



GIỚI THIỆU VUE 3X



III NỘI DUNG CHÍNH

I CÚ PHÁP

II DIRECTIVES

1. Cú Pháp

1. Nội suy

- Để hiển thị dữ liệu của biến “msg” trong ứng dụng div, chúng ta sử dụng **phép nội suy** đặt biến “msg” giữa các khóa kép `{{}}` bên trong div.

```
<body>
  <div id="app">
    <p>{{msg}}</p>
    <script src="._/main.js"></script>
  </body>
```

```
const app=Vue.createApp({
  data(){
    return{
      msg:"Hello world"
    }
  }
}).mount('#app');
```

1. Cú Pháp

1. Nội suy

Thực hiện nội suy một lần không cập nhật về thay đổi dữ liệu bằng cách sử dụng [chỉ thị v-once](#) , nhưng hãy nhớ rằng điều này cũng sẽ ảnh hưởng đến bất kỳ liên kết nào khác trên cùng một nút:

```
<span v-once>This will never change: {{ msg }}</span>
```

2. HTML

Để xuất ra HTML thuần, cần sử dụng chỉ thị **v-html**

1. Cú Pháp

2. HTML

```
<div id="example1" class="demo">
  <p>Using mustaches: {{ rawHtml }}</p>
  <p>Using v-html directive: <span v-
html="rawHtml"></span></p>
</div>
```

File HTML

Using mustaches: This should be red.

Using v-html directive: **This should be red.**

Châu Trần Trúc Ly

File JS

```
const RenderHtmlApp = {
  data() {
    return {
      rawHtml: '<span style="color: red">This should be red.</span>'
    }
  }
}
```

```
Vue.createApp(RenderHtmlApp).mount('#example1')
```

Kết quả

1. Cú Pháp

3. Liên kết nhiều thuộc tính

thì ra nhiều thuộc tính thì

```
const objectOfAttrs = {  
  id: 'container',  
  class: 'wrapper',  
  style: 'background-color:green'  
}
```

Có thể liên kết với thành phần bằng V-bind

```
<div v-bind="objectOfAttrs"></div>
```

1. Cú Pháp

4. Sử dụng biểu thức JavaScript

Vuejs hỗ trợ các biểu thức Javascript bên trong tất cả các ràng buộc dữ liệu:

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

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

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

```
<div v-bind:id="'list-' + id"></div>
```

Lưu ý: các biểu thức sau là không hợp lệ

```
<!-- this is a statement, not an expression: -->
{{ var a = 1 }}
```

```
<!-- flow control won't work either, use ternary expressions -->
{{ if (ok) { return message } }}
```

2. Directives

- **Directives (chỉ thị)** là một phần ý nghĩa của **Vue.js** và hành vi đặc biệt cho các phần tử html thuần túy trên trang
- **Directives** là các thuộc tính đặc biệt có tiền tố **v-**. Các giá trị thuộc tính **directives** được mong đợi là một biểu thức JavaScript duy nhất (ngoại trừ **v-for** và **v-on**). Công việc của chỉ thị là áp dụng một cách phản ứng các tác dụng cho DOM khi giá trị của biểu thức của nó thay đổi.

2. Directives

- **v-model** directive để thực hiện điều hướng 2 chiều **two way binding** giữa thành phần **HTML** và thuộc tính data mà nó đề cập đến trong code **JavaScript** **Vue**

```
<div id="app">
  <input type="text" v-model="name" />
  <p>{{ name }}</p>
<script src="./main.js"></script>
```

File HTML

File JS

```
const app=Vue.createApp({
  data(){
    return{
      name: ""
    }
  }
}).mount( '#app' );
```

2. Directives

- **v-model.lazy** : sẽ đồng bộ giá trị input với dữ liệu sau sự kiện change của input.
- **v-model.number**: cast giá trị từ input nhập vào từ kiểu string thành kiểu số.
- **v-model.trim**: tự động loại bỏ khoảng trắng trước và sau giá trị trong input.

2. Directives

- **v-bind**: được sử dụng để liên kết động một hoặc nhiều thuộc tính hoặc một thành phần prop với một biểu thức
- v-bind:<tên_thuộc_tính>
- Ví dụ:

```
<a v-bind:href="url"> ... </a>
```

```
<td class="btn btn-primary" colspan="3" v-bind:style="float:right">  
  <button @click="checkout">Checkout</button>  
  //checkout là 1 methods trong thuộc tính methods của Vue.  
</td>
```

Châu Trần Trúc Ly

9/7/2024

11

2. Directives

- Chúng ta có thể thay **v-bind<tên thuộc tính>** thành **:<tên thuộc tính>**
- Ví dụ:

```
<!-- full syntax -->  
<a v-bind:href="url"> ... </a>
```

```
<!-- shorthand -->  
<a :href="url"> ... </a>
```

```
<!-- shorthand with dynamic argument -->  
<a :[key]="url"> ... </a>
```

Châu Trần Trúc Ly

2. Directives

- **v-on:** lắng nghe các sự kiện DOM khác nhau (click, keyup, submit, etc..)

v-on: đây là directive sử dụng để kích hoạt các events trong Javascript với các phần tử HTML.

```
<a v-on:click="doSomething"> ... </a>
```

Chúng ta cũng có thể viết tắt **v-on** thành **@click**. Ngoài ra **v-on** còn có thêm các option khác để chúng ta dễ tùy chọn để thay đổi sự kiện gốc như sau:

2. Directives

- v-on.stop
- v-on.prevent
- v-on.capture
- v-on.self
- v-on.once: sự kiện click chỉ được kích hoạt tối đa 1 lần

2. Directives

HTML

```
<!--
  sự kiện click sẽ không được propagate (lan truyền)
  điều này tương đương với event.stopPropagation()
-->
<a v-on:click.stop="doThis"></a>

<!--
  sự kiện submit sẽ không reload trang
  điều này tương đương với event.preventDefault()
-->
<form v-on:submit.prevent="onSubmit"></form>

<!-- ta có thể nối modifier với nhau -->
<a v-on:click.stop.prevent="doThat"></a>

<!-- chỉ có modifier, không có phương thức xử lý -->
<form v-on:submit.prevent></form>

<!--
  dùng "capture mode" khi thêm event listener
  nghĩa là một sự kiện xảy ra với một phần tử bên trong sẽ được xử lý ở đây
  trước khi được xử lý bởi phần tử đó
  đọc thêm về event capturing: https://javascript.info/bubbling-and-capturing#cap
-->
<div v-on:click.capture="doThis">...</div>

<!--
  chỉ kích hoạt phương thức xử lý nếu event.target là chính phần tử được click,
  chứ không phải là một phần tử con
-->
<div v-on:click.self="doThat">...</div>
```

2. Directives

1. Lắng nghe sự kiện

```
<div id="basic-event">  
  <button @click="counter += 1">Add 1</button>  
  <p>The button above has been clicked {{ counter }} times.</p>  
</div>
```

```
Vue.createApp({  
  data() {  
    return {  
      counter: 0  
    }  
  }  
}).mount('#basic-event')
```

Add 1

The button above has been clicked 3 times.

2. Directives

2. Phương pháp xử lý sự kiện

```
<div id="event-with-method">
  <!-- `greet` is the name of a method defined below -->
  <button @click="greet">Greet</button>
</div>
```

```
Vue.createApp({
  data() {
    return {
      name: 'Vue.js'
    }
  },
  methods: {
    greet(event) {
      // `this` inside methods points to the current active instance
      alert('Hello ' + this.name + '!')
      // `event` is the native DOM event
      if (event) {
        alert(event.target.tagName)
      }
    }
  }
}).mount('#event-with-method')
```

Châu Trần Trúc Ly

Kết quả



This Pen is owned by [Vue](#) on [CodePen](#).

See more by [@Vue](#) on [CodePen](#)

2. Directives

Thay vì liên kết trực tiếp với tên phương thức, chúng ta có thể sử dụng các phương thức trong câu lệnh javascript

```
<div id="inline-handler">
  <button @click="say('hi')">Say hi</button>
  <button @click="say('what')">Say what</button>
</div>
```

Kết quả

Say hi Say what

```
Vue.createApp({
  methods: {
    say(message) {
      alert(message)
    }
  }
}).mount('#inline-handler')
```

Châu Tấn Trâm

9/7/2024

18

2. Directives

Để truy cập vào sự kiện DOM ta sử dụng biến \$event

```
<button @click="warn('Form cannot be submitted yet.', $event)">  
  Submit  
</button>
```

```
// ...  
methods: {  
  warn(message, event) {  
    // now we have access to the native event  
    if (event) {  
      event.preventDefault()  
    }  
    alert(message)  
  }  
}
```

2. Directives

3. Nhiều trình xử lý sự kiện: mỗi sự kiện phân cách nhau bằng dấu phẩy

```
<!-- both one() and two() will execute on button click -->
<button @click="one($event), two($event)">
  Submit
</button>
```

```
// ...
methods: {
  one(event) {
    // first handler logic...
  },
  two(event) {
    // second handler logic...
  }
}
```

2. 4 Event Modifiers

4. 1 click.stop

```
<template>
  <div class="event-handling">
    <div @click="alertHello">
      <button @click="alertWelcome">Click</button>
    </div>
  </div>
</template>

<script>
  export default {
    methods: {
      alertHello() {
        alert('Hello')
      },
      alertWelcome() {
        alert('Welcome')
      }
    }
  }
</script>

<style lang="scss" scoped>
</style>
```

2. 4 Event Modifiers

4. 1.click.stop: để ngăn chặn 2 sự kiện xảy ra đồng thời ở ví dụ trên ta dùng **.stop**, khi click vào sự kiện thì chỉ duy nhất sự kiện nào gọi phương thức **.stop** mới có hiệu lực

html

```
<button @click.stop="alertWelcome">Click</button>
```

2. 4 Event Modifiers

4. 2.click.prevent: bình thường khi khai báo form, mỗi lần nhấn submit thì trang sẽ bị load lại, để ngăn chặn việc load lại trang ta dùng **prevent**

```
html

<template>
  <div class="event-handling">
    <form @submit.prevent>
      <button type="submit">Submit</button>
    </form>
  </div>
</template>

<script>
  export default {
  }
</script>

<style lang="scss" scoped>
</style>
```

2. 4 Event Modifiers

4. 2.click.prevent:

Để thực hiện một hành động khi submit ta làm như sau:

html

```
<form @submit.prevent="sayWelcome">  
  <button type="submit">Submit</button>  
</form>
```


2.4 Event Modifiers

4. 2.click.prevent:

Hoặc thêm trực tiếp **prevent** vào button như sau

```
html

<template>
  <div class="event-handling">
    <form>
      <button type="submit" @click.prevent="sayWelcome">Submit</button>
    </form>
  </div>
</template>

<script>
  export default {
    methods: {
      sayWelcome() {
        alert(1)
      }
    }
  }
</script>

<style lang="scss" scoped>
</style>
```

2.4 Event Modifiers

Nếu muốn thực hiện stop và prevent đồng thời ta làm như sau

html

```
<button type="submit" @click.stop.prevent="sayWelcome">Submit</button>
```

4.3 click.once: Khi muốn chỉ thực hiện hành động một lần khi click

html

```
<button type="submit" @click.once="sayWelcome">Submit</button>
```

4.4 click.capture: ví dụ button trong thẻ div khi để bình thường không có stop thì hành động trong element button là con của div được gọi trước. Để thực hiện ngược lại, tức khi click vào button thì phương thức trong thẻ div được gọi trước sau đó đến con của nó là button

2.4 Event Modifiers

4.4 click.capture:

```
html

<template>
  <div class="event-handling">
    <div @click.capture="alertHello">
      <button @click="alertWelcome">Click</button>
    </div>
  </div>
</template>

<script>
  export default {
    methods: {
      alertHello() {
        alert('Hello')
      },
      alertWelcome() {
        alert('Welcome')
      }
    }
  }
</script>

<style lang="scss" scoped>
</style>
```

2.4 Event Modifiers

4.4 click.self: Một cách khác để ngăn chặn khi click vào button mà phương thức trong thẻ div cũng được gọi, thay vì sử dụng click.stop cho button đó là sử dụng click.self cho div, khi đó, khi click vào con của div thì hành động trong thẻ div sẽ không được gọi

```
html
<template>
  <div class="event-handling">
    <div @click.self="alertHello">
      <button @click="alertWelcome">Click</button>
    </div>
  </div>
</template>

<script>
  export default {
    methods: {
      alertHello() {
        alert('Hello')
      },
      alertWelcome() {
        alert('Welcome')
      }
    }
  }
</script>

<style lang="scss" scoped>
</style>
```

2. 4 Event Modifiers

4.4 click.passive

html

```
<!--
```

hành vi mặc định của sự kiện scroll (cuộn trang) sẽ xảy ra ngay lập tức, thay vì đợi `onScroll` hoàn tất.

```
-->
```

```
<div v-on:scroll.passive="onScroll">...</div>
```

2.4 Event Modifiers

Modifier là các hậu tố đặc biệt được biểu thị bằng dấu chấm, biểu thị rằng một chỉ thị phải được ràng buộc theo một cách đặc biệt nào đó. Ví dụ, modifier `.prevent` yêu cầu chỉ thị `v-on` gọi `event.preventDefault()` trên sự kiện được kích hoạt:

```
<form @submit.prevent="onSubmit">...</form>
```

v-on:submit.prevent="onSubmit"

Name

Starts with v-
May be omitted when
using shorthands

Argument

Follows the colon
or shorthand
symbol

Modifiers

Denoted by the
leading dot

Value

Interpreted as
JavaScript expressions

2.5 Key Events

Khi muốn lắng nghe sự kiện người dùng tương tác với nút trên bàn phím. Ta có thể sử dụng một số cách như sau:

```
html

<template>
  <div class="event-handling">
    <input @keyup.enter="alertHello">

    <input @keyup.13="alertHello">
  </div>
</template>

<script>
  export default {
    methods: {
      alertHello() {
        alert('Hello')
      }
    }
  }
</script>

<style lang="scss" scoped>
</style>
```

Châu Trần Trúc Ly

9/7/2024

31

2.5 Key Events

Vue cung cấp 1 số sự kiện mặc định cho một số nút như sau:

```
.enter  
.tab  
.delete (hiệu lực với cả 2 nút "Delete" và "Backspace")  
.esc  
.space  
.up  
.down  
.left  
.right
```

Vue hỗ trợ chúng ta một sự kiện ấn tổ hợp phím như sau:

- `.ctrl`
- `.alt`
- `.shift`
- `.meta`

Châu Trần Trúc Ly

2.5 Key Events

Vue cung cấp 1 số sự kiện mặc định cho một số nút như sau:

```
<!-- Alt + Enter -->  
<input @keyup.alt.enter="clear" />  
  
<!-- Ctrl + Click -->  
<div @click.ctrl="doSomething">Do something</div>
```

Lưu ý:

Ở ví dụ trên ta dùng @keyup.alt.enter thay cho tổ hợp phím Alt và Enter, thì clear sẽ được gọi khi ta thả phím Enter ra trước, nếu ta thả phím Alt ra trước sự kiện sẽ không được gọi.

2.5 Key Events

Exact

exact được sử dụng khi ta muốn thực hiện một hành động nào đó chỉ khi người dùng bấm chính xác các nút hay tổ hợp nút như ta thiết lập mà không có sự tác động đồng thời của các system modifiers như ta đã nói ở phần trên.

<!-- hành động bên dưới được gọi ngay cả khi các nút khác được bấm đồng thời như Alt hay Shift -->

<button @click.ctrl="onClick">A</button>

<!-- chỉ thực hiện khi bấm chính xác tổ hợp click+ctrl -->

<button @click.ctrl.exact="onCtrlClick">A</button>

<!-- Chỉ thực hiện khi click -->

<button @click.exact="onClick">A</button>

2. 6 Mouse Event

Vue cung cấp cho chúng ta sự kiện khi click chuột

- Left
- Right
- middle

```
<div id="app" style=
  "text-align: center;
  padding-top: 40px;">
  <button v-on:click.left=
    "data = !data">Show
  </button>

  <h1 v-if="data">GeeksforGeeks</h1>
</div>
</div>

<script>
  var app = new Vue({
    el: '#app',
    data: {
      data: false
    }
  })
</script>
```

3. Class và Style Bindings

- Vue cung cấp các cải tiến đặc biệt khi **v-bind** được sử dụng với **class** và **style**

1. Ràng buộc với Class HTML

- **:class**(viết tắt của **v-bind:class**)
- Ví dụ:

```
<div :class="{ active: isActive }"></div>
```

3. Class và Style Bindings

1. Ràng buộc với Class HTML

- Ràng buộc nhiều lớp

- Ví dụ 1:

HTML

```
<div
  class="static"
  :class="{ active: isActive, 'text-danger': hasError }"
></div>
```

Dữ liệu

```
data() {
  return {
    isActive: true,
    hasError: false
  }
}
```

Châu Trần Trúc Ly

Kết quả

```
<div class="static active"></div>
```

9/7/2024

37

3. Class và Style Bindings

1. Ràng buộc với Class HTML

- Ràng buộc nhiều lớp
- Ví dụ 2:

HTML

```
<div :class="classObject"></div>
```

Kết quả như nhau

Châu Trần Trúc Ly

Dữ liệu

```
data() {  
  return {  
    classObject: {  
      active: true,  
      'text-danger': false  
    }  
  }  
}
```

3. Class và Style Bindings

1. Ràng buộc với Class HTML

- Có thể liên kết với thuộc tính **computed** để trả về 1 object

- Ví dụ 2:

HTML

```
<div :class="classObject"></div>
```

Dữ liệu

```
data() {  
  return {  
    isActive: true,  
    error: null  
  }  
},  
computed: {  
  classObject() {  
    return {  
      active: this.isActive && !this.error,  
      'text-danger': this.error && this.error.type === 'fatal'  
    }  
  }  
}  
}
```

Kết quả như nhau

Châu Trần Trúc Ly

9/7/2024

39

3. Class và Style Bindings

1. Ràng buộc với Class HTML

- Nếu trong một thành phần chứa nhiều class thì chúng ta có thể dung mảng

- Ví dụ 2:

HTML

```
<div :class="[activeClass, errorClass]"></div>
```

Dữ liệu

```
data() {  
  return {  
    activeClass: 'active',  
    errorClass: 'text-danger'  
  }  
}
```

Kết quả

```
<div class="active text-danger"></div>
```

Châu Văn Trúc Ly

9/7/2024

40

3. Class và Style Bindings

1. Ràng buộc với Class HTML

- Ngoài ra chúng ta còn có thể viết dưới dạng biểu thức bậc 3

```
<div :class="[isActive ? activeClass : '', errorClass]"></div>
```

- Hoặc sử dụng cú pháp đối tượng trong mảng

```
<div :class="[ { active: isActive }, errorClass]"></div>
```

3. Class và Style Bindings

2. Ràng buộc Style

- Cú pháp: `:style`

```
<div :style="{ color: activeColor, fontSize: fontSize + 'px' }"></div>
```

```
data() {  
  return {  
    activeColor: 'red',  
    fontSize: 30  
  }  
}
```

3. Class và Style Bindings

2. Ràng buộc Style

- Thường nên ràng buộc `:style` trực tiếp với đối tượng để rõ ràng hơn.

```
<div :style="styleObject"></div>
```

```
data() {  
  return {  
    styleObject: {  
      color: 'red',  
      fontSize: '13px'  
    }  
  }  
}
```

Châu Trần Trúc Ly

3. Class và Style Bindings

2. Ràng buộc Style

- Cú pháp mảng:

```
<div :style="[baseStyles, overridingStyles]"></div>
```

- Hoặc mảng nhiều giá trị

```
<div :style="{ display: ['-webkit-box', '-ms-flexbox', 'flex'] }"></div>
```

4. Điều Kiện

- **v-if, v-else-if, v-else:** đây là những directive điều kiện, muốn cho các tag nào được hiện ra hoặc ko hiện ra cho người dùng xem
- **v-if** directive có thể được sử dụng để kiểm tra một số điều kiện trên dữ liệu đối tượng JavaScript
- **v-else** directive có thể sử dụng khi điều kiện trên v-if sai
- **v-else-if** directive có thể được sử dụng giống như bạn sử dụng mệnh đề else-if trong JavaScript

4. Điều Kiện

- Ví dụ:

```
const app=Vue.createApp({
  data() {
    return{
      product: 'Socks',
      image: 'https://www.vuemastery.com/images/challenges/vmSocks-green-onWhite.jpg',
      inStock: false,
      onSale: false
    }
  }
})
```

```
<div class="product-info">
  <h1>{{ product }}</h1>
  <p v-if="inStock">In Stock</p>
  <p v-else>Out of Stock</p>
  <span v-if="onSale">On Sale!</span>
</div>
```

Châu Trần Trúc Ly

4. Điều Kiện

- **v-show:** cách sử dụng giống v-if.
- Khác với **v-if** ở chỗ **v-show** render tất cả các phần tử html và sau đó các phần tử này hiển thị hay không thông qua thuộc tính CSS `display:none` .
- v-show không hỗ trợ thành phần **<template>**, cũng như không hoạt động với **v-else**

5. List Rendering

- Trong **v-for** còn hỗ trợ đối số thứ 2 để cho biết vị trí của **item** hiện tại.

```
<ul id="array-with-index">
  <li v-for="(item, index) in items">
    {{ parentMessage }} - {{ index }} - {{ item.message }}
  </li>
</ul>
```

```
Vue.createApp({
  data() {
    return {
      parentMessage: 'Parent',
      items: [{ message: 'Foo' }, { message: 'Bar' }]
    }
  }
}).mount('#array-with-index')
```

Châu Trần Trúc Ly

Kết quả

- Parent - 0 - Foo
- Parent - 1 - Bar

*Lưu ý: Cũng có thể
dùng **of** thay cho **in**
để gần với cú pháp
Javascript*

5. List Rendering

- v-for với 1 object

```
<ul id="v-for-object" class="demo">
  <li v-for="value in myObject">
    {{ value }}
  </li>
</ul>
```

```
Vue.createApp({
  data() {
    return {
      myObject: {
        title: 'How to do lists in Vue',
        author: 'Jane Doe',
        publishedAt: '2016-04-10'
      }
    }
  }
}).mount('#v-for-object')
```

Châu Trần Trúc Ly

Kết quả

- How to do lists in Vue
- Jane Doe
- 2020-03-22

- Cũng có thể cung cấp đối số thứ 2 gọi là khóa

```
<li v-for="(value, name) in myObject">
  {{ name }}: {{ value }}
</li>
```

Kết quả

- title: How to do lists in Vue
- author: Jane Doe
- publishedAt: 2020-03-22

9/7/2024

5. List Rendering

- Trong khi lặp các bạn nên bind thêm thuộc tính key.

```
<div v-for="item in items" :key="item.id">  
  <!-- content -->  
</div>
```

6. Hiển thị kết quả Filter/Sort

- Muốn hiển thị kết quả Filter hoặc sắp xếp của một mảng mà không thực sự thay đổi hoặc đặt lại dữ liệu ban đầu. Trong trường hợp này tạo thuộc tính **computed** để trả về kết quả Filter và sort

```
<li v-for="n in evenNumbers" :key="n">{{ n }}</li>
```

```
data() {  
  return {  
    numbers: [ 1, 2, 3, 4, 5 ]  
  }  
},  
computed: {  
  evenNumbers() {  
    return this.numbers.filter(number => number % 2 === 0)  
  }  
}
```

6. Hiển thị kết quả Filter/Sort

- Trong trường hợp mà Computed không khả thi thì sử dụng phương pháp sau

```
<ul v-for="numbers in sets">
  <li v-for="n in even(numbers)" :key="n">{{ n }}</li>
</ul>
```

```
data() {
  return {
    sets: [[ 1, 2, 3, 4, 5 ], [6, 7, 8, 9, 10]]
  },
  methods: {
    even(numbers) {
      return numbers.filter(number => number % 2 === 0)
    }
  }
}
```

6. Hiển thị kết quả Filter/Sort

1. Phạm vi v-for: nhận một số nguyên

```
<div id="range" class="demo">  
  <span v-for="n in 10" :key="n">{{ n }} </span>  
</div>
```

Kết quả

12345678910

Châu Trần Trúc Ly

6. Hiển thị kết quả Filter/Sort

2. V-for trên <template>: hiển thị 1 khối nhiều phần tử

```
<ul>
  <template v-for="item in items" :key="item.msg">
    <li>{{ item.msg }}</li>
    <li class="divider" role="presentation"></li>
  </template>
</ul>
```

6. Hiển thị kết quả Filter/Sort

3. V-for với v-if: Không khuyến khích sử dụng v-if và v-for cùng lúc

- Khi chúng tồn tại cùng lúc thì v-if có mức độ ưu tiên cao hơn v-for. Có nghĩa là v-if không có quyền truy cập vào các biến phạm vi của v-for

6. Hiển thị kết quả Filter/Sort

```
<li v-for="todo in todos" v-if="!todo.isComplete">
  {{ todo.name }}
</li>
```

- Lỗi vì todo không được xác định, để khắc phục lỗi trên chuyển v-for sang <template>

```
<template v-for="todo in todos" :key="todo.name">
  <li v-if="!todo.isComplete">
    {{ todo.name }}
  </li>
</template>
```


6. Hiển thị kết quả Filter/Sort

4. V-for với component

```
<my-component  
  v-for="(item, index) in items"  
  :item="item"  
  :index="index"  
  :key="item.id"  
></my-component>
```

6. Hiển thị kết quả Filter/Sort

Ví dụ:

Add a todo

Add

- Do the dishes
- Take out the trash
- Mow the lawn
- gggg

Ví dụ

```
<div id="todo-list-example">
  <form v-on:submit.prevent="addNewTodo">
    <label for="new-todo">Add a todo</label>
    <input
      v-model="newTodoText"
      id="new-todo"
      placeholder="E.g. Feed the cat"
    />
    <button>Add</button>
  </form>
  <ul>
    <todo-item
      v-for="(todo, index) in todos"
      :key="todo.id"
      :title="todo.title"
      @remove="todos.splice(index, 1)"
    ></todo-item>
  </ul>
</div>
```

Châu Trần Trúc Ly

```
const app = Vue.createApp({
  data() {
    return {
      newTodoText: '',
      todos: [
        {
          id: 1,
          title: 'Do the dishes'
        },
        {
          id: 2,
          title: 'Take out the trash'
        },
        {
          id: 3,
          title: 'Mow the lawn'
        }
      ],
      nextTodoId: 4
    }
  },
})
```

7. Form input binding

Để có thể bind dữ liệu cho các loại input form thì chúng ta sẽ sử dụng v-model. Đây là kiểu "2 way binding" tức là dữ liệu các bạn khai báo từ data sẽ được bind với các input và dữ liệu nhập từ input sẽ được bind trực tiếp với những gì các bạn khai báo trong data

7.1 Input text

```
<div id="v-model-basic" class="demo">
  <input v-model="message"
placeholder="edit me" />
  <p>Message is: {{ message }}</p>
</div>
```

Châu Trần Trúc Ly

```
Vue.createApp({
  data() {
    return {
      message: ''
    }
  }
}).mount('#v-model-basic')
```

Kết quả

9/7/2024

Message is:

7. Form input binding

7.2 Multiline text

```
<div id="v-model-textarea" class="demo">
  <span>Multiline message is:</span>
  <p style="white-space: pre-line;">{{
message }}</p>
  <br />
  <textarea v-model="message"
placeholder="add multiple lines">
</textarea>
</div>
```

```
Vue.createApp({
  data() {
    return {
      message: ''
    }
  }
}).mount('#v-model-textarea')
```

Kết quả

Multiline message is:

add multiple lines

9/7/2024

7. Form input binding

7.3 Checkbox

```
<div id="v-model-checkbox" class="demo">
  <input type="checkbox" id="checkbox" v-
model="checked" />
  <label for="checkbox">{{ checked }}
</label>
</div>
```

```
Vue.createApp({
  data() {
    return {
      checked: false
    }
  }
}).mount('#v-model-checkbox')
```

Kết quả

☐ false

7. Form input binding

7.3 Checkbox: ví dụ về multiple checkbox

```
<div id="v-model-multiple-checkboxes">
  <input type="checkbox" id="jack" value="Jack" v-model="checkedNames" />
  <label for="jack">Jack</label>
  <input type="checkbox" id="john" value="John" v-model="checkedNames" />
  <label for="john">John</label>
  <input type="checkbox" id="mike" value="Mike" v-model="checkedNames" />
  <label for="mike">Mike</label>
  <br />
  <span>Checked names: {{ checkedNames }}</span>
</div>
```

```
Vue.createApp({
  data() {
    return {
      checkedNames: []
    }
  }
}).mount('#v-model-multiple-checkboxes')
```

Châu Trần Trúc Ly

Kết quả

☐ Jack ☐ John ☐ Mike
Checked names: []

7. Form input binding

7.4 Radio

```
<div id="v-model-radiobutton">
  <input type="radio" id="one" value="One" v-model="picked" />
  <label for="one">One</label>
  <br />
  <input type="radio" id="two" value="Two" v-model="picked" />
  <label for="two">Two</label>
  <br />
  <span>Picked: {{ picked }}</span>
</div>
```

```
Vue.createApp({
  data() {
    return {
      picked: ''
    }
  }
}).mount('#v-model-radiobutton')
```

Châu Trần Trúc Ly

Kết quả

☐ One
☐ Two
Picked:

7. Form input binding

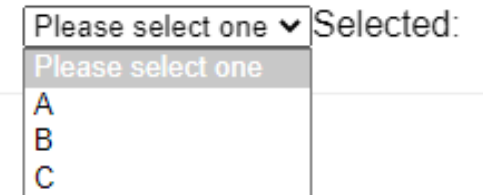
7.5 Select

```
<div id="v-model-select" class="demo">
  <select v-model="selected">
    <option disabled value="">Please select one</option>
    <option>A</option>
    <option>B</option>
    <option>C</option>
  </select>
  <span>Selected: {{ selected }}</span>
</div>
```

```
Vue.createApp({
  data() {
    return {
      selected: ''
    }
  }
}).mount('#v-model-select')
```

Châu Trần Trúc Ly

Kết quả



The screenshot shows the rendered output of the Vue.js code. It features a select dropdown menu with the text "Please select one" and a downward arrow. The dropdown is open, showing three options: "A", "B", and "C". To the right of the dropdown is a text label "Selected:".

7. Form input binding

7.5 Select với v-for

```
<div id="v-model-select-dynamic" class="demo">
  <select v-model="selected">
    <option v-for="option in options" :value="option.value">
      {{ option.text }}
    </option>
  </select>
  <span>Selected: {{ selected }}</span>
</div>
```

```
Vue.createApp({
  data() {
    return {
      selected: 'A',
      options: [
        { text: 'One', value: 'A' },
        { text: 'Two', value: 'B' },
        { text: 'Three', value: 'C' }
      ]
    }
  }
}).mount('#v-model-select-dynamic')
```

Châu Trần Trúc Ly

Kết quả

One ▼	Selected: A
One	
Two	
Three	

7. Form input binding

7.6 Ràng buộc về giá trị: đối với radio và checkbox các ràng buộc giá trị thường là chuỗi tĩnh

```
<!-- `picked` is a string "a" when checked -->  
<input type="radio" v-model="picked" value="a" />
```

```
<!-- `toggle` is either true or false -->  
<input type="checkbox" v-model="toggle" />
```

```
<!-- `selected` is a string "abc" when the first option is selected -->  
<select v-model="selected">  
  <option value="abc">ABC</option>  
</select>
```

7. Form input binding

- Nhưng muốn liên kết dữ liệu động thì chúng ta dùng ràng buộc v-bind

```
<input type="radio" v-model="pick" v-bind:value="a" />
```

```
// when checked:  
vm.pick === vm.a
```

```
<select v-model="selected">  
  <!-- inline object literal -->  
  <option :value="{ number: 123 }">123</option>  
</select>
```

```
// when selected:  
typeof vm.selected // => 'object'  
vm.selected.number // => 123
```

Châu Trần Trúc Ly

7. Form input binding

7.6.1 lazy

Mặc định thì Vue sẽ bind dữ liệu ngay khi chúng ta thực hiện một input event tức là ngay sau khi chúng ta nhập một kí tự. Ta có thể dùng lazy để Vue sẽ nhận ra và chỉ bind sau khi chúng ta kết thúc quá trình nhập, tức là mỗi lần nhập 1 kí tự không thấy xuất ra màn hình, thay vào đó chúng ta có thể bấm enter hoặc click ra ngoài ô input sẽ thấy có dữ liệu mới được xuất ra.

7. Form input binding

7.6.1 lazy

```
<!-- synced after "change" instead of "input" -->  
<input v-model.lazy="msg" />
```

7.6.2 number: bình thường nếu chúng ta bind như sau

```
<input type="number" v-model="message" name="">
```

- Ta mở Vue-devtool có thể thấy rằng giá trị của message là kiểu string, mặc dù input ta để type là number. Do đó để Vue cast giá trị ta nhập từ input sang number ta làm như sau:

```
Châu Trần Trúc Ly  
<input type="number" v-model.number="message" name="">
```

7. Form input binding

7.6.3 trim: nếu muốn tự động cắt bớt khoảng trắng do người dung nhập vào, ta dùng trim

```
<input v-model.trim="msg" />
```



The
end