Học Vuejs 2

sudo npm i -g vue-cli // dùng để instal vue cli vào máy tinh

* khởi taoj 1 dự án vuejs

**vue init webpack-simple vueCliTest**

**npm install**

**npm run dev**

- command dùng để build forder dist nếu bạn muốn :

npm run build

extension Visual Studio Code : Vetur

* command chạy khi lỗi limit watches: echo fs.inotify.max\_user\_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p

**Cấu trúc folder của Vue**

[**https://www.youtube.com/watch?v=jgHS3NhRVCc&list=PLU4OBh9yHE95G\_Y1cUVY-5Mc9P-rQBY3F&index=190**](https://www.youtube.com/watch?v=jgHS3NhRVCc&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=190)

<https://panjiachen.github.io/vue-element-admin-site/guide/#features>

<https://github.com/PanJiaChen/vue-element-admin>

| v-once | dùng để show value của biến trong này chỉ 1 lần thôi khi biến message có thay đổi thì cũng ko update lại ở trong này  <span v-once>This will never change: {{ message }}</span> |
| --- | --- |
| v-if="seen" | v-if sẽ thêm vào hoặc bỏ đi phần tử <p> dựa trên tính đúng sai của giá trị của biến seen. |
| Vuejs sẽ ko thể làm vc trực tiếp đc với các attribute của thẻ html như id, href, ... nên ta ko thể khai báo ms cú pháp href=”{{link}}”  mà vuejs hỗ trợ khai báo bằng **v-bind:tên thuộc tính** or sử dụng viết tắt **:tên thuộc tính**  <https://v2.vuejs.org/v2/guide/syntax.html#Raw-HTML>  VD:<a :id="idElement" v-bind:href="link">Binance</a> | |
| v-html="testHtml"  VD:<span v-html="testHtml"></span> | Dùng để in ra biến có value là 1 element. Nếu dùng cách bthg thì sẽ ko render ra đc element như mong muốn mà sẽ chỉ ra đc chuỗi string nên ta cần app dụng v-html |
| To address this problem, Vue provides **event modifiers** for v-on. Recall that modifiers are directive postfixes denoted by a dot.   * .stop * .prevent * .capture * .self * .once * .passive   <!-- the click event's propagation will be stopped --> <a v-on:click.stop="doThis"></a>  <!-- the submit event will no longer reload the page --> <form v-on:submit.prevent="onSubmit"></form>  <!-- modifiers can be chained --> <a v-on:click.stop.prevent="doThat"></a>  <!-- just the modifier --> <form v-on:submit.prevent></form>  <!-- use capture mode when adding the event listener --> <!-- i.e. an event targeting an inner element is handled here before being handled by that element --> <div v-on:click.capture="doThis">...</div>  <!-- only trigger handler if event.target is the element itself --> <!-- i.e. not from a child element --> <div v-on:click.self="doThat">...</div> | |
| v-on:click.once | chỉ cho phép click 1 lần các lần sau sẽ ko apply  VD:  <a v-on:click.once="increase()">test "v-on:click.once"</a> |
| v-on:keyup.delete | Event check khi b ấn vào nút “Delete” or “Backspace” keys |
| v-on:mousemove.stop | stop để dừng sự kiện này lại  <span v-html="testHtml" v-on:mousemove.stop=""></span> |
| v-on:keyup.enter.space | có thể khai báo chống các keyup lên ntn |
| Vue provides aliases for the most commonly used key codes when necessary for legacy browser support:   * .enter * .tab * .delete (captures both “Delete” and “Backspace” keys) * .esc * .space * .up * .down * .left * .right | |
| v-model | Sử dụng giống trong angular  sẽ tự động in và cập nhật biến mà gán vào  <input type="text" v-on:keyup.delete="showMessageDelete()" v-model="message"> |
| Computed dùng để định nghĩa các function chỉ dùng riêng rẽ ko muốn chung ms các function khác, và khi call function này thì sẽ ko cần dấu () đi kèm  // tat ca du lieu truyen qua html de o dang du lieu tho  // computer thuc hien nhu mot function nhung no chi dc coi la mot thuoc tinh  // computer va watch thuong di voi nhau  // watcher theo doi sy thay doi cua bien de lam mot hanh dong nao | |
| Watch  watch: {  diem: function(newValue) {  console.log(newValue)  console.log('gias tri da duoc thay doi')  }  }  - để theo dõi sự thay đổi của 1 biến nào đấy để thực hiện hành động | |
| :class="[color, totalCaculator, {red: quanlyRed}  ]" | Khai báo nhiều class gán cho element |
| <https://v2.vuejs.org/v2/guide/class-and-style.html>  khai báo style and class với vuejs | |
| v-if | V-if dùng để khai báo cho câu điều kiện nếu true thì ms show element r a  <div v-if="xacdinh">show khi true</div> |
| template | Vuejs cung cấp 1 box component là template để có thể đóng gói html cho dễ kiểm soát  - nó ko phải là 1 thẻ html, nhưng vẫn có thể apply các tính năng v-for, v-if  nhưng ko thể apply v-show vì v-show sẽ chỉ apply với các element mà template ko phải là 1 thẻ  <template v-for="(menu,index) in menus">  <h3>{{menu}}</h3>  <div>{{index}}</div>  </template> |
| v-for | dùng vòng lặp  <!-- lặp các số tự nhiên từ 1 -> 10 -->  <li v-for="n in 10">{{n}}</li>  <!-- lặp theo mảng -->  <li v-for="menu in menus">{{menu}}</li> |
| **Refs**  - Dùng để đánh dấu element và theo dõi trạng thái của element  VD:<span class="xxxx yjghj" data-title="title ok di em" @click="increase" ref="infoButton">Increase</a>  console.log(this.$refs.infoButton)  console.log(this.$refs.infoButton.classList)  this.$refs.infoButton.innerText = 'New text';  this.$refs.infoButton.classList.add('yyyyy')  - có thể đc call bên ngoài app.$refs.infoButton.innerText = 'ok di em' | |
| **API**  - Component,  - Component đc tại ra để 1 đối tg có thể quản lý đc nhiều đoạn giống nhau phỉa luôn đc call trong 1 instance nào đấy, nó ko đứng riêng lẻ đc  Since components are reusable Vue instances, they accept the same options as new Vue, such as data, computed, watch, methods, and lifecycle hooks. The only exceptions are a few root-specific options like el.  - khai báo:  Vue.component('hello', {  data: function () {  return {  count: 0  }  },  template: '<button v-on:click="count++">You clicked me {{ count }} times.</button>'  })  Call theo tên của component <hello></hello>  - Chia sẻ dữ liệu giữa các component,  - Vue.component('hello', {  data: function () {  return {  count: 0  }  },  template: '<button v-on:click="count++">You clicked me {{ count }} times.</button>'  })  Khi component đc khai báo data ntn thì component nào sẽ hoạt động riêng ở component đấy, khi click thì sẽ không ảnh hưởng đến thằng khác  Nhưng nếu data lại là 1 object đc khai báo ngoài thì khi click vào button thì các giá trị sẽ thay đổi đồng loạt  **- có 2 cachs để khai báo component: global (có thể dùng chung ) , local (chỉ đc dùng khi instance khai báo)**  - Khai báo Global  Vue.component('hello', {  data: function () {  return data  },  template: '<button v-on:click="increase">You clicked me {{ count }} times.</button>',  methods: {  increase:function() {  this.count += 1;  console.log(this)  }  }  })  - Khi khai báo ntn thì call component này ở instace nào cũng đc  - Khai báo Local: khai báo sẵn các thông số của component dưới dạng 1 biến xong khi nào cần thì call nó trong instance  var cpt = {  data: function () {  return {  count: 0,  };  },  template:  '<button v-on:click="increase">You clicked me {{ count }} times.</button>',  methods: {  increase: function () {  this.count += 1;  console.log(this);  },  },  };  /// call component local ở trong instant  var app = new Vue({  el: '#app',  data: {  message: 'Hello Vuexxxxx!',  count: 0,  },  components: {  hello: cpt,  },  })   * call compoent có giá trị là 1 biến thay đổi để sử dụng trong case có nhiều compoent show thay thế nhau cùng 1 vị trí   <!-- Component changes when currentTabComponent changes -->  <component v-bind:is="currentTabComponent"></component>  // khai bao component trong main.js và sẽ call nó trong component App.vue  import Vue from 'vue'  import Contact from './Contact.vue'  import ChuaDom from './App.vue'  Vue.component('contact-component', Contact)  new Vue({  el: '#xxx',  render: h => h(ChuaDom)  })  import conponent vào component  <template>  <div>  <h1>{{ msg }}</h1>  <contact-content></contact-content>  </div>  </template>  <script>  import ContentContact from './ContentContact.vue';  export default {  name: "app",  data() {  return {  msg: "from contact page",  };  },  components: {  'contact-content': ContentContact,  },  };  </script>  **Đặt tên cho Component**  components: {  'contact-content': ContentContact, //dặt tên thông thường  contactContent: ContentContact, // đặt tên viết liền thì ko cần để trong dấu ngoặc  ContentContact // đặt tên theo luôn tên file component mình import vào  },  **Phạm vi hoạt động của thẻ Style trong các component**  <style scoped>  div {  border: 1px solid red;  }  </style>   * ta dùng thuộc tính scoped để cố định style chỉ apply cho component đấy, nếu ko dùng thì nó sẽ bị đè style và sẽ ưu tiên style của children hơn parent * có bao nhiêu component thì khi render trong head sẽ có bấy nhiêu link css đc synce ra | |
| **Vuejs lifecycle**  https://www.youtube.com/watch?v=KpDn4yO5s6w&list=PLU4OBh9yHE95G\_Y1cUVY-5Mc9P-rQBY3F&index=55  var app = new Vue({  el: '#app',  data: {  title: 'Hello Vuexxxxx!',  count:0  },  beforeCreate: function() {  console.log('beforeCreate')  },  created: function() {  console.log('created')  },  beforeMount: function() {  console.log('beforeMount')  },  mounted: function() {  console.log('mounted')  },  beforeUpdate: function() {  console.log('beforeUpdate')  },  updated: function() {  console.log('updated')  },  beforeDestroy: function() {  console.log('beforeDestroy')  },  destroyed: function() {  console.log('destroyed')  },  methods: {  destroyAction: function() {  this.$destroy();  }  }  })  Khi destroy() dc call thì có nghĩ là vòng đời của instance vuejs này đã kết thúc lúc này ta ko thể thao tác đc gì trên element đã gắn ms Vuejs | |
| Dùng server để chạy vueJs <http://127.0.0.1:5500/api/index.html>  chứ ko phải dùng link trực tiếp [file:///var/www/Vuejs/VueJS2/api/index.html](http://vuejs2/api/index.html)  vì khi dùng server nó sẽ trải qua 1 cái bước load những file cần thiết sử dụng trong dự án của vuejs như thế sẽ tối ưu hóa đc dung lượng phải load, cải thiện tốc độ load trang. Còn nếu dùng link path trực tiếp từ ổ cứng thì ta sẽ phải load hết tất cả file có lên như vậy sẽ ko tối ưu  <https://www.youtube.com/watch?v=PSQ0ZlDIZY4&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=59> | |
| **Props**  props dungf để truyền dữ liệu từ component cha sang component con để có thể dùng dc  // truyền props name từ component cha  <userDetail name="ok di em"></userDetail>  <!-- truyền prop 'name' có value bằng với thuộc tính name của component cha -->  <userDetail :name="name"></userDetail>  gọi trong component con  <script>  export default {  name: 'app',  // props dungf để truyền dữ liệu từ component cha sang component con để có thể dùng dc  props:['name']  }  </script> Validating trong Props props: {  nameUser: {  type: Object,  required: true,  default: 'ok di em' // nếu biến ko đc truyền thì sẽ có value default là ntn  },  },   * **$emit** * dungf để gửi event đc định nghĩa từ con lại cho cha để thông báo 1 vấn đề gì đấy * $emit có thể định nghia cả các event bình thường: input, click để có thể dùng khi apply v-model vào và v-model sẽ theo dõi event và hoạt động như với 1 thẻ input thông thường * this.$emit('nameWasReset', this.nameUser) * như vậy ở component cha ta sẽ theo dõi event có tên đc định nghĩa như trên để biết lúc nào nó đc xảy ra * <userDetail :nameUser="name" @nameWasReset="name = $event"></userDetail> | |
| **giao tiếp giữa các component qua call back**   * nguyên lý là ta sẽ truyền cả function từ cha sang con để dùng props để quản lý, khi bên component child xử lý gì thì mình call đến cái function cha đấy từ đấy sẽ ko phải dùng $emit để định nghĩa 1 event mới nữa * khai báo ở component cha, resetNameFn là 1 function   <userDetail :resetNameFn ="resetNameFn" >  </userDetail>  // bắt props truyền vào từ cha  props: {  resetNameFn : {  type: Function // type của props call back thì phải là function  }  }, | |
| **2 component cùng cấp thì ko thể giao tiếp trực tiếp ms nhau mà phải thông qua thằng parent. Component 1 muốn thay đổi gì của bên component 2 thì phải gửi lên parent rồi từ parent thay đổi sẽ ảnh hưởng đến component 2** | |
| **Giao tiếp trực tiếp giữa 2 component mà ko thông quan parent**   * cách này thì sẽ ko gây thay đổi value của parent component   <https://www.youtube.com/watch?v=KLZkBYtP-C0&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=84> | |
| **Slot**  <https://www.youtube.com/watch?v=sR3CswhrMCI&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=88>  dùng để đánh dấu chỗ sẽ in ra html đc truyền từ component cha sang con  <div class="title">  <slot name="title"></slot>  </div>  <div class="content">  <slot name="content"></slot>  </div>  <appCard>  <h2 slot="title">Car {{title}}</h2>  <p slot="content">This is my car</p>  <p slot="content">ok di em</p>  </appCard>   * ta có thể đặt tên cho slot rồi call ở thẻ cha để gán html cho từng vị trí slot riêng biệt trong case component con có nhiều slot ở nhiều vị trí  Default Slots  * nếu slot nào ko đc đặt tên và ở bên thẻ cha các html ko đc chỉ định vào 1 slot name cụ thể nào thì khi render các html đấy sẽ vào hết các slow ko có tên * <https://www.youtube.com/watch?v=F0yfjwUhftM&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=91> | |
| **thẻ <component> trong vue js**  <https://www.youtube.com/watch?v=KA_qWwne88Y&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=93>  vue js cung cấp 1 thẻ component có thuộc tính :is=’tên biến’ tên biến phải có value là tên của componet đc khai báo, như thế thì nó sẽ call cái component đấy và render ra   * có thể hiểu như là 1 dạng router, khi value biến = component nào thì nó sẽ call component đấy ra * <button class="btn btn-outline-danger" @click="selectComponent = 'appCard'">Card</button> * <button class="btn btn-outline-danger" @click="selectComponent = 'appAbout'">About</button> * <button class="btn btn-outline-danger" @click="selectComponent = 'appContact'">Contact</button> * <component :is="selectComponent"> * <h2 slot="title">Car {{ title }}</h2> * <p slot="content">This is my car</p> * <p>ok di em</p> * </component> * ở đây biến selectComponent đc khai báo để mang value tên các component như vậy khi click vào button nào thì selectComponent sẽ thay đổi value và sẽ đc render component đấy ra * <script> * import Card from "./components/Card.vue"; * import Contact from "./components/Contact.vue"; * import About from "./components/About.vue"; * export default { * data() { * return { * title: "Title App", * selectComponent:'appCard', * }; * }, * components: { * appCard: Card, * appContact: Contact, * appAbout: About, * }, * }; * </script> * Khi chuyển giữa các component ntn thì các vòng đời của component bị destroyed vì thế muốn giữ chúng in live khi chuyển qua lại ta sẽ sử dụng thẻ keep-alive của Vuejs bao lấy thẻ <component> * <keep-alive> * <component :is="selectComponent"> * </component> * </keep-alive> | |
| component Lifecycle Hooks  * Vuejs cung cấp thêm 2 thuộc tính để theo dõi lúc nào component đc sử dụng lúc nào thì tạm ngưng đấy là **deactivated() và activated()** * <script> * export default { * data() { * return { * count:0 * } * }, * destroyed() { * console.log('destroyed') * }, * deactivated() { * console.log('deactivated') * }, * activated() { * console.log('activated') * }, * }; * </script> * **deactivated khác với destroyed** ở chỗ là nó sẽ chỉ tạm dừng hoạt động của component còn destroyed thì nó dừng luôn component kết thúc vòng đời, khi call lại component thfi nó sẽ bắt đầu 1 vòng đời mới | |
| **@click.native**   * để event click phân biệt đc click vào element vào trong case có nhiều element giống nhau đc render qua for each * <appItems v-for="(item, index) in items" v-bind:data="item" v-bind:key="item" @click.native="removeItems(index)">{{item}}</appItems> | |
| **v-model.lazy**  <**input type="text" id="email" v-model.lazy="userData**.**email" class="form-control"**>  <**p**>{{**userData**.**email**}}</**p**> | dùng để hoãn realtime khi nhập input, khi nào ng dùng nhập xong và click ra khỏi thẻ input thì lúc đấy ms apply giá trị vừa nhập   * áp dụng khi input email, thì ta đợi ng dùng nhập xong r ms check validate chứ ko phải ng dùng nhập ký tự nào mình cũng validate |
| <https://v2.vuejs.org/v2/guide/forms.html#Modifiers>  <input v-model.number="age" type="number"> | If you want user input to be automatically typecast as a Number, you can add the number modifier to your v-model managed inputs: |
| <input v-model.trim="msg">  có thể chồng các modifier lên ms nhau như sau  **v-model.lazy.trim** | If you want whitespace from user input to be trimmed automatically, you can add the trim modifier to your v-model-managed inputs: |
| Bind dữ liệu textarea và lưu kí tự xuống dòng sử dụng style ntn để nhận biết ddc các ký tự xuống dòng để render cho phù hợp  <**p style="white-space: pre"**>Messages{{**message**}}:</**p**> Sử dụng Radio Buttons cho form trong Vue.js ta cần tạo 1 thuộc tính dạng mảng để lưu  **sendEmail**:[] khi checkbox tich thì value sẽ đc add vào mảng sendEmail  <**div class="form-check"**>  <**input type="checkbox" class="form-check-input" value="Send Mail" id="sendmail" v-model="sendEmail"**>  <**label class="form-check-label" for="sendmail"**>Send Mail</**label**>  </**div**>  <**div class="form-check"**>  <**input type="checkbox" class="form-check-input" value="Send Info Mail" id="send-info-mail" v-model="sendEmail"**>  <**label class="form-check-label" for="send-info-mail"**>Send InfoMail</**label**>  </**div**>  <**div class="form-check"**>  <**input type="checkbox" class="form-check-input" value="Cut Mail" id="cutmail" v-model="sendEmail"**>  <**label class="form-check-label" for="cutmail"**>Cut Mail</**label**>  </**div**> Sử dụng Radio Buttons cho form trong Vue.js  * cần 1 biến có dạng string , và default value nên = 1 value của input radio nào đấy * **gender**:**'Male'**   <**div class="form-check"**>  <**input type="radio" class="form-check-input" id="male" value="Male" v-model="gender"**>  <**label class="form-check-label" for="male"**>Male</**label**>  </**div**>  <**div class="form-check"**>  <**input type="radio" class="form-check-input" id="female" value="Female" v-model="gender"**>  <**label class="form-check-label" for="female"**>Female</**label**>  </**div**>  **Sử dụng custom event để theo dõi change chứ ko dùng v-model**  <**input type="text" id="email"**  **:value="userData**.**email"**  **@input="userData**.**email** = $event.**target**.**value"**  **class="form-control"**>  <**p**>{{**userData**.**email**}}</**p**>  apply **v-model** đồi với component để thay đổi giá trị của 1 data value  <https://www.youtube.com/watch?v=dMPV21eL4-w&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=113>  khi $emit ta sẽ sử dụng emit input như vậy khi nào v-model sẽ bắt đc event input đc chạy vào sẽ tiền hành update value như với 1 thẻ input thông thường  <**appCustomControl v-model="dataSwitch"**></**appCustomControl**>  // component **appCustomControl**  <**div id="on" @click="**switched(**true**)**" :class="**{**active**:value}**"**>ON</**div**>  <**div id="off" @click="**switched(**false**)**" :class="**{**active**:!value}**"**>Off</**div**>  **export default** {  **props**:[**'value'**],  **methods**: {  switched(isOn){  **this**.$emit(**'input'**, isOn)  }  }  } | |
| Submit một form và chú ý prevent trong Vue.js **@click.prevent="**submited**"**  sử dụng prevent để tránh load lại trang và sẽ chạy vào method xử lý mà mình mong muốn | |
| **v-text="'ok di em'"** | dùng để khai báo content cho thẻ, lưu ý text phải để trong dấu ‘ ’  <**p v-text="'ok di em'"**></**p**> |
| **Khai baos Directives**   * <https://v2.vuejs.org/v2/guide/custom-directive.html#ad> * khai báo thêm nhiều tính năng để dùng lại trong nhiều thẻ khác nhau: * v-tangsize, v-giamsize, v-backgroundRed, …. có thể tạo nhiều biến với đa dạng các loại name   VD: tạo directive tên là **'highlight' để setbackground cho thẻ khai báo** directive này  Vue.directive(**'highlight'**, {  bind(el,binding,vnode) {  el.**style**.**backgroundColor** = **'yellow'**;  }  });  <**span v-highlight**>xxxxxx</**span**>   * Cách tạo 2 có thể viết trong file .vue   <**script**>  **import** CustomControl **from "./components/CustomControl"**;  **export default** {  **directives**: {  **'local-custom'** : {  bind(el,binding,vnode) {  **var** delay = 0;  **if**(binding.**modifiers**[**'delayed'**]) {  delay = 3000;  }  *setTimeout*(() => {  **if**(binding.**arg** == **'test'**) {  el.**style**.**backgroundColor** = binding.**value**;  } **else** {  el.**style**.**color** = binding.**value**;  }  }, delay);  }  }  }  }  </**script**>   * Truyền giá trị vào directive ta sử dụng đến param binding * VD:   Vue.directive(**'highlight'**, {  bind(el,binding,vnode) {  el.**style**.**backgroundColor** = binding.**value**;  }  });  <**span v-highlight="'red'"**>xxxxxx</**span**> Truyền tham biến trong Custom Directives **v-highlight:test="'green'"**   * ta truyền tham biến tên là “test” vào ,rồi check khi nào argument bằng “test” thì mới cho hoạt động còn không sẽ chỉ đổi màu   Vue.directive(**'highlight'**, {  bind(el,binding,vnode) {  **if**(binding.**arg** == **'test'**) {  el.**style**.**backgroundColor** = binding.**value**;  } **else** {  el.**style**.**color** = binding.**value**;  }  *// el.style.backgroundColor = 'green';*  }  }); Modify một custom Directives <https://www.youtube.com/watch?v=3p0gF9b-gpw&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=120>   * thêm modife delayed trong khai báo, sau đó check modifiers = binding.**modifiers**   <**span v-highlight:test.delayed="'green'"**>xxxxxx</**span**>  Vue.directive(**'highlight'**, {  bind(el,binding,vnode) {  **var** delay = 0;  **if**(binding.**modifiers**[**'delayed'**]) {  delay = 3000;  }  *setTimeout*(() => {  **if**(binding.**arg** == **'test'**) {  el.**style**.**backgroundColor** = binding.**value**;  } **else** {  el.**style**.**color** = binding.**value**;  }  }, delay);  }  });   * **Truyền value phức tạp vào directive:** * <**span v-local-custom:test.delay.blink="**{**mainColor**:**'red'**, **secondColor**:**'yellow'**, **delay**:500}**"**>test custom</**span**> * Trong directive sẽ lấy ra bằng cách: * **let** mainColor = binding.**value**.**mainColor**; * **let** secondColor = binding.**value**.**secondColor**; | |
| local filter <https://v2.vuejs.org/v2/guide/filters.html#ad>  Vue.js allows you to define filters that can be used to apply common text formatting. Filters are usable in two places: mustache interpolations and v-bind expressions (the latter supported in 2.1.0+). Filters should be appended to the end of the JavaScript expression  call <**span**>{{**text** | lowerCase}}</**span**> xử lý value của biến text với filter lowerCase   * <**script**> * **export default** { * **name**: **'app'**, * data () { * **return** { * **text**: **'Hello Everyone, Vuejs'** * } * }, * **filters**: { * lowerCase(text) { * text = text.toString(); * **return** text.toLowerCase(); * } * } * } * </**script**> * có thể apply nhiều filter cùng 1 lúc * When the global filter has the same name as the local filter, the local filter will be preferred. * <**span**>{{**text** | lowerCase | upCase}}</**span**> | |
| **tạo 1 input filter đơn giản**  <**input type="text" class="form-control" v-model="filterProduct"**>  <**div class="card"**>  <**div class="card-body"**>  <**ul**>  <**li v-for="*product* in** filterProducts**" v-bind:key="*product*"**>{{***product***}}</**li**>  </**ul**>  </**div**>  </**div**>  **export default** {  **name**: **'app'**,  data () {  **return** {  **text**: **'Hello Everyone, Vuejs'**,  **products**:[**'Iphone'**, **'Samsung'**, **'Nokia'**, **'Motorola'**, **'Oppo'**, **'LG'**, **'Sony'**],  **filterProduct**:**''**  }  },    **computed**: {  filterProducts() {  **return this**.**products**.filter((element) => {  **return** element.match(**this**.**filterProduct**);  })  }  }  }  </**script**> | |
| mixins <https://www.youtube.com/watch?v=R1NF1s-UdKk&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=127>  <https://v2.vuejs.org/v2/guide/mixins.html#ad>  Mixins are a flexible way to distribute reusable functionalities for Vue components. A mixin object can contain any component options. When a component uses a mixin, all options in the mixin will be “mixed” into the component’s own options.  Mixin là một cách linh hoạt để phân phối các chức năng có thể tái sử dụng cho các thành phần Vue.  Mixin sẽ chia sẻ dữ liệu của mình mới các vue khác nhưng khi các vue khác thay đổi dữ liệu đó thì Mixin sẽ vẫn giữ nguyên dữ liệu đầu và ko thay đổi theo. Mixin sẽ đóng vai trò như 1 ng trung gian cung cấp dữ liệu, function nhưng sẽ ko bị ảnh hưởng bởi các bên khác Global Mixins <https://www.youtube.com/watch?v=P3cffkMDc6U&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=130>  khai báo trong main.js  **import** Vue **from 'vue'**  Vue.mixin({  created() {  ***console***.log(**'Global Mixins - create hook'**);  }  });  Vì nó là global mixin nên nó sẽ đc call ở mọi Vue khi khởi tạo dù b có muốn hay ko nên có thể sẽ gây ra tình trạng xung đột của 1 vue nào đấy ms thằng mixin đc khai báo kiểu global ntn | |
| **transition**  <https://v2.vuejs.org/v2/guide/transitions.html#ad>  <https://www.youtube.com/watch?v=pg4NVG1IBQc&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=133> [Transition Classes](https://v2.vuejs.org/v2/guide/transitions.html#Transition-Classes)  * v-enter: Starting state for enter. Added before element is inserted, removed one frame after element is inserted. * v-enter-active: Active state for enter. Applied during the entire entering phase. Added before element is inserted, removed when transition/animation finishes. This class can be used to define the duration, delay and easing curve for the entering transition. * v-enter-to: Only available in versions 2.1.8+. Ending state for enter. Added one frame after element is inserted (at the same time v-enter is removed), removed when transition/animation finishes. * v-leave: Starting state for leave. Added immediately when a leaving transition is triggered, removed after one frame. * v-leave-active: Active state for leave. Applied during the entire leaving phase. Added immediately when leave transition is triggered, removed when the transition/animation finishes. This class can be used to define the duration, delay and easing curve for the leaving transition. * v-leave-to: Only available in versions 2.1.8+. Ending state for leave. Added one frame after a leaving transition is triggered (at the same time v-leave is removed), removed when the transition/animation finishes. * apply transition cho 1 the element, rồi sau đó viết style theo tên transition đó, nếu apply cho 2 element cùng 1 lúc thì sẽ báo lỗi. Nhưng có thể dùng if else để thay đổi luôn phiên giữ 2 element   VD: <**transition name="slide"**>  <**div class="alert alert-success" v-if="show"**>  This is some thing infomation alert-success  </**div**>  <**div class="alert alert-danger" v-else="show"**>  This is some thing infomation alert-danger  </**div**>  </**transition**>  <**transition name="fade"**>  <**div class="alert alert-success" v-if="show"**>  This is some thing infomation  </**div**>  </**transition**>  <**style**>  */\*enter\*/*  .**fade-enter** {  **opacity**: 0;  }  .**fade-enter-active** {  **transition**: **opacity** .5**s**;  }  */\*leave\*/*  .**fade-leave-to** {  **opacity**: 0;  }  .**fade-leave-active** {  **transition**: **opacity** .5**s**;  }  </**style**>   * appear : thuộc tính này để thực hiện hiệu ứng ngay khi load trang trong trường hợp thẻ element sẽ luôn dc hiển thị ngay từ đầu   <**transition name="slide" appear**>  <**div class="alert alert-warning" v-show="show"**>  This is some thing infomation  </**div**>  </**transition**>  -**enter-active-class="animate\_\_animated animate\_\_shakeX"**  **leave-active-class="animate\_\_animated animate\_\_wobble"**  thay vì đặt tên qua attribute name để tự thêm css thì ta có thể dùng thư viện hiệu ứng bên ngoài rồi add class của thư viện dựa vào 2 thuộc tính **enter-active-class và leave-active-class để có thể add class khi transition enter or leave**  **mode="out-in"**  <**transition name="slide" mode="out-in"**>  <**div class="alert alert-success" v-if="show" key="success"**>  This is some thing infomation alert-success  </**div**>  <**div class="alert alert-danger" v-else="show" key="danger"**>  This is some thing infomation alert-danger  </**div**>  </**transition**>  thêm thuộc tính **mode="out-in"** để chuyển qua loại các element trong if else mượt hơn  **transition-group**   * để khai báo nhiều element và apply hiệu ứng * vd:   <**ul class="list-group"**>  <**transition-group name="slide"**>  <**li class="list-group-item"**  **style="cursor: pointer"**  **v-for="**(***number***, ***index***) **in numbers"**  **v-bind:key="*index*"**  **@click="**removeItem(***index***)**"**>  {{***number***}} - {{***index***}}  </**li**>  </**transition-group**>  </**ul**> | |
| **vue-resource**  <https://github.com/pagekit/vue-resource>   * khai báo trong main.js   **import Vue from 'vue'**  **import VueResource from 'vue-resource'**  **import App from './App.vue'**  **Vue.use(VueResource);**  **new Vue({**  **el: '#app',**  **render: h => h(App)**  **})**  **Filebase**  <https://www.youtube.com/watch?v=kyuxGIvCjps&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=147>   * để tạo Nosql để có thể thêm sửa xóa data .   **có thể khai báo root của http để cho dễ sử dụng http ko cần phải khai báo lại url dài dòng**   * Vue.**http**.options.**root** = **'**[**https://vue-ytb-form-f4678-default-rtdb.firebaseio.com/data.json**](https://vue-ytb-form-f4678-default-rtdb.firebaseio.com/data.json)**'**;   getAllUser: **function** () {  **this**.**$http**.get(**''**).then(response => {  **return** response.json();  }).then(data=> {  **const** newArr = [];  **for**(**let** i **in** data) {  newArr.push(data[i]);  }  ***console***.log(data);  ***console***.log(newArr);  **this**.**users** = newArr;  });  }   * https://www.youtube.com/watch?v=6XI\_kwDa6qQ&list=PLU4OBh9yHE95G\_Y1cUVY-5Mc9P-rQBY3F&index=149 | |
| **interceptors**  Interceptor có thể hiểu như một bước tường lưới chặn các request, response của ứng dụng để cho phép kiểm tra, thêm vào header hoặc thay đổi các param của request, response. Nó cho phép chúng ta kiểm tra các token ứng dụng, Content-Type hoặc tự thêm các header vào request. Điều này cho phép chúng ta tận dụng tối đa thao tác chỉnh sửa header, body, param request gửi lên server sao cho hợp lý nhất, bảo mật nhất.  <https://github.com/pagekit/vue-resource/blob/master/docs/http.md>  <https://www.youtube.com/watch?v=mpRELupZen8&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=150>  Vue.**http**.interceptors.push((request, next) => {  ***console***.log(request);  **if**(request.**method** == **'POST'**) {  }  next();  }); | |
| **VueRouter**  npm install vue-router@3 --force  <https://www.youtube.com/watch?v=n6MWPVc4wco&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=155>  **BUild router-link**  <**router-link to="/" class="nav-link"**>Home</**router-link**>  <**router-link to="/" tag="a" class="nav-link"**>Home</**router-link**>  <**router-link**  **to="/"**  **tag="li"**  **class="nav-item"**  **active-class="active"**  **exact**  >  <**a class="nav-link"**>Home</**a**>  </**router-link**>  <https://v3.router.vuejs.org/api/#v-slot-api-3-1-0>  <https://www.youtube.com/watch?v=CM3xmbfprPY&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=159>  <**router-view**></**router-view**>  **Remove dấu # trên url vue router**  <https://stackoverflow.com/questions/34623833/vue-js-how-to-remove-hashbang-from-url>  Liên quan đến mode của router  **const** router = **new** VueRouter({  routes,  **mode**:**'history'**  });  **Taoj button Go Back sử dụng this.$router.push('/')**  <**template**>  <**div**>  <**h1**>User Page</**h1**>  <**hr**>  <**button type="button" class="btn btn-primary" @click="**backHome**"**>Back To Home</**button**>  </**div**>  </**template**>  <**script**>  **export default** {  **methods**: {  backHome: **function** () {  **this**.**$router**.push(**'/'**)  }  }  }  </**script**> Truyền param vào trong routes {{ **$route**.**params**.id }} gọi value param  {**path**: **'/user/:id'**, **name**:**'User Detail'**, **component**: UserDetail}, Theo dõi sự thay đổi của params **ta ssử dụng watch của vuejs để theo dõi sự thay đổi**  **watch**: {  $route : **function** (to, form) {  ***console***.log(form.**params**.**id**);  ***console***.log(to.**params**.**id**);  }  } Routes Child {**path**: **'/user'**, **name**:**'User'**, **component**: User,  **children**: [  {**path**: **''**, **name**:**'Index'**, **component**: UserStart},  {**path**: **':id'**, **name**:**'User Detail'**, **component**: UserDetail},  {**path**: **'edit/:id'**, **name**:**'User Edit'**, **component**: UserEdit},  ],  },  **Call router:**  **-cachs 1 dùng even click**  <**button class="btn btn-warning" @click="**goEdit**"**>Edit id {{**id**}}</**button**>  goEdit : **function** () {  **this**.**$router**.push(**`edit/**${**this**.**id**}**`**)  }  **Cách 2 dùng router-link**  <https://v3.router.vuejs.org/api/>  <**router-link :to="**'edit/' + **id" tag="button" class="btn btn-warning" active-class="active"**><**a class="nav-link"**>Edit id {{**id**}}</**a**></**router-link**>  <**router-link :to="**`edit/${**id**}`**" tag="button" class="btn btn-warning" active-class="active"**><**a class="nav-link"**>Edit id {{**id**}}</**a**></**router-link**>  **Router params and query**  <**router-link :to="**{**name**:'user\_edit', **params**:{**id**:**$route**.**params**.**id**}, **query**:{**locate**:'vi', **browser**:'chrome'}}**" tag="button" class="btn btn-warning" active-class="active"**><**a class="nav-link"**>Edit</**a**></**router-link**>  **Router View**  Router view sẽ render theo cấp độ, khai báo ở thằng cha sẽ là nơi để render các thằng con mà tằng cha đó call ra  <**router-view**></**router-view**>  <**router-view name="page-header"**></**router-view**>  <**router-view**></**router-view**>  Khi mà muốn phân vùng router-view, chỉ định rõ router view nào sẽ show các component gì thì khi khai báo router ta sẽ sử dụng name của router-view để khai báo các component sẽ hiển thị riêng cho nó. Như thế ta sẽ xử lý đc case vd: home page sẽ có header footer nhưng trang checkout thì ko cần header footer  <https://v3.router.vuejs.org/guide/essentials/named-views.html#nested-named-views>  <**router-view name="page-header"**>  </**router-view**>  <**router-view**></**router-view**>  Thuộc tính default để render vào các **router-view ko có name**  {  **path**: **'/'**, **name**: **'home\_page'**, **components**: {  **default**: Home,  **'page-header'**: PageHeader  }  },  Redirect tới một trang khác <https://v3.router.vuejs.org/guide/essentials/redirect-and-alias.html#redirect>  **const** router = **new** VueRouter({  **routes**: [  { **path**: **'/a'**, **redirect**: **'/b'** }  ]  })  Redirecting is also done in the routes configuration. To redirect from /a to /b: Alias  * Sử dụng giống ms url rewrite, url nhập 1 đằng nhưng nó sẽ tự hiểu để vào đúng 1 router có name khác   A redirect means when the user visits /a, the URL will be replaced by /b, and then matched as /b. But what is an alias?  An alias of /a as /b means when the user visits /b, the URL remains /b, but it will be matched as if the user is visiting /a.  The above can be expressed in the  **const** router = **new** VueRouter({  **routes**: [  { **path**: **'/a'**, **component**: A, **alias**: **'/b'** }  ]  }) Sử dụng transition cho router vẫn sử dụng transition bthg 👍  <**transition name="slide"**>  <**router-view**></**router-view**>  </**transition**>  <**style**>  */\*Slide transition\*/*  .**slide-enter** {  }  .**slide-enter-active** {  **animation**: **slide-in** .2**s ease-out forwards**;  **position**: **absolute**;  }  .**slide-leave-to** {  }  .**slide-leave-active** {  **animation**: **slide-out** .2**s ease-out forwards**;  }  **@keyframes slide-in** {  **from**{  **transform**: **translateX**(20**px**);  **opacity**: 0;  }  **to**{  **transform**: **translateX**(0);  }  }  **@keyframes slide-out** {  **from**{  **transform**: **translateX**(0);  }  **to**{  **transform**: **translateX**(20**px**);  **opacity**: 0;  }  }  </**style**> Scrollbehavior in Vuejs <https://v3.router.vuejs.org/guide/advanced/scroll-behavior.html#scroll-behavior>  DÙng để scroll đến 1 vị trí nào đó khi vừa load url mới or đổi sang page khác  có thể dùng x, y để mô tả tọa độ, or có thể dùng selector để call đến element cần scroll đến  scrollBehavior (to, from, savedPosition) {  **if**(savedPosition) {  **return** savedPosition;  }  ***console***.log(savedPosition);  ***console***.log(to);  ***console***.log(from);  *// return { x: 0, y: 1000 }*  **if**(to.**hash**) {  **return** {  **selector**: to.**hash**,  **behavior**: **'smooth'**,  };  }  } Bảo vệ routes với định nghĩa guards <https://www.youtube.com/watch?v=MEvxx1TaO7A&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=175>  <https://v3.router.vuejs.org/guide/advanced/navigation-guards.html#global-before-guards>  **beforeEach apply cho all router**  router.beforeEach((to, from, next) => {  ***console***.log(to);  ***console***.log(from);  ***console***.log(next);  next();  });  **beforeEnter dùng để khai báo cụ thể cho từng router path**  **VD: với router parent admin thì ta sẽ check authorization cho nó để check login hay chưa**  {  **path**: **':id'**, **name**: **'user\_detail'**,  **component**: UserDetail,  beforeEnter: (to, from, next) => {  ***console***.log(**'beforeEnter'**);  next();  }  }, In-Component Guards Finally, you can directly define route navigation guards inside route components (the ones passed to the router configuration) with the following options:   * beforeRouteEnter * beforeRouteUpdate * beforeRouteLeave   Ngoài ra còn có 3 thuộc tính có thể dùng trong component cùng level ms data, watch, methods, created,....:  **beforeRouteEnter: function (to, from, next) {**  **if(true) {**  **next();**  **} else {**  **next(false);**  **}**  **}**  tối ưu hóa với lazy load Vue.js file nào cần thì ms load và chỉ load 1 lần chứ khi đổi router sẽ ko phải load lại <https://www.youtube.com/watch?v=N3xSGwGex8w&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=177>  **const** *User* = resolve => {  ***require***.ensure([**'./components/user/User'**], () => {  resolve(***require***(**'./components/user/User'**))  }, **'user'**)  };  khai báo import file khi page cần sử dụng | |
| **VueX**  npm install vuex@next --save để cài bản mới nhất  npm install vuex@3 --save để cài version 3 cho vuejs2   * các giá trị của biến trong vueX sẽ thay đổi đồng bộ cùng nhau chứ ko phải ở các component khác nhau thì sẽ có các value khác nhau.   <https://vuex.vuejs.org/installation.html#npm>  <https://www.youtube.com/watch?v=YIc1Z_nkg0c&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=180>   * **store.js**   **import Vue from 'vue'**  **import Vuex from 'vuex'**  **Vue.use(Vuex);**  **export const store = new Vuex.Store({**  **state: {**  **result: 0**  **}**  **});**   * **main.js**   **import Vue from 'vue'**  **import** App **from './App.vue'**  **import** {store} **from './store/store'**  **new** Vue({  **el**: **'#app'**,  store,  render: h => h(App)  });   * Sử dụng call   **export default** {  **computed**: {  result(){  **return this**.**$store**.**state**.**result**;  }  }  } mapGetters <https://www.youtube.com/watch?v=TrOFXOZGu58&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=183>  để có thể tự map tên biến của getter với tên hiện tại dùng trong component   * thay vì phải khai báo nhiều ntn   **computed**: {  tenResult() {  **return this**.**$store**.**getters**.tenResult;  },  nameResult() {  **return this**.**$store**.**getters**.nameResult;  }  }   * ta sẽ dùng   **import** {mapGetters} **from "vuex"**  **export default** {  *// computed: {*  *// tenResult() {*  *// return this.$store.getters.tenResult;*  *// },*  *// nameResult() {*  *// return this.$store.getters.nameResult;*  *// }*  *// }*  *// dấu ... là thể hiện cú phám của ES6 có nghĩa là thêm những dữ liệu này vào trong mảng computed*  ***computed****: {*  *...mapGetters([****'tenResult'****,* ***'nameResult'****])*  *}*  } | |
| Mutations thay đổi state từ components Vue.js  * khác nhau giữa getter và mutation là getter ta có thể call trực tiếp hành động còn mutation thì ta phải call qua commit * <https://v3.vuex.vuejs.org/> * <https://v3.vuex.vuejs.org/guide/mutations.html#commit-with-payload>   <https://www.youtube.com/watch?v=aG_XXU-IyvI&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=184>   * khai báo trong store.js   **mutations** : {  // state luôn là 1 biến cố định phải khai báo, n là parameter thêm vào  incrementOp(state, n) {  **return** state.**result** += n;  }  }   * sử dụng **this**.**$store**.**commit**(**'incrementOp'**, 30);   call ra bằng commit(tenMutation, tham số khác ngoài state)   * có thể sử dụng mapMutations Trong case có nhiều mutation   **import** {mapMutations} **from 'vuex'**;  **export default** {  **name**: **'app'**,  data() {  **return** {  }  },  **methods**: {  increment: **function** () {  *// this.$store.state.result++;*  **this**.**$store**.**commit**(**'incrementOp'**, 30);  },  *// decrement: function () {*  *// this.$store.state.result--;*  *// },*  ...mapMutations([  **'decrement'**  ])  }  } | |
| actions trong vuex <https://www.youtube.com/watch?v=mMtuViteEqU&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=186>  <https://v3.vuex.vuejs.org/guide/actions.html#dispatching-actions> | |
| v-model với vuex <https://www.youtube.com/watch?v=pha4Bu8pAkQ&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=188>  <**input type="text" v-model="*value*"**>  <**p**>Text test: {{***value***}}</**p**>   * **dispatch dùng để call action của vuex**   **value**: {  get() {  **return this**.**$store**.**getters**.**value**  },  set(value) {  **this**.**$store**.**dispatch**(**"updateValue"**, value)  }  } | |
| Chia nhỏ tiếp các thành phần bên trong store Vue.js <https://www.youtube.com/watch?v=jgHS3NhRVCc&list=PLU4OBh9yHE95G_Y1cUVY-5Mc9P-rQBY3F&index=190>  **import** \* **as** actions **from './actions'**  **import** \* **as** getters **from './getters'**  **import** \* **as** mutations **from './mutations'**  Vue.use(Vuex);  **export const** store = **new** Vuex.**Store**({  **state**: {  **result**: 0,  **value**:**''**  },  getters,  mutations,  actions,  **modules**: {  Result  }  });  **trong các file js ta sẽ khai báo các function export cost ntn**  **export const *value* = function (state) {**  **return state.value;**  **};** | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |