TRƯỜNG ĐẠI HỌC PHENIKAA
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO DỰ ÁN HỌC PHẦN ĐỒ ÁN CƠ SỞ

Đề tài: Tìm hiểu và xây dựng ứng dụng với VueJS

Thành viên nhóm: **Chu Tuấn Hùng - 20010958**

Ngô Phú Khang - 20010774

Giảng viên hướng dẫn: ThS. Nguyễn Thị Thùy Liên

Hà Nội, ngày 07 tháng 12 năm 2022

MỤC LỤC

GIỚI THIỆU VỀ VUEJS

Vue.js là một framework linh động dùng để xây dựng giao diện người dùng trên trang web cũng như Single Page Application(SPA). Với cách viết tường minh dễ hiểu, Vue đã và đang phát triển tạo ra sự cạnh tranh với các framework, library có cùng chức năng.

Được phát triển bởi Evan You, lần ra mắt đầu tiên là tháng 2 năm 2014. Phiên bản ổn định nhất của Vue hiện nay là 3.2.40.

Với sự ra mắt của phiên bản vue 3, đã có rất nhiều thay đổi trong cách viết code của vue. Cụ thể trong vue 3 đã có những chức năng sau :

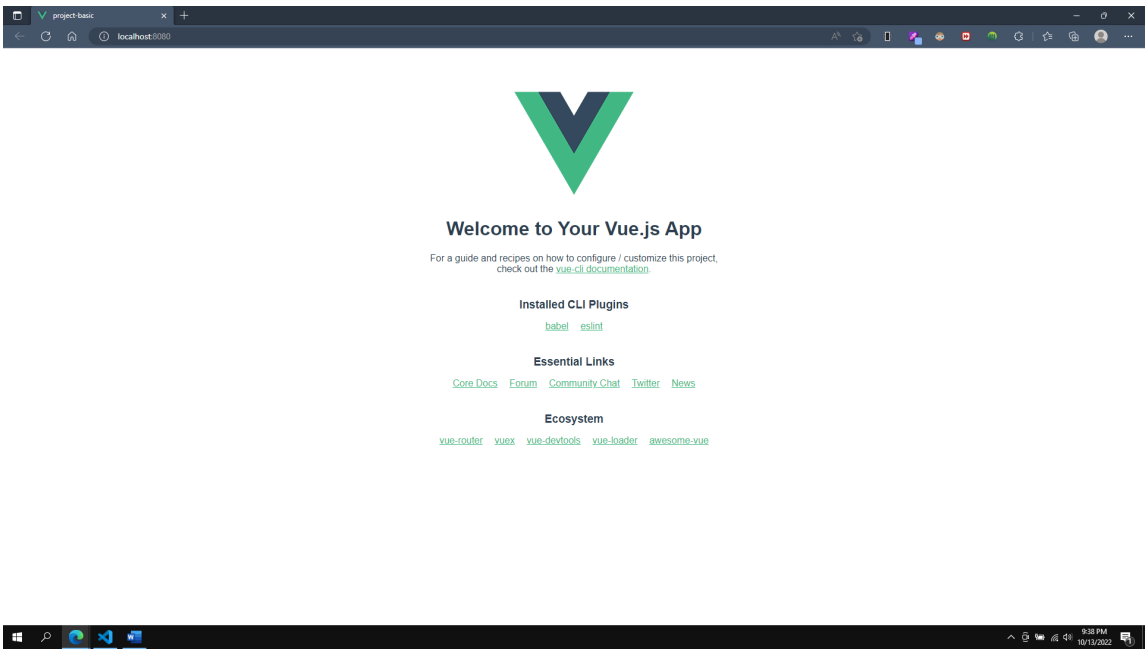
- Composition API (Now built-in).
- Multiple root elements (Template syntax).
- Suspense.
- Multiple V-models..
- Better Reactivity.

I. Cấu trúc thư mục của một project Vue.js

- Đầu tiên để có thể tạo được một project Vue, chúng ta cần cài đặt Vue CLI với phiên bản mới nhất của nó. Với Vue CLI, những bước cài đặt các gói cần thiết cho Vue sẽ đơn giản hơn rất nhiều so với việc cài thủ công những package đó.
- Câu lệnh để khởi tạo project vue có khuôn mẫu như sau :
    - Vue create [tên_project]

  Câu lệnh trên phải được chạy trên terminal tại thư mục mà chúng ta muốn đặt là nơi lưu trữ project của chúng ta. Sau khi chạy xong, khởi chạy project bằng câu lệnh tiếp theo :
    - Npm run serve

  Ảnh sau khi chạy thành công lần đầu sau khi khởi tạo

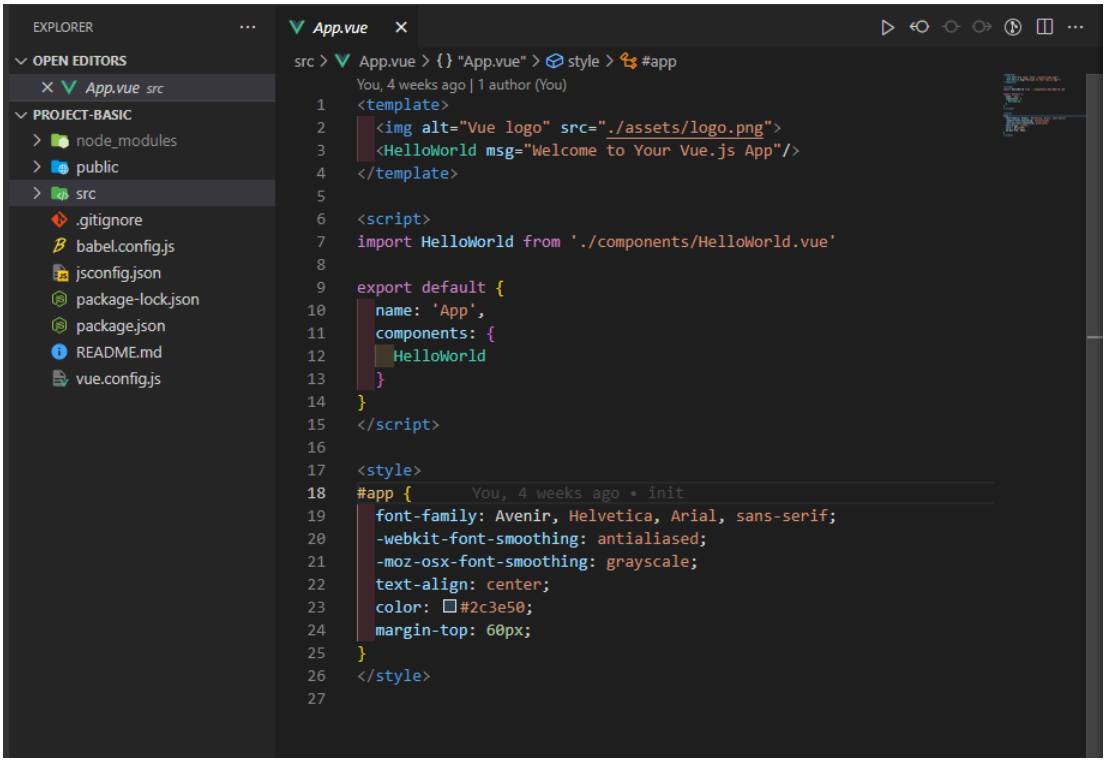


- Cấu trúc thư mục :
    - Node_modules
    - Public
    - Src
        - Assets
        - Components
        - Main.js
    - .gitignore

- Babel.config.js
- Jsconfig.json
- Package-lock.json
- Package.json
- Readme.md
- Vue.config.js

II. Thành phần chính của file .vue

- Trong mỗi file .vue có trong project, chúng sẽ có 3 thành phần chính. Lần lượt là <template>, <script> và <style>
- Đối với thẻ <template> là nơi chúng ta viết code html cho component đó trong vue.
- Trong vue nói riêng và các framework khác nói chung thì mỗi thành phần trên trang web đều nên phải được chia thành các thành phần con gọi là các components.
- Đối với thẻ <script> là nơi chúng ta viết code xử lý JS cho thành phần component, trong thẻ này chúng ta thực hiện việc đến các components con của component cha mà chúng ta đang sử dụng để viết code thông qua import trong JS.
- Cuối cùng thẻ <style> là nơi chúng ta viết code CSS cho thành phần component đó. (Chúng ta cũng có thể viết theo kiểu tách file).



III. Component, props, xử lý data trong VueJS

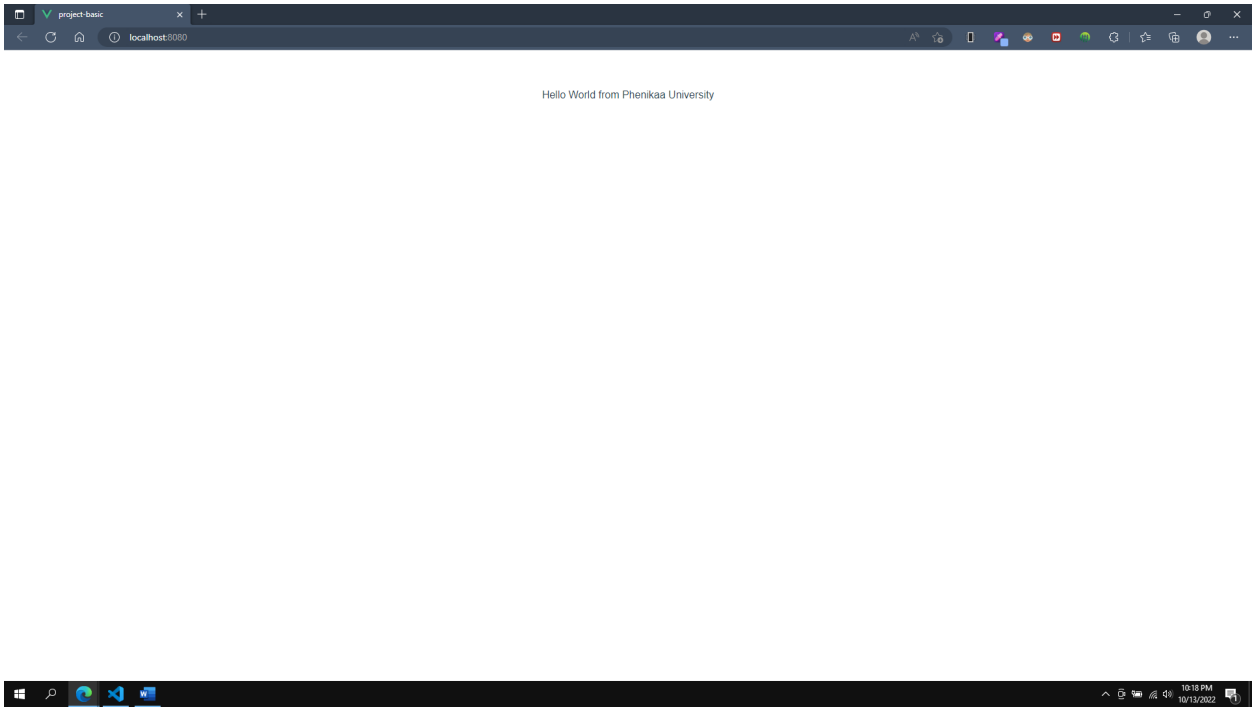
- Các component của trang web được lưu trữ trong thư mục Components, với file App.Vue là nơi tập hợp các component để tạo thành một trang web hoàn chỉnh.
- Cách chia component sẽ tùy theo từng chức năng, ý nghĩa của từng component. Và có một nguyên tắc đó là : "Component nào cầm dữ liệu thì component đó mới có quyền được thay đổi dữ liệu mà nó đang cầm, những component con của nó không có quyền thay đổi."
- Có hai loại component chính, tượng trưng cho hai chức năng chính của các components là : render dữ liệu và hiển thị dữ liệu.
- Props được hiểu là properties, là các thuộc tính của components các thuộc tính này có thể là một chuỗi, một số hoặc một object… Props có chức năng truyền dữ liệu từ component cha xuống cho component con có chức năng hiển thị, sử dụng dữ liệu mà chúng được truyền từ component cha.
- Khi component con muốn sử dụng props được truyền từ component cha thì trong thẻ <script> phải khai báo tên props được truyền thì mới có thể sử dụng được. Trong khi đó ở component cha muốn truyền props thì ta truyền thẳng tới thẻ component con.

```vue
<template>
  <HelloWorld msg="Hello World from Phenikaa University"/>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'App',      You, 4 weeks ago • init
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

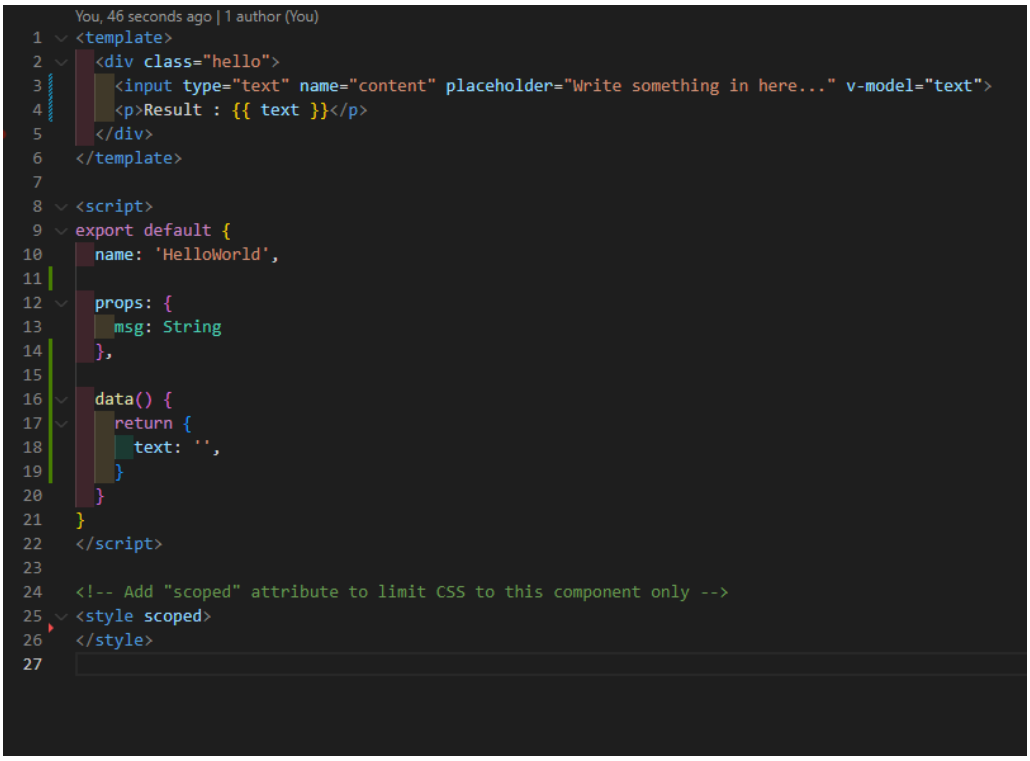
- Dòng số 2 chính là minh họa cho việc truyền props ở component cha xuống component con

```vue
<template>
  <div class="hello">
    {{ this.msg }}
  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  props: {
    msg: String
  }
}
</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>
</style>
```

- Hình trên là đoạn code làm hiển thị props sau khi component con nhận được, hiển thị ra màn hình nội dung của msg(tên của props được truyền).
- Kết quả:



- Trong vue còn có các directive, conditional có sẵn giúp các lập trình viên dễ dàng sử dụng hơn.
    - Two way binding : Trong vue chúng ta sử dụng v-models để có thể binding được dữ liệu theo cách two-way binding. V-models chỉ sử dụng được đối với các input đầu vào.
    - o



- Để có thể render ra một list data, Vue cung cấp cho người dùng directives v-for giúp tiết kiệm thời gian để render list. Cách sử dụng giống với vòng lặp for – in

  Ví dụ :

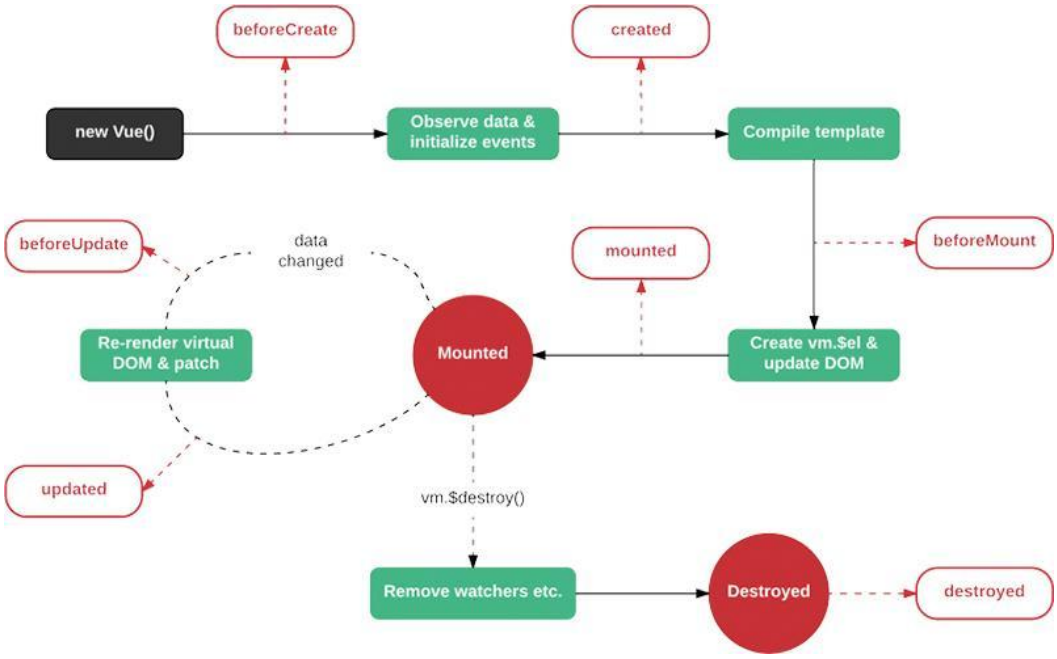
  Code :

```vue
<template>
  <div class="hello">
    <input type="text" name="content" placeholder="Write something in here..." v-model="text">
    <p>Result : {{ text }}</p>
  </div>

  <ul class="list-student">
    <li v-for="(student, index) in listStudent" :key="index"> {{ student }} </li>
  </ul>
</template>

<script>
export default {
  name: 'HelloWorld',

  props: {
    msg: String
  },

  data() {
    return {
      text: '',
      listStudent: [
        'Nguyen Van Nam',
        'Ngo Van Tu',
        'Nguyen Thi Hoa',
        'Do Nhu Tuan'
      ]
    }
  }
}      You, 4 weeks ago • init …
</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>
</style>

```

Kết quả :

- V-show/v-if/v-if-else/: là những directives giúp cho người dùng có thể ẩn hiện một element trong vue. Tuy nhiên với v-show === false thì nó sẽ thay đổi display của riêng element đó thành display none, ngược lại với các directives khác sẽ gỡ element đó ra khỏi cây DOM.

IV. Lifecycle Hooks của component vue.js



- Lifecycle hooks chính là những hàm, phương thức đặc biệt xử lý một chức năng. Cũng giống trong các framework khác, vue.js cũng có lifecycle hook.
- Lifecycle hooks được chia thành những giai đoạn chính sau :
    - Trước khi khởi tạo component.
    - Sau khi khởi tạo component.
    - Trước khi mount và sau khi mount.
    - Re-render và destroy.
- beforeCreate: Được gọi khi khởi tạo vue.js và trước khi thực hiện tiến trình obverse Data và init Events.
- Created : Được gọi khi tiến trình obverse Data và init Events hoàn thành.
- beforeMount: Được gọi ngay sau khi tiến trình render function hoàn tất
- mounted: Được gọi khi element được cấy vào cây DOM.
- beforeUpdate: Được gọi khi element có sự thay đổi.
- updated: Được gọi khi element đã có sự thay đổi.
- Ngoài các hook trên, vue còn có những hook sau :
    - Watcher : Theo dõi quá trình thay đổi data

    ….

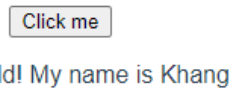
V. Event và custom event vue.js

- Theo document của vue, chúng ta có hai cách để có thể bắt sự kiện xảy ra trên element như sau :

- Sử dụng directive v-on:event=”tên_function xử lý”
- @tên_event=”tên_function xử lý”

- Tên event ở đây có nghĩa là những event mà chúng ta thường xử lý chúng với JS thuần, ví dụ event khi chúng ta click button làm ẩn hiện một element trên trang web, …
  Ví dụ :
  Ban đầu chỉ có một nút button Click me

  Sau khi bấm Click me sẽ xuất hiện element là thẻ p với nội dung là Hello World My name is Khang.



- Trong khi xử lý các event chúng ta cũng có thể xử lý các Event Modifiers, bằng cách thêm trực tiếp vào event đó. Ví dụ đối với event.preventDefault() chúng ta sẽ làm như sau :

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

  Hoặc cũng có thể xử lý nhiều event modifiers cùng một lúc :

  ```
  <a @click.stop.prevent=”doSth”>Click me</a>
  ```

- Ngoài ra chúng ta cũng có thể tạo được riêng cho mình những event như mong muốn bằng cách sử dụng $emit để truyền sự kiện từ component con lên component cha. Hoặc nếu muốn chúng ta cũng có thể truyền dữ liệu thông qua biến $root hoặc sử dụng provide/inject từ component ở bất kỳ cấp nào đến component mà chúng ta mong muốn.
- Tuy nhiên ở phiên bản vue 3, đã không còn hỗ trợ $on để bắt custom event từ $emit thay vào đó muốn sử dụng chúng ta phải cài đặt package có tên là eventbus trên github để có thể sử dụng được $on và $emit để bắt sự kiện.

VI. Vue Router

- Định nghĩa: Cũng giống với các framework khác, VueJS cũng sở hữu cho mình một thư viện riêng cho việc điều hướng người dùng trên ứng dụng. Được gọi là Vue Router. Phiên bản mới nhất của thư viện này hiện tại là v4.1.5
- Cài đặt:
  + Link CDN: https://unpkg.com/vue-router@4.1.5/dist/vue-router.global.js
  + NPM: npm install vue-router@4
  + Yarn: yarn add vue-router@4
- Cơ bản về Vue Router
  + Thành phần hiển thị: Cặp thẻ <router-view /> là thành phần được dùng để định nghĩa khu vực mà các router được mount vào ứng dụng. Các component được khai báo trong createRouter() sẽ được hiện thị trong khu vực <router-view /> này.
  + Chuyển tiếp giữa các router: Vue Router cung cấp một component là <router-link />. Component này có tác dụng giống thẻ <a/> trong HTML nhưng sẽ chỉ được dùng trong ứng dụng Vue. Để thực hiện chuyển hướng với <router-link />, ta thực hiện khai báo thẻ <router-link to=”/tag-target”></router-link>. Ngoài ra Vue Router còn cung cấp cho chúng ta một phương thức chuyển hướng thông qua obj router và qua phương thức push().

+ Tuyến đường động: Trong một số trường hợp, web app cần hiển thị nhiều dữ liệu, mỗi dữ liệu này có các thành phần giống nhau nhưng cần hiển thị trên các trang riêng biệt (VD: trang sản phẩm, các bài báo, tin tức, …) lúc này chúng ta cần sử đến tuyến đương động.

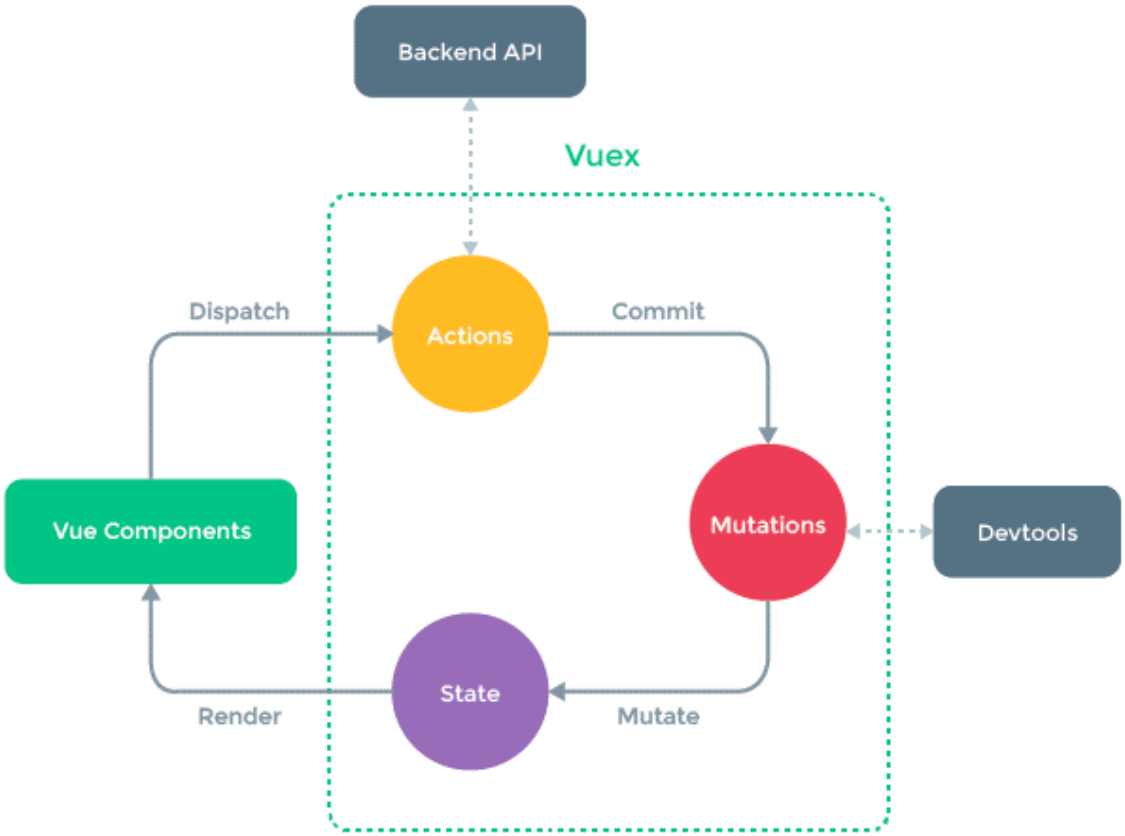
- Khai báo:

```
const routes = [
  // dynamic segments start with a colon
  { path: '/users/:id', component: User },
]
```

- Trong đó: thành phần :id là thành phần động được gọi là params.
  Params không nhất thiết phải cùng một giá trị. Thay đổi tùy thuộc vào người dùng.
- Khi thực hiện chuyển hướng ta thực hiện gọi <router-link to=”/users/1”</router-link>.
  + Named Router: Ta có thể thực hiện thêm tên cho router khi thực hiện khai báo. Điều này giúp chúng ta khi thực hiện chuyển hướng có thể gọi thông qua tên của Router
  + Named Views: Trong ứng dụng Vue có thể tồn tại nhiều <router-view />. Để thực hiện chuyển hướng chính xác thành phần cần thay đổi, ta cần đặt tên cho từng thành phần view.

VII. VueX / Pinia

- Để có thể lưu trữ, quản lý được dữ liệu/component phục vụ thuận lợi cho việc xử lý logic đối với các component có trong project chúng ta có thể sử dụng Pinia / VueX để làm việc đó.
- Các thành phần kiến trúc của VueX :



- Cài đặt : npm install vuex@next --save

  yarn add vuex@next --save

  yarn add pinia

  npm install pinia

- **State**

  State là nơi lưu giữ data trong từng component. Đối với Angular 2+, nó là các thuộc tính của class (component), đối với React thì nó rõ ràng là các phần trong this.state (Đối với loại Class component) và useState (Đối với function component). Còn riêng đối với VueJS thì nó là các phần trong data.

- Và khi mà chúng ta add VueX vào Vue thì hiểu đơn giản là nó không cục bộ ở từng Component nữa mà sẽ được share cho các component và service khác trong hệ thống.
- **Chú ý** chỉ nên gắn những phần là ***singleton***. Ở đây ta sẽ cần hiểu ***singleton*** nghĩa là ít thay đổi, hoặc không thay đổi trong suốt vòng đời của app.
- **Store**

  Store là phần quản lý State, nó sẽ có các phương thức cho phép thay đổi state một cách gián tiếp thông qua dispatch hoặc một commit. Store là duy nhất bên trong một app và sẽ được khởi tạo cùng với root.

- **Getters**

  Có thể hiểu Getters là một computed dùng để tính toán data, xử lí một logic chung nào đó mà nhiều component dùng. Hàm viết ở đây chỉ có thể dùng để lấy data ra chứ không thể chỉnh sửa (Liên hệ một chút là nó giống như Getter trong OOP vậy – thể hiện tính bao đóng).

- **Actions**

  Trong Action sẽ thường chứa logic liên quan đến nghiệp vụ business, nên hiểu là nó không trực tiếp thay đổi state. Nếu muốn thay đổi State thì cần dùng một *Commit* đã được định nghĩa tại Mutations. Lí do là bởi vì Actions thường được chạy bất đồng bộ (Code vẫn chạy khi mà actions chưa hoàn thành) và như vậy khi nó hoàn thành thì chúng ta mới lên Commit để change data.

- **Mutations**

  Các hàm trong **Mutations** thường sẽ không nên chứa logic hay nghiệp vụ business gì, nó chỉ nên có một việc là update state. Nó chạy đồng bộ và một hàm bên trong một **Mutations** được gọi là một *Commit*.

Tương tự với VueX, hiện nay Pinia đang được sử dụng rộng rãi bởi cộng đồng lập trình viên VueJS.

Link website thi trắc nghiệm online sử dụng VueJS :

http://14.225.205.132:3601/

Link source code (github) :

https://github.com/ngokhang/DoAnCoSo