Học Vuejs 2

sudo npm i -g vue-cli

vue init webpack-simple vueCliTest

- command dùng để build forder dist nếu bạn muốn :

npm run build

extension Visual Studio Code : Vetur

|  |  |
| --- | --- |
| 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](https://v2.vuejs.org/v2/guide/syntax.html" \l "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,  },  })  // 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> | |
| **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](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> | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |