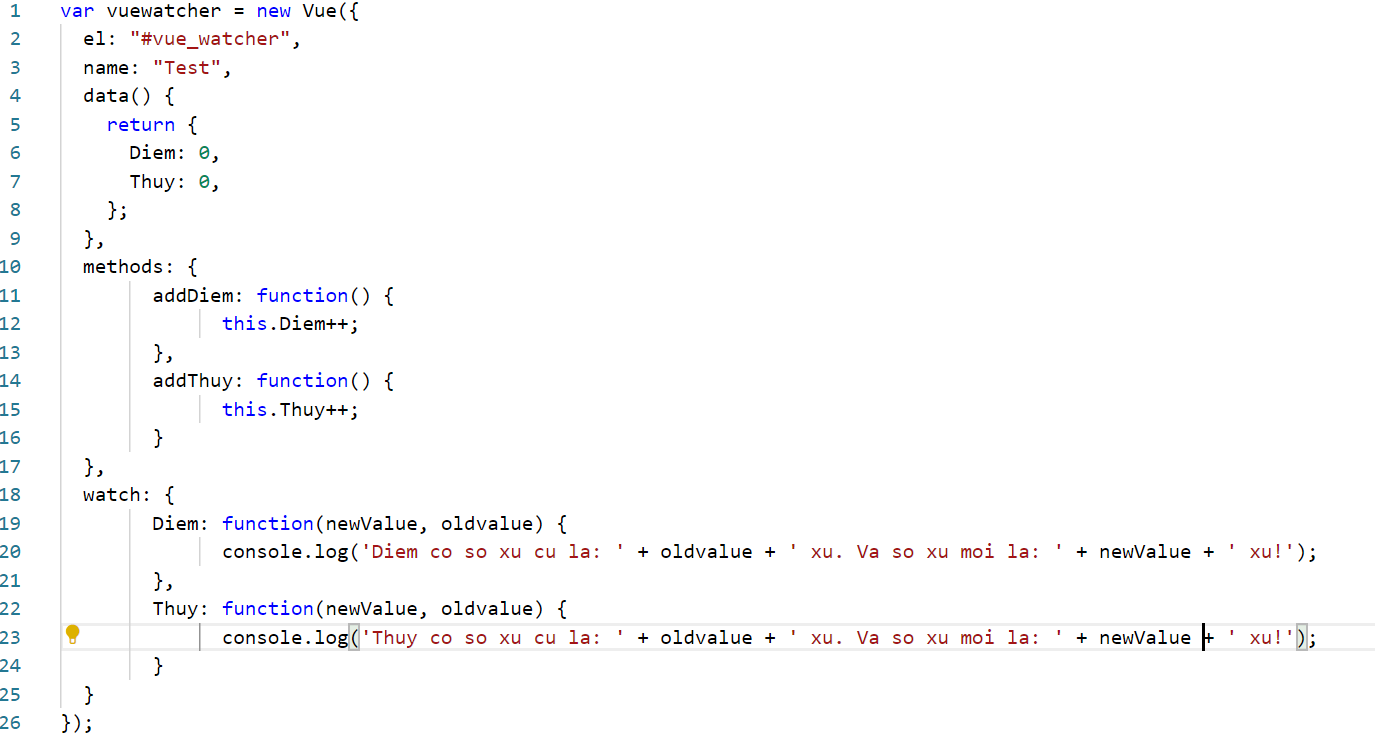


Hình 74: Màn hình Condole.log – nơi kiểm tra giá trị a và b sau khi thay đổi.

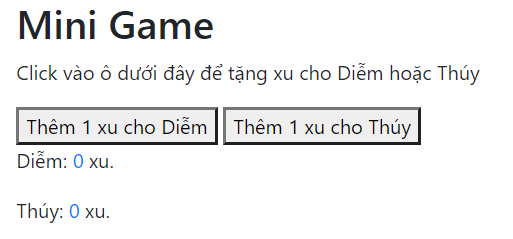
* 1. **Watchers:**
* Demo Watcher:



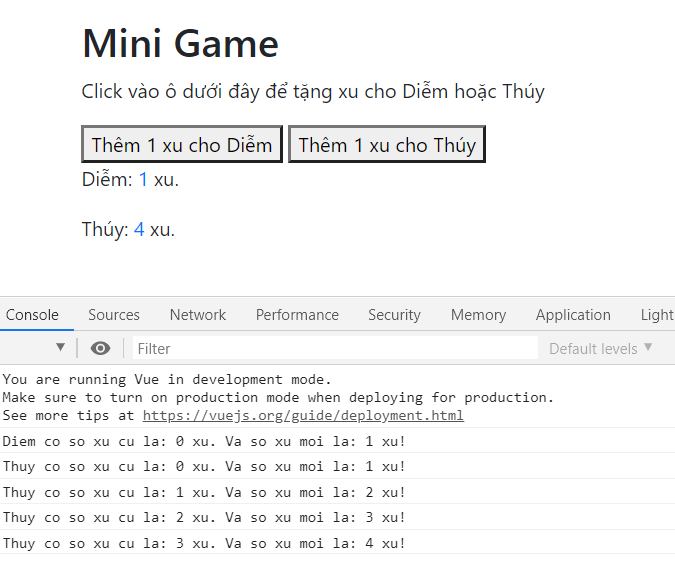
Hình 75: Demo Watcher ở màn hình Index.



Hình 76: Demo Watcher ở màn hình Script.



Hình 77: Demo Watcher khi chạy trên trình duyệt.



Hình 78: Màn hình Console.log – nơi kiểm tra khi thay đổi giá trị cho 1 đối tượng trong Watcher.

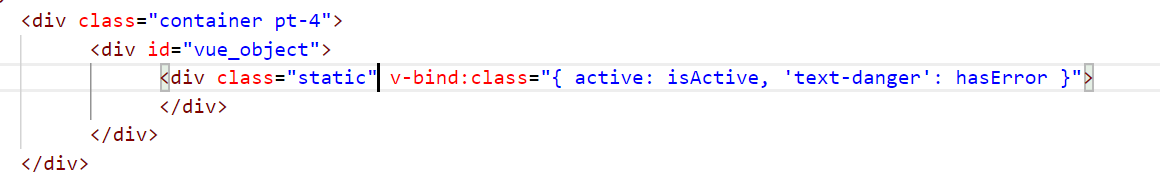
1. **Tìm hiểu Class and Style Bindings**

* Class Bindings và Style Bindings đều thuộc Cơ chế ràng buộc dữ liệu (Binding data 🡪 Nó là một directive thể hiện sự ràng buộc giữa các thuộc tính trong HTML như: Class, Style và Data trong Vue).
* VueJs cung cấp cho ta Class Bindings và Style Bindings bằng cú pháp *v-bind:class* và *v-bind:style.*
  1. **Binding HTML Classes**
     1. **Object Syntax**
* Truyền định dạng kiểu Object Syntax để bật và tắt các Class.
* Cú pháp:



Hình 79: Cú pháp truyền Object vào Class Bindings.

* Theo cú pháp trên, class Active sẽ được áp dụng tùy theo tính đúng sai của thuộc tính isActive. Có thể bật tắt nhiều Class bằng cách dùng nhiều Filed(trường) trong Object. Lưu ý Class thông thường và v-bind:class có thể dùng chung được với nhau theo cách như sau:

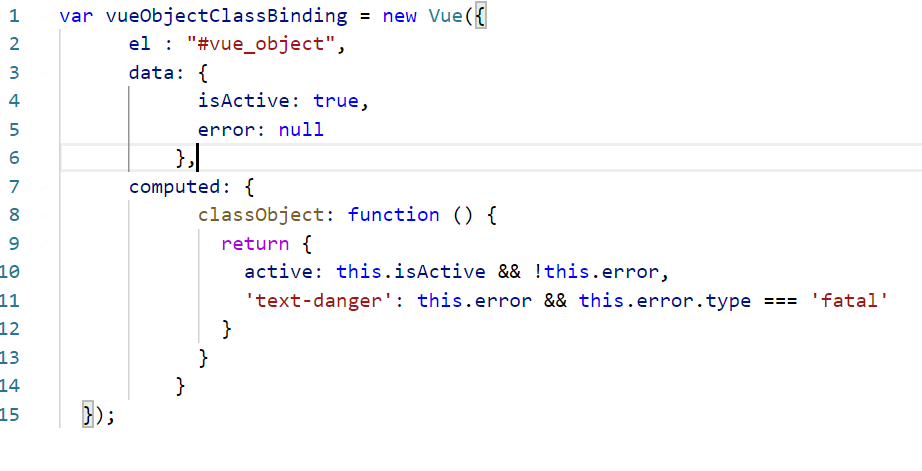


Hình 80: v-bind:class và class thông thường có thể được gọi chung với nhau mà không bị lỗi.

* Demo về Class Binding dùng Object truyền vào:



Hình 81: Demo truyền Object vào Class Binding ở màn hình Index.



Hình 82: Demo truyền Object vào Class Binding ở màn hình Script.

* + 1. **Array Syntax**
* Truyền Class Binding ở dạng Array(mảng) để áp dụng 1 danh sách Class.
* Cú pháp 1:



Hình 83: Cú pháp truyền Array vào Class Binding.

* Cú pháp 2: Dùng toán tử 3 ngôi(bật tắt Class theo điều kiện có ít 1 Class danh sách).



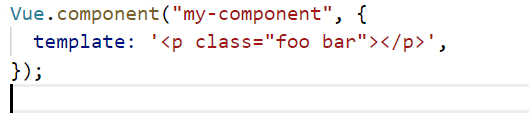
Hình 84: Cú pháp truyền Array vào Class Binding dùng toán tử 3 ngôi.

* Cú pháp 2 này khá dài dòng nếu phát sinh nhiều Class theo điều kiện 🡪 Cú pháp 3 ngắn gọn hơn bằng cách sử dụng lồng ghép Object và Array:



Hình 85: Cú pháp kết hợp truyền Object và Array vào Class Binding.

* + 1. **With Components:**
* Khi sử dụng 1 thuộc tính Class cho Component tùy biến, các Class này sẽ được thêm vào root của Component tương ứng, các Class có sẵn vẫn được giữ nguyên.



Hình 86: Khai báo 1 Component với Class có sẵn ở màn hình Script.

* Đồng thời, dùng Component để khai báo các Class khác:



Hình 87: Dùng Component để tạo thêm các Class mới ở màn hình Index.



Hình 88: Class mới đã xuất hiện ở màn hình Elements khi chạy trình duyệt.



Hình 89: Khi Binding Class mới dùng Component ở màn hình Index.

* Khi isActive đúng, Elements chạy lên từ trình duyệt sẽ có dạng:

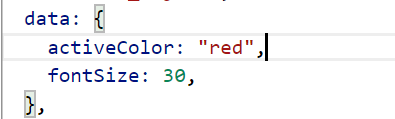


Hình 90: Kết quả khi chạy trình duyệt ở màn hình Elements.

* 1. **Binding Inline Styles**
* Các thuộc tính CSS được truyền trực tiếp vào thẻ HTML.
  + 1. **Object Syntax**
* Cú pháp 1:



Hình 91: Cú pháp truyền Object vào Style Binding ờ màn hình index.



Hình 92: Cú pháp truyền Object vào Style Binding ở màn hình Script.

* Cú pháp 2: Gọn hơn bằng cách Binding vào 1 Object dành riêng cho Style để code HTML được gọn gàng và dễ hiểu.



Hình 93: Cú pháp gọi bind dành riêng cho Style.



Hình 94: Cú pháp gọi bind dành riêng cho Style ở màn hình Script.

* + 1. **Array Syntax**



Hình 95: Cú pháp truyền Array vào Style Binding ở màn hình Index.

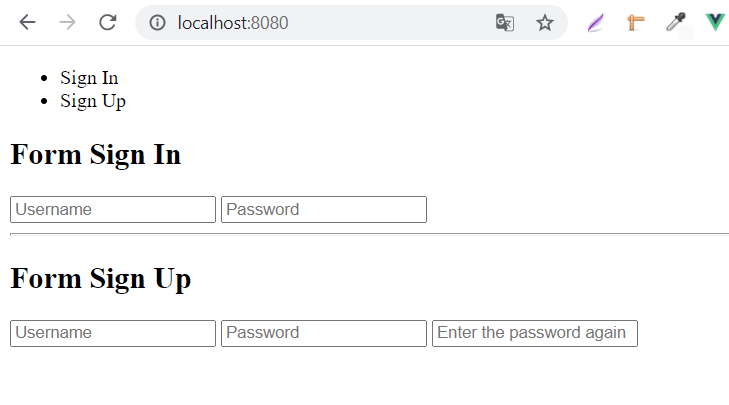
* Cú pháp truyền Array vào Style Binding dùng để áp dụng nhiều Object Style cho cùng 1 phần tử web.

1. [**Conditional Rendering**](https://vuejs.org/v2/guide/conditional.html)**: Điều kiện**

V-IF, V-ELSE, V-ELSE-IF, KEY: Ví dụ demo



Hình 96: Tạo form đăng nhập, đăng ký trên <template> chưa có v-if.



Hình 97:Màn hình localhost sau khi chạy chương trình khi chưa có v-if.



Hình 98: Dữ liệu khi đã có v-if.

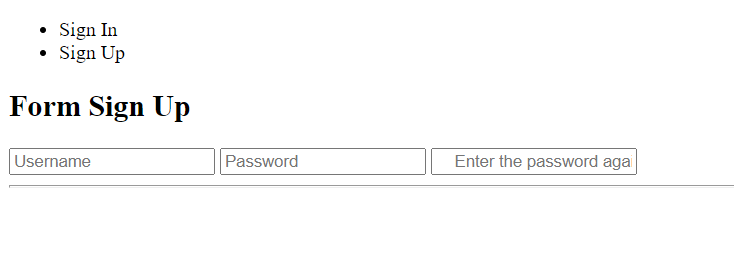
Mô tả hình ảnh: Sau khi đã tạo form đăng nhập/đăng ký và có ràng buộc bằng từ khó key  <input type="useruser" placeholder="Username" key="useruser-input"> bắt buộc khi người dùng đăng ký không bị trùng Username đã được đăng ký trước đó ta sẽ dùng v-if để ẩn/hiện những thông tin cần hiện.

v-on:click="changeTabs('login')" khi click vào Sign In thì sẽ hiện ra Form Sign In thông qua v-if="tabsSelected === 'login' ".



Hình 99:Màn hình localhost sau khi đã sử dụng v-if để ẩn Form Sign Up.

Và ngược lại khi ta click vào Sign Up v-on:click="changeTabs('register')" thì Form Sign In sẽ ẩn đi và hiện lên Form Sign Up v-else-if="tabsSelected === 'register'", nếu không dùng v-else-if thì ta cũng có thể sử dụng v-else.



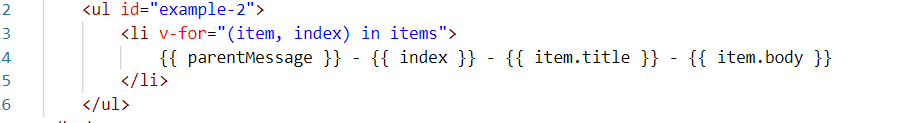
Hình 100:Màn hình localhost sau khi đã sử dụng v-if để ẩn Form Sign In…

1. **List Rendering: Vòng lặp**

V-FOR: Ví dụ demo:



Hình 101: Một dữ liệu đơn giản về các dữ liệu bị trùng nhau.

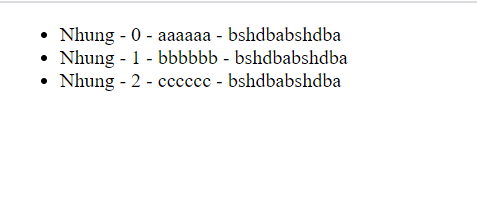


Hình 102:Dữ liệu sau khi đã thêm phần v-for.

Mô tả màn hình: Ta sử dụng v-for để tạo 1 vòng lặp để rút ngắn các dòng code tương tự nhau. index để lấy chỉ số vòng lặp.



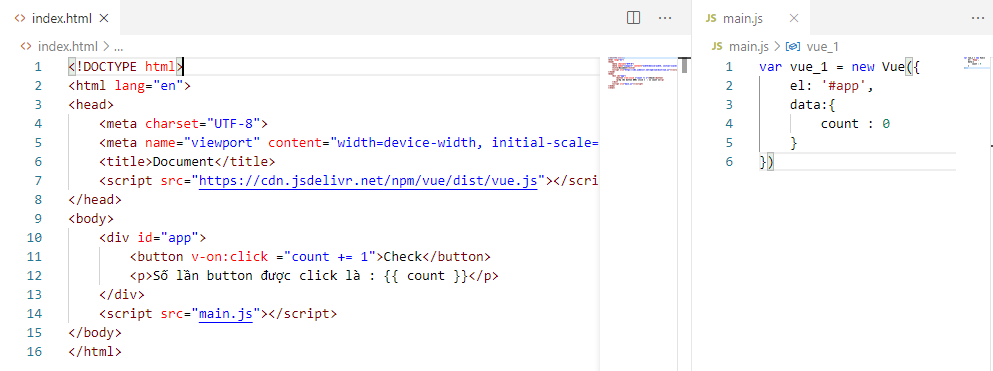
Hình 103:Trong main.js ta tạo một mảng danh sách tương tự.



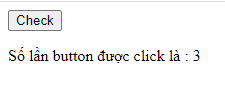
Hình 104: Sau khi chạy chương trình sẽ xuất hiện một danh sách có 3 hàng Nhung-chỉ số index-title-body.

1. **Event Handling**
   1. **Listening to Event**

Trong Vuejs chúng ta có directive “v-on” được dùng để lắng nghe các sự kiện DOM và thực hiện các lệnh JavaScript khi những Event này được kích hoạt. Ví dụ:



Hình 105: Code demo listenning to even.



Hình 106: Giao diện của demo dựa trên code ở hình phía trên.

* Giải thích ví dụ:

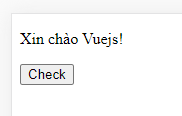
+ Ta có một biến “count” được khai báo bằng 0 trong main.js

+ Bên index.html ta tạo ra “button” và “thẻ p” trên button ta dùng v-on để bắt lại sự kiện click. Sau mỗi lần click thì ta thực hiện cộng count với 1 và show ra giá trị của conut ở thẻ p. đối với thao tác trực tiếp trên v-on ta chỉ có thể thao tác các phép toán đơn giản và toán tử 3 ngôi.

* 1. **Method Event Handlers**
* Như ở trên đã nói v-on chỉ có thể thao tác với các phép toán đơn giản hoặc là toán tử 3 ngôi. Đó cũng là lý do mà v-on cũng có thể nhận vào tên của một method mà ta muốn sử dụng. Ví dụ:



Hình 107: Code demo Method Event Handlers.



Hình 108: Giao diện của demo dựa trên code ở hình phía trên.

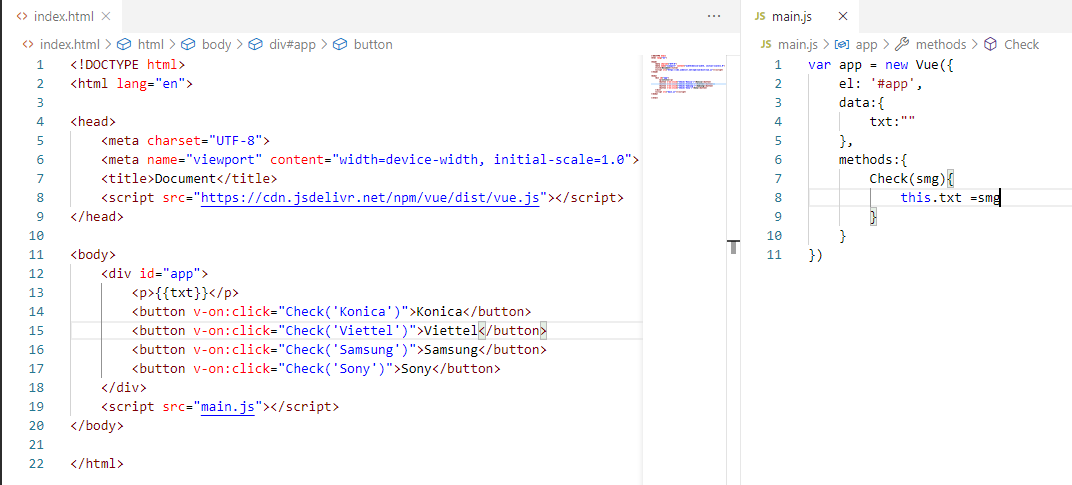
* Giải thích ví dụ:

+ ta có biến txt rỗng và biến name được gán chuỗi Vuejs và một method tên Check trong Check ta thực hiện gán dữ liệu cho txt bằng chuỗi xin chào cộng với biến name.

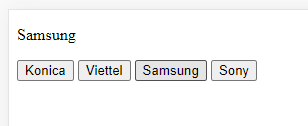
+ Bên html ta tạo ra thẻ p rỗng và button check. Ta tiếp tục dùng v-on để bắt sự kiện click của button. Khi click button thì sẽ thực thi method check và gán giá trị vào thẻ p.

* 1. **Methods in Inline Handlers**

Thay vì dùng bind trực tiếp ta còn có thể gọi method để sử dụng. Ví dụ:



Hình 109: Code demo Methods in inline Handlers.



Hình 110: Giao diện của code phía trên.

* Giải thích ví dụ:

Ở đây ta bắt lại sự kiện click của button và cho giá trị tuyền vào method check, trong method check ta lại gán giá trị truyền vào cho txt và show ra bằng thẻ p.

* 1. **Event Modifiers**
* Trong rất nhiều trường hợp, chúng ta cần gọi event.preventDefault() hoặc event.stopPropagation() bên trong một phương thức xử lí sự kiện. Tuy việc này không có gì khó, sẽ tốt hơn nếu các phương thức chỉ phải tập trung giải quyết logic dữ liệu thay vì cáng đáng các sự kiện DOM.
* Để giải quyết vấn đề này, Vue cung cấp các **event modifier** cho v-on. Event modifier là một hậu tố (postfix) cho directive, được biểu thị bằng một dấu chấm.

+ stop.

+ prevent.

+ capture.

+ self.

+ once.

* 1. **Key Modifiers**
* Thông thường chúng ta muốn lắng nghe các sự kiện của bàn phím thì thường phải kiểm tra mã phím. Ở đây Vue cũng hỗ trợ chúng ta điều này bằng v-on:keyup.mã key code.
* Việc nhớ tất cả các key code là điều mà không phải ai cũng làm được nên Vue có hỗ trợ chúng ta một số alias cho các phím thông dụng, cách gọi cũng như trên v-on:keyup.alias .
* Một số alias mà Vue có:

+ enter.

+ tab.

+ delete (được dùng cho phím delete lẫn phím back).

+ esc.

+ space.

+ up.

+ down.

+ left.

+ right.

* Vue cũng cho phép chúng ta tạo ra alias bằng object toàn cục: *Vue.config.keyCodes.h* = 72.
  1. **System Modifier Keys**
* Từ phiên bản 2.1.0 trở đi ta có thể sử dụng các Modifier sau để gọi các event listener khi các phím Modifier được nhấn.

+ ctrl.

+ alt.

+ shift.

+ meta (Phím window).

* Các phím Modifier có một cách hoạt động khác với các phím thông thường. ví dụ ta sử dụng v-on:keyup.ctrl.enter thì sự kiện chỉ được kích hoạt khi bạn nhấn phím enter trong khi vẫn đang ấn giữ phím Ctrl. Sự kiện sẽ không được kích hoạt nếu bạn chỉ nhấn mỗi phím ctrl.
* Ta có thêm Modifier exact có thể sử dụng kết hợp với các Modifier khác để chỉ rõ sự kiện nên được thực thi.
* Từ phiên bản 2.2.0 trở đi ta có thêm 3 Modifier là:

+ left.

+ right.

+ middle.

Những Modifier này giới hạn xử lí cho những sự kiện được kích hoạt bằng chuột(trái, phải, giữa).

* 1. **Why Listeners in HTML?**
* Việc sử dụng Event listener trong HTML không vi phạm quy tắc “separation of concerns”, vì tất cả các hàm và biểu thwucs xử lý của Vue đề được ràng buộc chặc chẽ với ViewModel và sẽ không gây ra khó khan trong việc bảo trì. Một số lợi ích khi sử dụng v-on:

+ Định vị được hàm xử lí trong code JavaScript dễ dang thông qua đọc HTML.

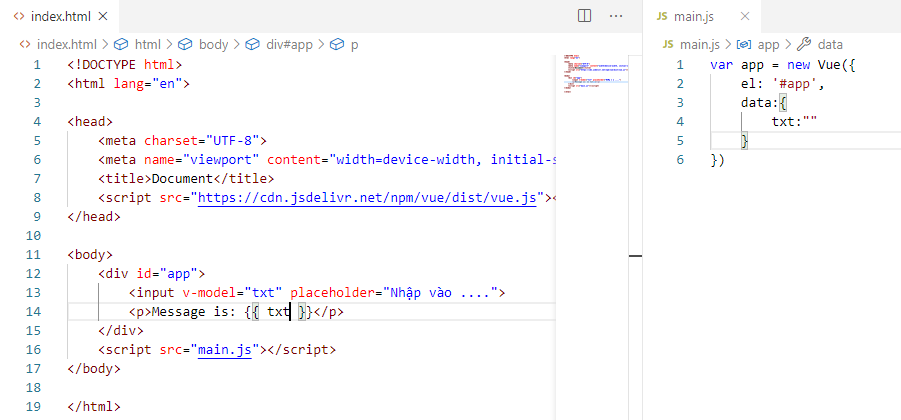
+ Vì không phải attach hàm sử lí sự kiện trong JavaScript thủ công nên trong ViewModel trở nên thuần logic và không phụ thuộc vào DOM. Điều này giúp chúng ta dễ dàng viết test.

+ Khi một ViewModel bị hủy thì tất cả các hàm sử lí kèm theo cũng sẽ được gỡ bỏ mà chúng ta không cần phải tìm và gỡ bỏ.

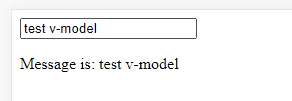
1. [**Form Input Bindings**](https://vuejs.org/v2/guide/forms.html)
   1. **Basic Usage**

* Bạn có thể sử dụng directive v-model để tạo ràng buộc dữ liệu 2 chiều lên các phần tử form input và textarea. Vue sẽ tự động chọn cách phù hợp để cập nhật phần tử này dựa trên kiểu của input. v-model là syntax sugar trong việc cập nhật dữ liệu dựa trên các sự kiện input từ người dùng kèm theo một số trường hợp đặc biệt khác.
* Text:

Chuỗi đơn giản trong type input.



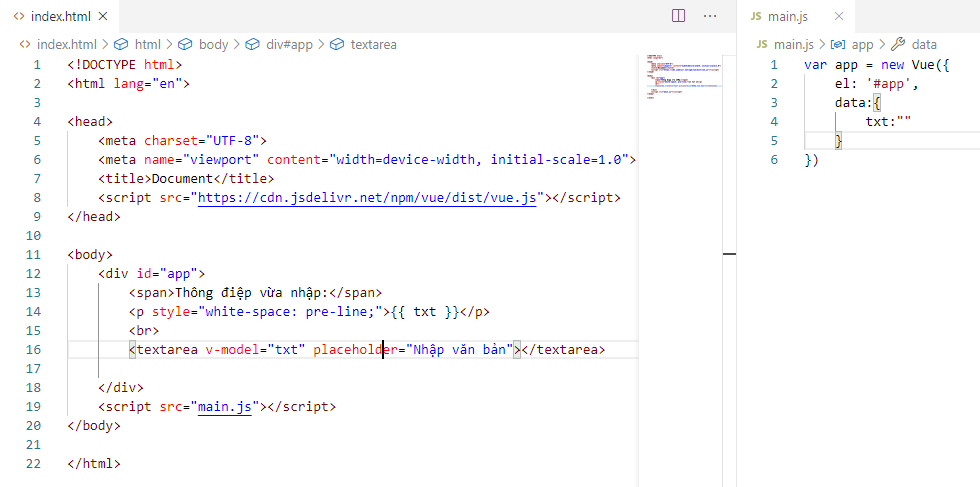
Hình 111: Code demo v-model đơn giản



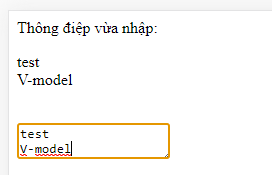
Hình 112: Giao diện của code demo phía trên

* Multiline text:

Chuỗi có nhiều dòng trong type textarea.



Hình 113 Code demo v-model nhiều dòng.



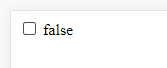
Hình 114: Giao diện của code demo phía trên

* Checkbox

Checkbox đơn:

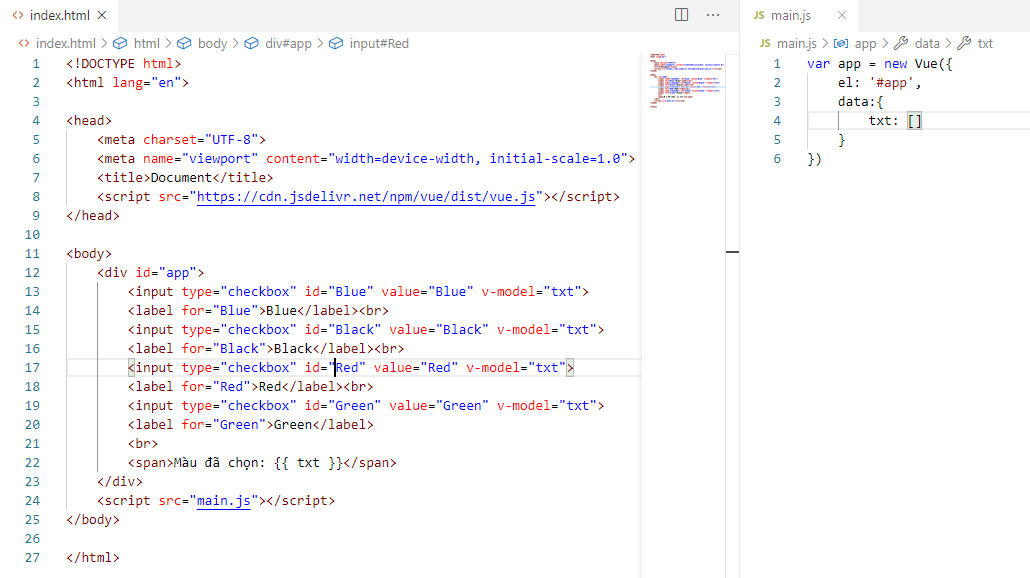


Hình 115: Code demo v-model checkbox đơn.

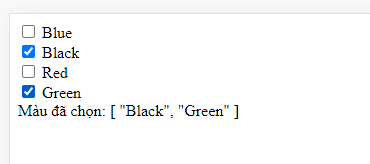


Hình 116: Giao diện của code demo phía trên.

Nhiều checkbox:

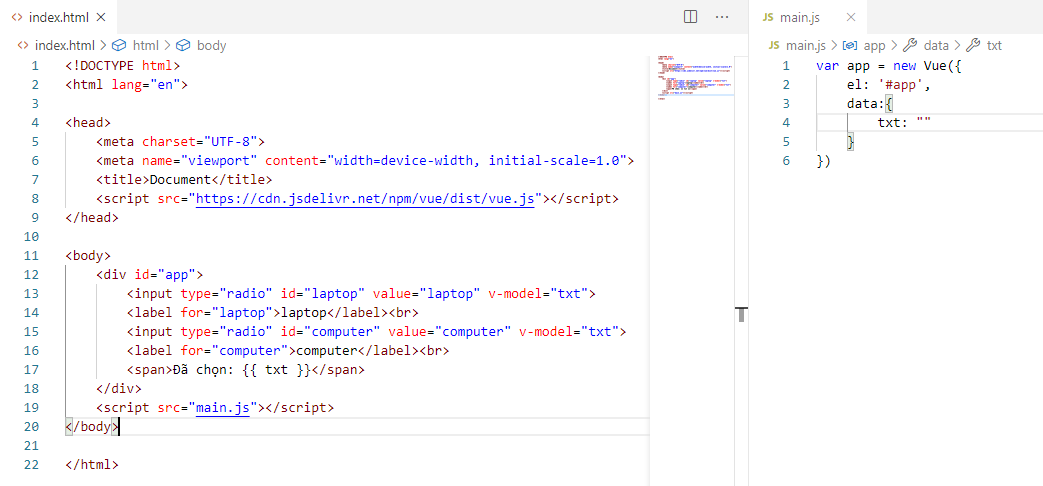


Hình 117: Code demo v-model nhiều checkbox.



Hình 118: Giao diện của code demo phía trên.

* Radio



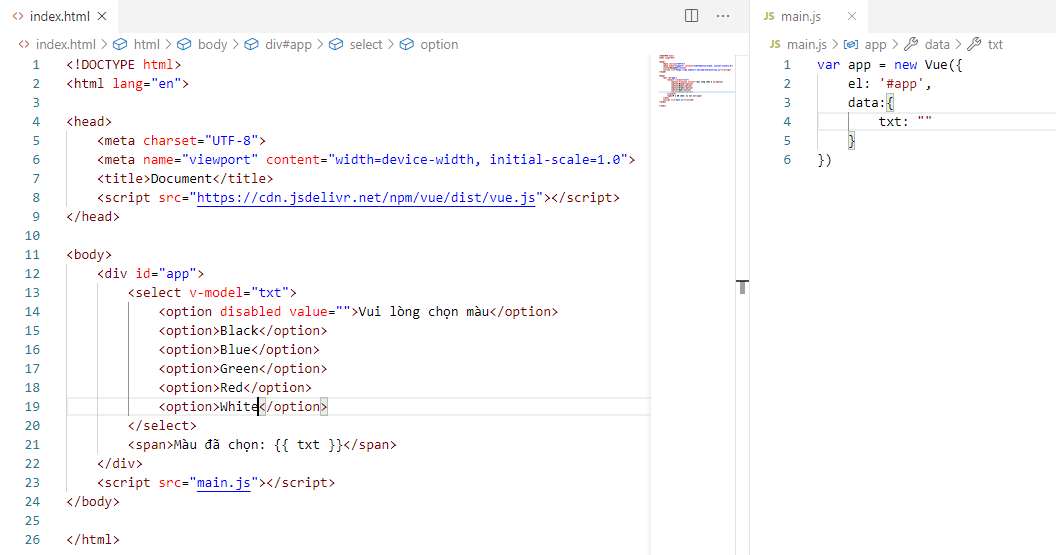
Hình 119: Code demo v-model Radio.



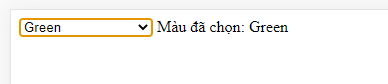
Hình 120: Giao diện của code demo phía trên.

* Select

Select đơn:

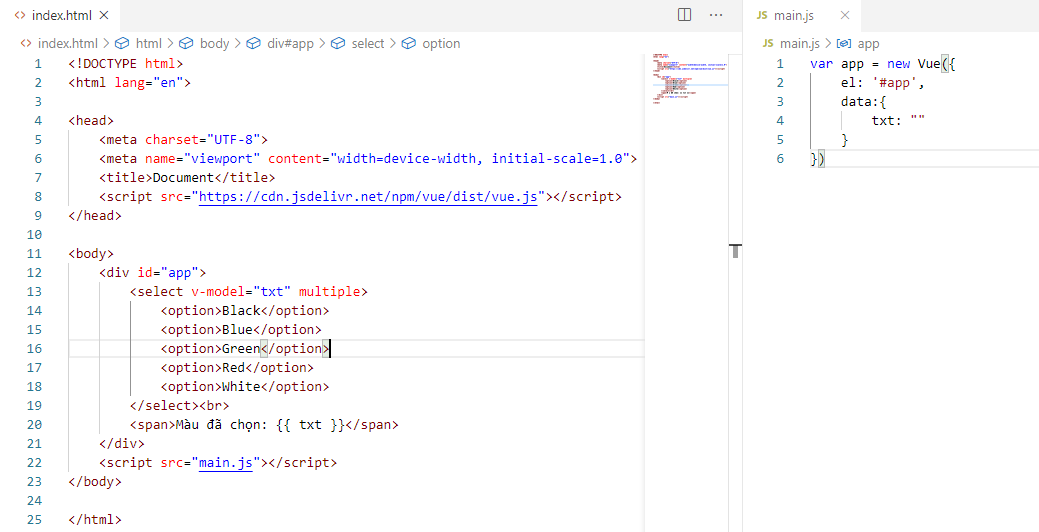


Hình 121: Code demo v-model Select.

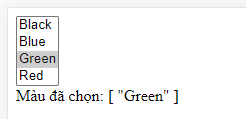


Hình 122: Giao diện của code demo phía trên.

Select nhiều lựa chọn:



Hình 123: Code demo v-model Select nhìu lựa chọn.

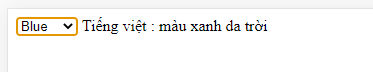


Hình 124: Giao diện của code demo phía trên.

Render <option> với v-for:

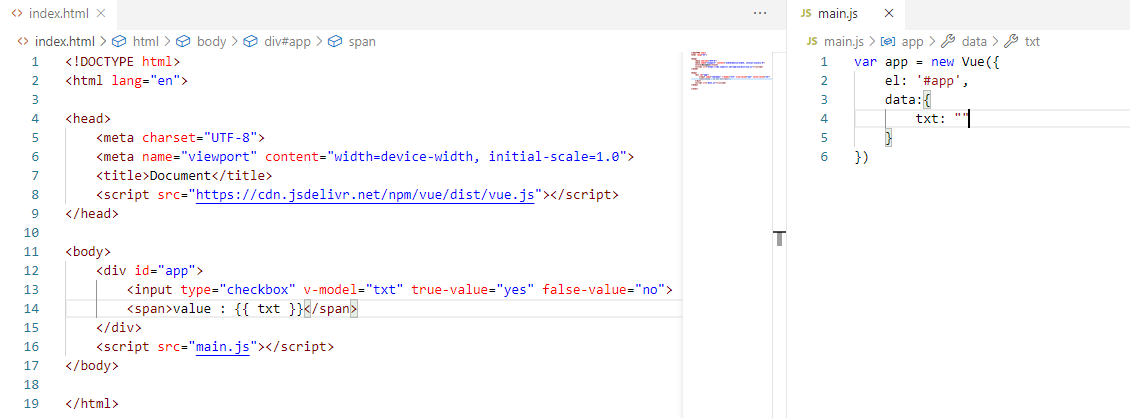


Hình 125: Code demo v-model Select render.

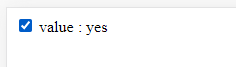


Hình 126: Giao diện của code demo phía trên.

* 1. **Value Bindings**
* Đối với radio, checkbox và select option, v-model thường ràng buộc giá trị kiểu chuỗi tĩnh. Nhưng đôi khi chúng ta muốn ràng buộc giá trị với một thuộc tính động trong đối tượng Vue. Chúng ta có thể sử dụng v-bind để làm điều này. Thêm nữa, sử dụng v-bind còn cho phép ràng buộc giá trị của input với các trị không phải kiểu chuỗi.
* Checkbox



Hình 127: Code demo Value Bindings checkbox.



Hình 128: Giao diện của code demo phía trên.

* Radio



Hình 129: Code demo Value Bindings radio.

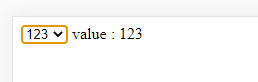


Hình 130: Giao diện của code demo phía trên.

* Select Options

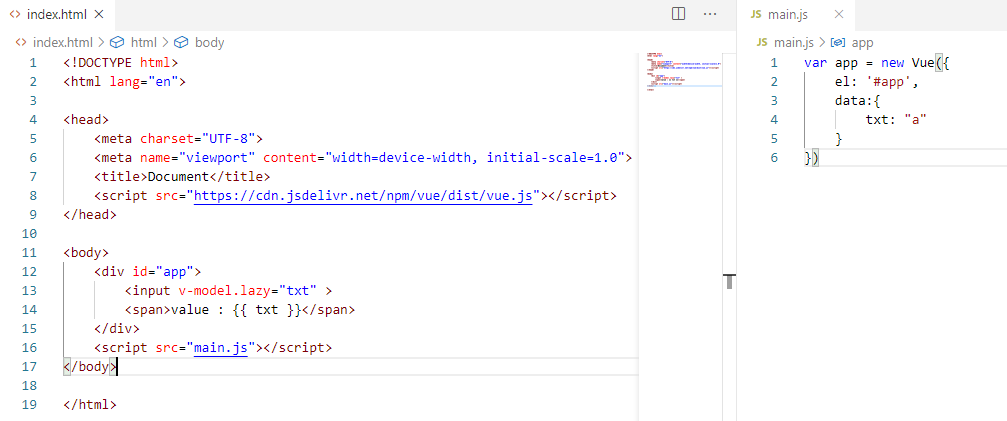


Hình 131: Code demo Value Bindings Select.



Hình 132: Giao diện của code demo phía trên.

* 1. **Modifiers**
* Lazy:

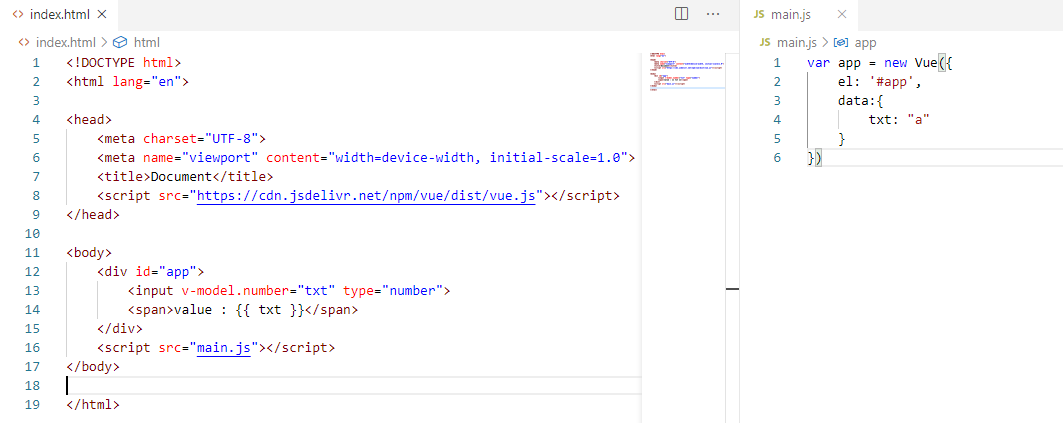


Hình 133: Code demo Modifiers lazy.

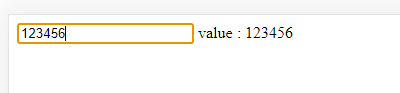


Hình 134: Giao diện của code demo phía trên.

* Number:



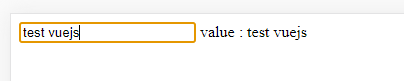
Hình 135: Code demo Modifiers Number



Hình 136: Giao diện của code demo phía trên.



Hình 137: Code demo Modifiers Trim.

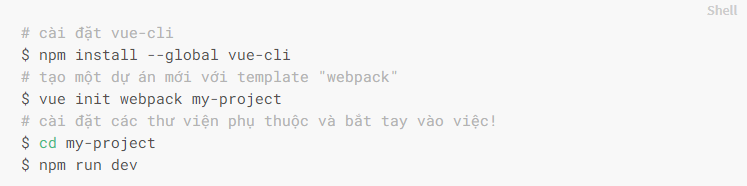


Hình 138: Giao diện của code demo phía trên.

* 1. **V-model with Components**
* Không phải lúc nào các kiểu input có sẵn của HTML cũng có thể đáp ứng nhu cầu của bạn. May mắn là các component của Vue cho phép bạn xây dựng các input riêng tái sử dụng được với các hành vi (behavior) được tùy chỉnh hoàn toàn. Các input này thậm chí còn hoạt động với v-model.

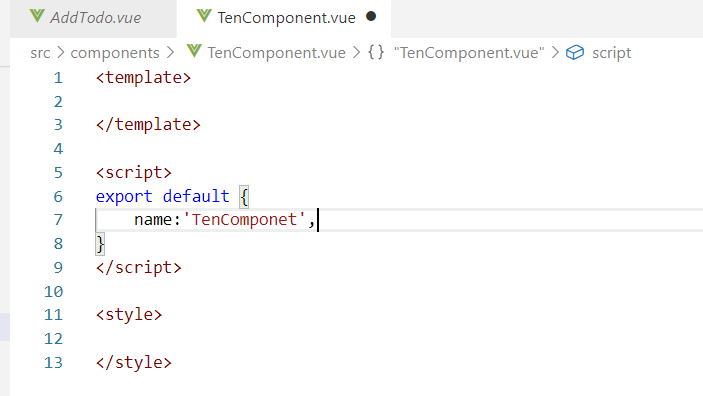
1. **Component**

* Download vue-cli về:



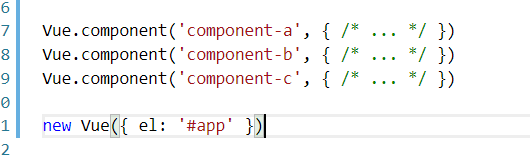
Hình 139: Cách tải vue-cli.

* Mô tả: Cài đặt vue-cli bằng npm.
  1. **Cách tạo một component trong vue-cli**
* B1: Tạo folder component trong folder sau đó tạo một file TenComponent.vue.
* B2: Cài đặt Extension Vetur.
* B3: Vào trong file TenComponent.vue gõ <vue> và nhấn enter sẽ xuất hiện cấu trúc của một component.
* B4: thiết lập tên của component.



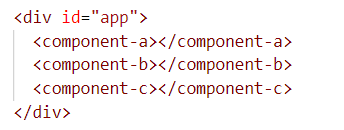
Hình 140: Tạo component.

* Mô tả: cấu trúc của một component trong vue-cli: <template>: là giao diện html,<script> là nơi viết javascrip <style> là nơi viết css/scss cho component.
  1. **Đăng ký component**
* Gồm có đăng ký toàn cục và đăng ký cục bộ.
* Đăng ký toàn cục:



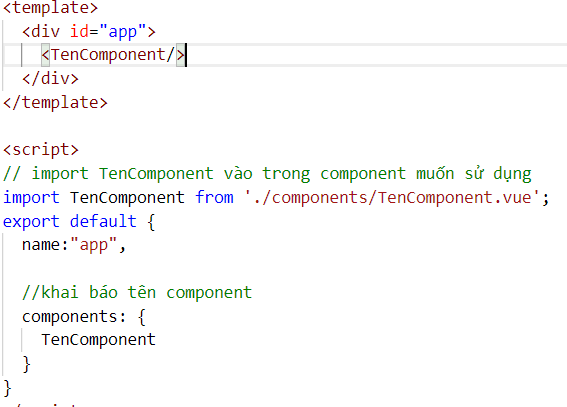
Hình 141: Đăng ký component toàn cục.

* Mô tả: Đăng ký 3 component khởi tạo toàn cục.



Hình 142: Sử dụng component toàn cục.

* Mô tả: 3 component có thể sử dụng lẫn nhau.
* Lưu ý: componet đăng ký cục bộ có thể dẫn đến việc dư thừa các component không dùng tới dẫn đến việc tăng khối lượng javascript mà người dùng tải xuống.
* Để sử dụng được component cục bộ ta phải đăng ký component:
  + B1: Vào component App.vue import TenComponet.vue.
  + B2: Ta khai báo tên component trong components.



Hình 143: Đăng ký component cục bộ.

* Mô tả: Cách đăng ký sử dụng component.

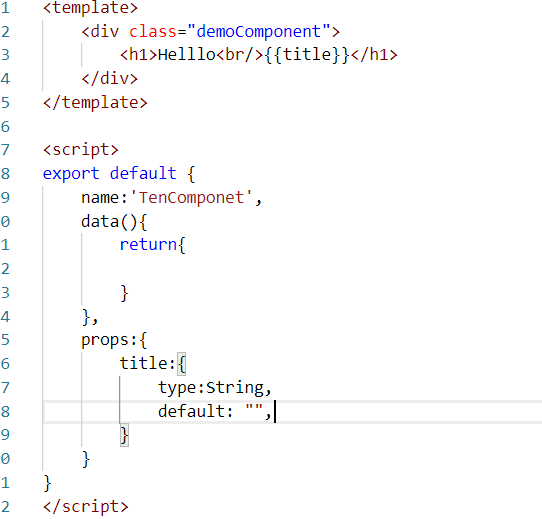
## **10.3.** **Props trong component**

* Component cha sẽ truyền dữ liệu qua cho component con thông qua v-bind:prop-name.
* Khi component con muốn sử dụng dữ liệu của component cha thì component con phải nhận dữ liệu thông qua Props, ta phải khai báo kiểu dữ liệu nhận vào từ component cha(array,object,string,number,…).



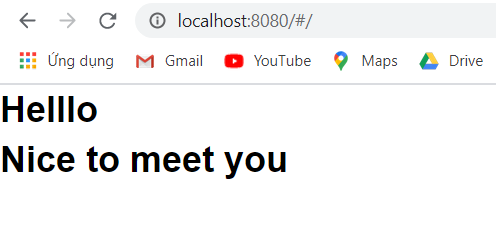
Hình 144: Gửi dữ liệu cho component con.

* Mô tả: tại component app(cha) ta gửi dữ liệu title cho component TenComponent(Con) thông qua giá trị title.



Hình 145: Nhận dữ liệu Props.

* Mô tả: Component con sẽ nhận lại thông qua Props:



Hình 146: Show dữ liệu.

* Mô tả: kết quả hiển thị khi dùng Component Props.