**VueJS**

**el**: id nơi làm việc của vue

**data:** dữ liệu của vue

**methods:** các hàm xử lý trong vue

**computed:** Các hàm xử lý trong vue

**watch:** xem sự thay đổi của biến, khi biến thay đổi watch sẽ chạy.

===============================================================

**Sự khác nhau giữa methods và computed:**

+ **medthods:** khi biến được xử dụng dù có thay đổi giá trị hay không tất cả các hàm chứa biến đó đều chạy mặc dù không gọi tới.

+ **computed**: Chỉ gọi hàm khi biến đó thay đổi giá trị.

==> Nên xử dụng computed khi tính toán với dữ liệu trong data, medthods thì xử dụng với đối tượng ngoài data.

===============================================================

**:href** : gán giá trị đối tượng vue vào

**@click** : bắt sự kiện click từ vue

**@keyup:** bắt sự kiện từ bàn phím sử dụng: *@keyup.13 === @keyup.enter*

**@submit.prevent**: chặn chuyển hướng trang khi click submit của form trong vue || trong JS: e.preventDefault();

**@submit.stop:** chặn sự kiện của đối tượng cha khi chạm vào đối tượng con || trong JS: e.stopPropagation();

**v-model = “**biến trong data**” :** ràng buộc 2 chiều đối với thẻ input. Gía trị trong data bằng giá trị của thẻ input.

*vd:* <input v-model="firstName" type="text" placeholder="first name">

<p>FirstName: {{ firstName }}</p>

*-* ***v-once:*** *Giữ lại giá trị cũ của thuộc tính(không bị thay đổi).*

*=====================================================*

- **Ràng buộc class**: *v-bind:class=” { <tên class1>: true||false, …. } ”*

- **if else trong Vue:** v-if, v-else, v-else-if

- **for trong Vue:** v-for=”item in items”

- **Kết hợp v-for và –if:** *v-for=”item in items” v-if=”item.active”*

*-* ***Thêm phần tử hay thuộc tính cho array và object phải dùng “$set”***

***- v-html=”<tên biến>” :*** *Hiển thị nội dung là thẻ html*



*=====================================================*

*-* ***Props Dow: props****: [ “<tê thuộc tính>”, “<tên thuộc tính>”]. Props có thể truyền từ cha sang con -> con chỉ được xài chứ không được chỉnh sửa.*

*-* ***Event up: $emit(<tên hàm>)*** *truyền thông điệp cho Component cha biết là components con muốn thay đổi dữ liệu -> nhiệm vụ của components cha là nhận thông điệp và tiến hành thay đổi data.*

**- Ref:** khi click vào button này thì có thể gọi được sự kiện click ở input khác thông qua this.$refs.<ten ref đã đặt cho thẻ input đó>.click()

===============================================================

*-* **LifeCycle:** Theo thứ tự bao gồm các funtion()

+ **beforeCreated**: Chạy trước khi khởi tạo Vue.

+ **created**: Chạy sau khi Vue khởi tạo hoàn tất. Dùng gọi API,Call Ajax để lấy dữ liệu từ server về.

+ **beforeMount**: Chạy trong quá trình khởi tạo DOM.

+ **mounted**: Chạy sau khi khởi tạo DOM hoàn tất. Muốn sử dụng Jquery -> chỉ truy xuất DOM trong mounted -> có thể sử dụng jQuery.

+ **beforeDestroy:** Chạy trước khi kết thúc Vue.

+ **destroyed:** Chạy khi Vue đã kết thúc.

\*\*\* Chú ý: Với các methods LifeCycle thì không được dùng ‘ *hàm mũi tên*’hoặc lệnh gọi lại.

===============================================================

- **Object.assign**(<data muốn update>, <dataupdate>): thường dùng để edit dữ liệu nguồn.

**- Object.freeze()**: ngăn không cho các thuộc tính hiện có bị thay đổi.(đóng băng)

===============================================================

- **$el, $data, $watch(**'a', function (newValue, oldValue){ Lệnh gọi này sẽ gọi khi app.a thay đổi. }**) : Dấu $<”tên biến”>**  gọi biến cá thể của Vue, dấu $ dùng để phân biệt biến của Vue và biến thông thường.

**==========================Add VueX=============================**

**- npm install vuex –save :** add thư viện Vuex vào dự án Vue.

- **store Vuex:** import Vue from “vue” and import Vuex from “vuex”.

- **Vue.use(Vuex):** thêm Vuex vào Vue.

- **export const store = new Vuex.Store({ state, getters, mutations, actions }):** tạo một đối tượng Vuex.

- **state:** nơi lưu dữ liệu.

- **getters:** get dữ liệu, các components lấy dữ liệu bằng computed gọi đến các hàm trong getters. Không thể viết các hàm thay đổi dữ liệu nguồn ở đây.

**- mutations:** Sự kiện dùng để thay đổi dữ liệu nguồn. Các component sử dụng methods để viết các hàm gọi hàm trong mutations bằng ***this.$store.commit(‘<tên hàm>’).***

**-** **actions:** hành động, xử lý hành động cho các hàm ở mutations, vd:

Actions: {

reducePricce: context=>{

setTimeout(() => {

context.commit(‘reducePrice’);

}, 2000)

}

}

\*Components dùng ***this.$store.dispatch(‘tên hàm’)*** để gọi hàm trong actions.

- **Truyền biến từ methods -> actions -> mutations:** this.$store.commit(“tên hàm”, biến) || this.$store.dispatch(“ tên hàm”, tên biến.)

**- strict:** true || falsexác định mã javascript phải được thực thi ở chế độ nghiêm ngặt. vd: các hàm thay đổi dữ liệu nguồn phải được viết trong mutations.

===============================================================

- **setTimeout(() =>{ //method }, timeout):** hàm chờ trong thời gian timeout mới thực hiện hàm method.

===============================================================

- **babelrc:** thư viện dịch mã javascript để các trình duyệt web hiểu ngôn ngữ. để cài đặt ta sử dụng cú pháp: ***npm install babel-preset-stage-2 –save-dev***

**=========================Add Routing============================**

***- npm install vue-router –save:*** *add thư viện router cho dự án vuejs.*

**-** *import Router from "vue-router"*

*-Vue.use(Router);*

*- export default new Router({*

*routes: [*

*{ path: "/", name: "home", component: Home },*

*],*

*});*

**-** <router-link to=”/”>