Cu Van Blue

Cuvankhoa.dev@gmail.com

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Vuejs documentation

Tổng hợp kiến thức tham khảo về Vuejs

Mục lục

[I. Mở đầu 1](#_Toc144996832)

[1. Giới thiệu 1](#_Toc144996833)

[2. Bắt đầu 1](#_Toc144996834)

[II. Cơ bản 1](#_Toc144996835)

[1. Create app 1](#_Toc144996836)

[2. Template syntax 2](#_Toc144996837)

[3. Reactivitiy fundamentals 4](#_Toc144996838)

[4. Computed props 4](#_Toc144996839)

[5. Class & style bindings 4](#_Toc144996840)

[6. Conditional redering 4](#_Toc144996841)

[7. List rendering 4](#_Toc144996842)

[8. Event 4](#_Toc144996843)

[9. Form input binding 4](#_Toc144996844)

[10. Life cycle hooks 4](#_Toc144996845)

[11. Watcher 4](#_Toc144996846)

[12. Template ref 4](#_Toc144996847)

[13. Component basics 4](#_Toc144996848)

[III. Component nâng cao 4](#_Toc144996849)

[IV. Tái sử dụng ( reuse) 4](#_Toc144996850)

[V. Component có sẵn ( built-in) 4](#_Toc144996851)

[VI. Mở rộng ( Scale up) 4](#_Toc144996852)

[VII. API 4](#_Toc144996853)

1. Mở đầu
2. Giới thiệu

* Vuejs là framework Js xây dựng giao diện người dùng
* Sử dụng html, css, js tiêu chuẩn
* Dựa trên mô hình code theo thành phần ( component-based) và khai báo
* 2 tính chất cốt lõi:
* Render giao diện dựa trên dữ liệu truyền vào
* React Render: tự động cập nhật giao diện khi dữ liệu thay đổi
* Điểm mạnh:
* File component đơn
* API style:
* Option api: khai báo script theo một khung có sẵn gồm: data ( dữ liệu dùng trong component), method ( các hàm xử lý), lifecycle hook
* Composition api: khai báo state với ref, các hàm xử lý, các lifecycle hook tự do không cần theo thứ tự

1. Bắt đầu

* Tạo dự án: > npm create vue@latest
* Cài đặt các gói cần thiết: >npm install
* Host local: >npm run dev
* Build product: >npm run build

1. Cơ bản
2. Create app

* Tạo một app với createApp()
* Truyền vào một component với vai trò là root (cha lớn nhất)
* Mount app vừa tạo vào div có id là ‘app’ trong file index.html => nội dung của component App sẽ hiển thị trong div có id là app
* Mount app luôn được gọi sau cùng

|  |
| --- |
| import { createApp } from ‘vue’  import App form ‘./App.vue’  const app = createApp({  /\* root component opts \*/  App  });  app.mount(‘#app’); |

* Cấu hình ứng dụng với app.config để xử lý tất cả lỗi của con cháu:

|  |
| --- |
| app.config.errorHandler = (err) => {  /\* handle error \*/  } |

* Đăng ký component có phạm vi mọi nơi trong ứng dụng

|  |
| --- |
| app.component('TodoDeleteButton', TodoDeleteButton) |

* Cho phép đăng ký nhiều phiên bản ứng dụng:

|  |
| --- |
| const app1 = createApp({  /\* ... \*/  })  app1.mount('#container-1')  const app2 = createApp({  /\* ... \*/  })  app2.mount('#container-2') |

1. Template syntax

Vue sử dụng cú pháp dựa trên html sau đó biên dịch thành js kết hợp với hệ thống render động ( react) có thể tìm ra số lượng thành phần tối thiểu, thao tác DOM tối thiểu để render lại

Có thể viết các hàm render như react nhưng không được hưởng tối ưu thời gian biên dịch

* 1. Nội suy văn bản ( chèn dữ liệu)
* Chèn dữ liệu vào html bằng dấu mustaches (ngoặc nhọn kép): {{ x }}
  1. Html thô
* Dấu mustaches chỉ chèn được plain text, nếu muốn chèn html phải sử dụng: v-html, nội dung thẻ span bên dưới sẽ bị thay bởi rawHtml:

|  |
| --- |
| <p>Using text interpolation: {{ rawHtml }}</p>  <p>Using v-html directive: <span v-html="rawHtml"></span></p> |

* 1. Thuộc tính động ( Attribute bindings)
* Dấu mustache cũng không thể dùng trong attribute html, thay vào đó dùng v-bind:tenthuoctinh hoặc viết tắt :tenthuoctinh :
* Nếu giá trị bind vào là null hoặc undefinded, thuộc tính sẽ bị bỏ qua.
* Với các thuộc tính chỉ chấp nhận giá trị true false, nếu truyền vào chuỗi rỗng, thuộc tính nhận giá trị chuỗi rỗng, nếu truyền các giá trị khác rỗng hoặc true, false, thuộc tính sẽ bị bỏ qua

|  |
| --- |
| <div v-bind:id="dynamicId"></div>  /\* short hand \*/  <div :id="dynamicId"></div> |

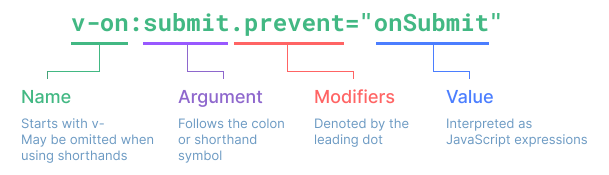
* Bind nhiều thuộc tính :

|  |
| --- |
| objectOfAttrs: {  id: 'container',  class: 'wrapper'  } |
| <div v-bind="objectOfAttrs"></div> |

* Các giá trị được biểu thị bởi phép nội suy hay binding sẽ được đồng bộ với biến truyền vào và được cập nhật mỗi khi biến đó thay đổi
  1. Biểu thức Javascript
* Trong Vue template, javascript có thể được viết tại:
* Bên trong dấu mustaches : {{ }}
* Bên trong thuộc tính v-bind: :id= “currentId++”
* Mỗi binding hoặc mustache chỉ chứa một biểu thức (toán tử 3 ngôi được tính là 1 biểu thức)

|  |
| --- |
| *<!-- this is a statement, not an expression: -->*  {{ var a = 1 }}  *<!-- flow control won't work either, use ternary expressions -->*  {{ if (ok) { return message } }} |

* Biểu thức trong template không thể truy cập các thuộc tính global như window, tuy nhiên có thể khai báo nó trong app.config.globalProperties
* Một số thư viện có thể dùng khi viết js trong template, nếu muốn dùng thêm thì phải khai báo: https://github.com/vuejs/core/blob/main/packages/shared/src/globalsAllowList.ts#L3
  1. Chỉ thị điều phối ( Directives)



Định nghĩa:

* Là một thuộc tính đặc biệt với tiền tố v- (v-bind, v-html,…)
* Nhận giá trị là một biểu thức javascript đơn ( single js expression) ngoại lệ: v-for, v-on, v-slot
* Nhiệm vụ: cập nhật cây DOM khi giá trị biểu thức nó nắm giữ thay đổi
* Một số chỉ thị có thể nhận đối số: v-bin:href hay :href dùng để update thuộc tính của thẻ html. Ví dụ với v-on dùng để lắng nghe sự kiện
* Truyền động thuộc tính : tên thuộc tính để trong ngoặc vuông, chấp nhận string. Nếu thuộc tính truyền vào bằng null || không hợp lệ => chỉ thị bị bỏ qua.

|  |
| --- |
| <a :[someAttr]="value"> ... </a> |

Modifiers:

* Là các hậu tố đặc biệt đằng sau dấu chấm, cho biết chỉ thị phải hoạt động theo một cách nào đó. Ở dưới đây, event.preventDefault() được gọi khi sự kiện submit xảy ra

|  |
| --- |
| <form @submit.prevent="onSubmit">...</form> |

1. Reactivitiy fundamentals
2. Computed props
3. Class & style bindings
4. Conditional redering
5. List rendering
6. Event
7. Form input binding
8. Life cycle hooks
9. Watcher
10. Template ref
11. Component basics
12. Component nâng cao
13. Tái sử dụng ( reuse)
14. Component có sẵn ( built-in)
15. Mở rộng ( Scale up)
16. Single-File Components
17. Tooling
18. Routing
19. State Management
    1. Vấn đề
    2. Quản lý state với Reactivity API
    3. SSR Considerations
    4. Pinia
20. Testing
21. Server-Side Rendering (SSR)
22. API
23. Global API
24. Composition API
25. Options API
26. Built-ins
    1. Directives

|  |  |  |  |
| --- | --- | --- | --- |
| Chỉ thị | Đầu vào | Chi tiết | Ví dụ |
| v-text | String | * Thay thế, ghi đè nội dung trong thẻ html bằng text truyền vào. * Nếu cần update 1 phần nội dung thì dùng mustache | <span v-text="msg"></span> |
| v-html | String | * Nội dung của v-html chèn vào như html thuần * Vue template syntax không thực thi | <div v-html="html"></div> |
| v-show | Any | * Setting display inline css * Trigger transition khi thay đổi * Ứng dụng làm render có điều kiện | <h1 v-show="ok">Hello!</h1> |
| v-if | Any | * Khi điều kiện trong v-if thay đổi, thành phần và chỉ thị chứa nó sẽ bị phá hủy và xây dựng lại * Nếu điều kiện là sai thì không hiển thị * Trigger transition * V-if có độ ưu tiền > v-for * Không nên sử dụng v-if và v-for trên cùng 1 component | <h1 v-if="awesome">Vue is awesome!</h1> |
| v-else, v-else-if | Any | * Theo sau v-if * Có thể lặp lại nhiều lần | <button @click="awesome = !awesome">Toggle</button>  <h1 v-if="awesome">Vue is awesome!</h1>  <h1 v-else>Oh no 😢</h1> |
| v-for | Array | Object | number | string | Iterable | * Render lại thành phần chứa nó nhiều lần dựa trên data source * Thành phần render tiếp theo gắn vào sau thành phần trước đó * Để render theo thứ tự, cần thêm thuộc tính :key * V-for | <div v-for="(item, index) in items"></div>  <div v-for="(value, key) in object"></div>  <div v-for="(value, name, index) in object"></div>  <div v-for="item in items" :key="item.id">  {{ item.text }}  </div> |
| v-on ( viết tắt là @) | Function | Inline Statement | Object (without argument) | * Gắn event listener lên element * Trên thành phần DOM gốc, nó listen các sự kiện DOM gốc, trên thành phần tự định nghĩa, nó lắng nghe các sự kiện tự định nghĩa * Nhận tham số là event * Có các modifier là:  |  |  | | --- | --- | | .stop |  | | .prevent |  | | .capture |  | | .self |  | | .{keyAlias} |  | | .once |  | | .left |  | | .right |  | | .middle |  | | .passive |  | | <button v-on:[event]=" doThat('hello', $event)"></button> |
| v-bind ( viết tắt là : ) | Any || arr hoặc obj nếu thuộc tính là class hoặc style | * Liên kết động thuộc tính, prop với một biểu thức * Khi dùng với thuộc tính class hoặc style v-bind chấp nhận kiểu obj hoặc arr * Modifier:  |  |  | | --- | --- | | .camel | Chuyển sang camel format (ex: camelFormat) | | .prop | Bắt buộc binding trở thành thuộc tính | | .attr |  | | <button v-bind:[key]="value"></button>  <div :class="[classA, { classB: isB, classC: isC }]"></div>  <MyComponent :prop="someThing" /> |
| v-model |  |  |  |
| v-slot |  |  |  |
| v-pre |  |  |  |
| v-once |  |  |  |
| v-memo |  |  |  |
| v-cloak |  |  |  |

* 1. Component
  2. Special Element
  3. Special Attribute

1. Single-File Component
2. Advanced APIs