# **Vue introduction**

## **Là một frameword của JS dùng để phát triển UI người dùng và các ứng dụng web đơn trang (SPA)**

* + Dễ học và sử dụng
  + Mã nguồn mở và cộng đồng lớn:
  + Hiệu suất:
    - Virtual DOM
      1. Khi khởi chạy sẽ tạo một cây Virtual DOM dựa trên cấu trúc và dữ liệu của ứng dụng
      2. Render và Re-render: Khi có sự thay đổi thì VUE render lại => tạo ra một cây Virtual DOM mới dựa trên dữ liệu mơi
      3. So sánh : So sánh khác biệt giữa Virtual DOM mới và cũ => đưa ra sự khác biệt
      4. Tối ưu hóa : Vue chỉ cập nhập các phần tử có thay đổi chứ k cập nhập toàn web => tối ưu hóa hiệu xuất
      5. Áp dụng thay đổi: Cập nhập DOM từ các thay đổi
  + Ứng dụng động (SPA - Single Page Application) :
    - Single Page Applications (SPA):
      1. Là loại ứng dụng web mà khi người dùng tương tác với nó , trang web sẽ k làm mới lại hoàn toàn => Sử dụng hệ thống "Virtual DOM" để chỉ làm mới các phần của trang cần thay đổi.
    - Vue Router:
      1. Xác định các tuyến đường (routes) và liên kết chúng với các thành phần Vue cụ thể .
      2. Khi người dùng điều hướng tới một tuyến khác thì vue sẽ hiện thị nội dung và không cần tải lại trang
    - Cập nhật dữ liệu động:
      1. Cung cấp các khả năng cập nhật dữ liệu động mà bạn có thể tích hợp vào các ứng dụng => hiển thị dữ liệu trả về mà không cần tải lại trang ( dự liệu động là dữ liệu tđ liên tục )

## **SPA & MPA/MVC ( SSR , SSG )**

* SPA (Single Page Application)
  + - * Tải toàn bộ nội dung HTML và JS một lần duy nhất tại **máy khách** khi người dùng truy cập .Sau đó thay vì tải lại trang khi người dùng điều hướng sang trang khác thì SPA sẽ sử dụng JS để tải và hiện thị nội dung
      * Hầu hết đều xử lý ở khía mấy khách ( client )
        + Trải nghiệm người dùng mượt và nhanh chóng và không yêu cầu tải lại trang
      * Khởi đầu ban đầu chậm : vì phải tải nội dung và JS trước khi hiển thị nội dung
        + Sử dụng Router để quản lý các tuyến và điều hướng các trang của ứng dụng SPA DÙNG <router-view />
  + SSR ( Server-Side Rendering)
* Tạo ra HTML hoàn chỉnh cho mỗi trang tại **máy chủ** và gửi nó đến máy khách khi người dùng yêu cầu.
* Máy khách sẽ nhận đc trang web có nội dung ngay lập tức mà k cần chờ JS chạy
* Phù hợp cho trang web có dữ liệu động hoạt cần cập nhập thường xuyên
* Yêu cầu phần lớn xử lý ở máy chủ.
  + - * SSR thích hợp cho các ứng dụng đòi hỏi SEO tốt hơn
      * Thời gian khởi đầu nhanh : SSR có thể có thời gian khởi đầu nhanh hơn vì trang đã được tạo sẵn tại máy chủ ( dễ SEO ) .
      * Prefetching : trước khi trang web được gửi đến trình duyệt, bạn có thể thực hiện prefetching dữ liệu, nghĩa là bạn có thể tải dữ liệu cần thiết từ máy chủ trước khi render trang
* MPA (Multi Page Application):
  + - * \* Mỗi trang web là một tài liệu HTML độc lập và có thể chứa các tệp CSS và JavaScript riêng biệt tại máy chủ
      * Khi người dùng điều hướng sang trang khác thì sẽ tải lại toàn bộ trang
      * Thường có thời gian phản hồi nhanh vì k phải tải tất cả JS cùng lúc
      * Không có trải nghiệm người dung mượt và tương tác động như SPA
        + Sử dụng Router để quản lý các tuyến và điều hướng các trang của ứng dụng MPA KHÔNG DÙNG <router-view />
    - SSG (Static Site Generation):
      * Trang web được tạo và render ra tệp HTML tại thời điểm xây dựng trang web thường xảy ra ở phía **server** .
      * Các tệp HTML này là tĩnh và không thay đổi => phục vụ trực tiếp mà k cần xử lý
      * Tốc độ nhanh vì render tại thời điểm xây dựng
      * SEO tốt vì các trang tĩnh hiển thị đầy đủ trên công cụ
    - MVC (Model-View-Controller):
      * Model
        + Đại diện cho dữ liệu trong ứng dụng . Trong Vue sử dung các đốii tượng VUE để đại diện cho Model ( Thường chứa các thuộc tính dữ liệu và phương thức để thao tác với dữ liệu ).

VD

const count = ref(0)

data() {

*return* {

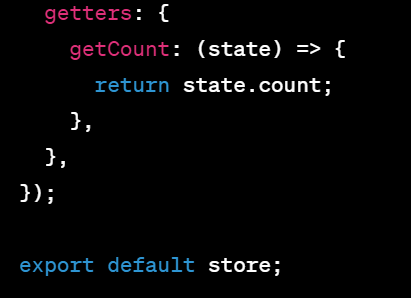
count: 0

}

}

* + - * View
        + Giao diện người dùng
        + Sử dụng các template HTML để tạo giao diện người dùng
        + Thường sử dụng các biểu thức và directives của Vue để hiển thị dữ liệu và tương tác với người dùng
      * Controller
        + Là lớp trung gian giữa Model và View.
        + Trong VUE thường sẽ dùng ViewModel thay vì Controller => để thực hiện các chức năng tương tự một phần của "Controller" trong MVC.
        + Đại diện cho logic và xử lý dữ liệu của ứng dụng đồng thời cũng cung cấp các phương thức và sự kiện để xử lý các tương tác của người dùng.

## **SPA frameworks**

* + - Vue Router:
      * Xác định các tuyến đường (routes) và liên kết chúng với các thành phần Vue cụ thể .
      * Khi người dùng điều hướng tới một tuyến khác thì vue sẽ hiện thị nội dung và không cần tải lại trang
* Vuex ( quản lý trạng thái của ứng dụng )
  + - * 1. Cài đặt Vuex
      * npm install vuex --save
        1. A screen shot of a computer code

           Description automatically generatedTạo store Vuex : chứa trạng thái ứng dụng và định nghĩa các mutations và actions để thay đổi trạng thái đó.

* + - * Mutations : xử lý các hàm đồng bộ

Được gọi thông qua commit.

* + - * Actions : xử lý các hàm bất đồng bộ

Được gọi thông qua dispatch.

A screen shot of a computer code

Description automatically generated

* + - * 1. A screen shot of a computer program

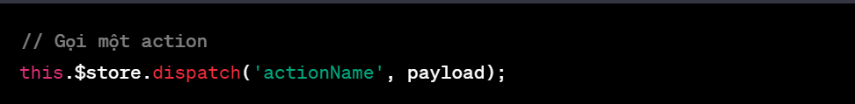
           Description automatically generatedKết nối Vuex với ứng dụng Vue.js
        2. Sử dụng

Gọi Mutations:

A screen shot of a computer

Description automatically generated

Gọi Actions:



* Pinia ( quản lý trạng thái của ứng dụng )
* Lazy Loading
  + - Sử dụng import để nhập các thành phần mà bạn muốn tải lazy:
    - Sử dụng import để nhập các tài nguyên mà bạn muốn tải lazy:

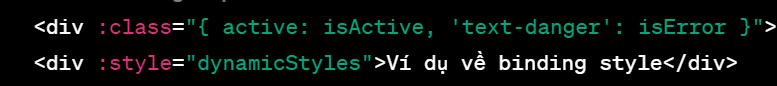
# **Dom interaction**

Dom interaction trong Vue.js đề cập đến cách Vue tương tác với DOM (Document Object Model). Vue cung cấp các cú pháp và cơ chế để thao tác với DOM trong ứng dụng web. Trong vue các hđ tương tác với DOM thường thông qua directive ( v-bind, v-on, v-if, v-for, và nhiều directive khác ).

* Interpolation : Sử dụng cặp dấu ngoặc nhọn {{ }} để hiển thị dữ liệu từ Vue instance trong các phần tử HTML hoặc văn bản.

## **Binding:** cho phép bạn kết nối dữ liệu từ Vue instance đến các phần tử DOM

* A black background with white text

  Description automatically generatedBinding Attributes: “v-bind” hoặc “:” => Sử dụng để gắn dữ liệu vào thuộc tính của một phần tử HTML.
* Binding Class và Style: cách để gắn dữ liệu vào lớp (classes) và kiểu (styles) của một phần tử HTML
* A black background with text

  Description automatically generatedTwo-Way Data Binding: Khả năng đồng bộ hóa dữ liệu giữa một biến trong Vue instance và một phần tử HTML bằng cách sử dụng directive “v-model”.
* Binding text ( v-text) , Binding html (v-html)
  + Ràng buộc một chiều (One-Way Binding) : v-bind, v-text, v-html, v-if, v-show, Interpolation.
  + Ràng buộc hai chiều (Two-Way Binding): v-model

## **Event Handling** (Xử lý sự kiện)

* v-on or @ : Lắng nghe sự kiện DOM và gọi các phương thức khi sự kiện xảy ra
* @click
* @input , @submit , @change
* @keydown, @keyup
* @mouseenter, @mouseleave
* @focus , @blur ,…
* Event Modifiers
  + stop : ngăn chặn lan truyền sự kiện

<div @click="outerClick">

  <div @click.stop="innerClick">Inner Div</div>

</div>

* Kích vào “Inner Div” ko lan truyền tới thằng cha . Nếu k có thì hàm “outerClick” vẫn đc gọi.
  + .prevent : ngăn chặn tải lại trang khi nhấn vào liên kết
  + .capture : xử lý theo cơ chế capture thay vì bubbling ( gọi thằng cha trước )

<div @click.capture="parentClick">

  <div @click="childClick">Child Div</div>

</div>

* Kích vào “childClick” thì sẽ xử lý sự kiện “parentClick” trước xong mới xuống sk “childClick”
  + . self : chỉ xử lý khi click đúng phần tử đó
  + .once : Chỉ kích hoạt 1 lần duy nhất , xong sẽ bị gỡ bỏ
  + .passive : Không cần gọi sự kiện “event.preventDefault()” trong hàm xử lý ( ngăn chặn các hành vi mặc định của trang web . vd : tải lại trang khi dùng thẻ a ) . Thường dùng trong các sự kiện @scroll, @touchmove, @wheel
* Key Modifiers
* .enter: Xử lý sự kiện khi nhấn enter
* .delete hoặc .backspace
* .esc
* .space
* .up, .down, .left, .right
  + - Keydown : nhấn
    - Keyup : nhả
  + System Modifier Keys : Nhấn cùng với các sự kiện khác
    - .ctrl
    - .alt
    - .shift

<input @keyup.alt.enter="clear" />

## A computer screen with white text Description automatically generatedA computer code with white text Description automatically generated**Refs** : sử dụng refs để truy cập các phần tử DOM trực tiếp trong các phương thức Vue.

## A computer screen with white text Description automatically generated**Custom Directives** : tạo chỉ thị tùy chỉnh để thêm logic xử lý vào phần tử DOM.

# Lifecycle Hooks

## Creation

* beforeCreate : Trước khi instance được tạo
* created : Sau khi instance được tạo
  + setup là một thành phần của Vue 3 thay thế cho các lifecycle hooks beforeCreate và created trong Vue 2.

## Mouting

* beforeMount / onBeforeMount : Trước khi kết nối vào DOM
* mounted / onMounted : Sau khi kết nối vào DOM

## Updating

* beforeUpdate / onBeforeUpdate : Trước khi dữ liệu được cập nhật
* update / onUpdate : Sau khi cập nhật dữ liệu và cập nhật DOM

## UnMounting

* beforeUnMount / onBeforeUnMount : Để thực hiện tác vụ tiền xử lý trước khi bị hủy
* UnMount/ onUnMount : Để thực hiện tác vụ tiền xử lý sau khi bị hủy

## effect hooks: sử dụng để theo dõi quá trình render của sự kiện ( **k nằm trong lifecycel** )

* onRenderTracked : khi một phần tử DOM hoặc một thành phần Vue bắt đầu được theo dõi trong quá trình render => ( đc khởi tạo )
* onRenderTriggered : khi một phần tử DOM hoặc một thành phần Vue đã được theo dõi và cần phải được render lại (re-render). => ( khi kích hoạt )

## Theo dõi Vue Route

* onActivated: khi một thành phần hoặc route được kích hoạt => điều hướng đến một route mới mà chứa thành phần này
* onDeactivated: khi một thành phần hoặc route bị vô hiệu hóa => điều hướng từ route chứa thành phần này sang một route khác

## onErrorCaptured :

* xử lý lỗi xảy ra trong các thành phần con (child components) bên trong một thành phần cha (parent component).
* Khi một lỗi xảy ra trong một thành phần con, Vue.js sẽ bắt lỗi này và truyền nó lên đến thành phần cha gần nhất chứa hook onErrorCaptured. Hook này sẽ được kích hoạt và bạn có thể xử lý lỗi ở đây

## onServerPrefetch:

* Sử dụng trong ngữ cảnh Server-Side Rendering (SSR)
* Được gọi khi trang cần được prefetch trước , trước khi trang được render

# **Composition API**

**Là một thành phần của Vue 3 giúp bạn tổ chức logic trong thành phần Vue một cách logic và có cấu trúc hơn**

1. Tái sử dụng code : cho phép bạn viết logic một lần và sử dụng lại nó trong nhiều thành phần khác nhau
2. Tích hợp dễ dàng: để kết hợp các phần nhỏ của logic vào một thành phần lớn hơn một cách dễ dàng
3. Khả năng định nghĩa trạng thái và logic: Bạn có thể định nghĩa trạng thái (state), computed properties, methods và lifecycle hooks bằng cách sử dụng các hàm trong Composition API.
4. Hiệu suất: Composition API có thể giúp tối ưu hóa hiệu suất bằng cách giảm bớt việc tạo và duy trì các thực thể Vue không cần thiết.

## Reactivity : Tự động cập nhập và theo dõi sự thay đổi của dữ liệu trong ứng dụng đồng bộ hóa dữ liệu và hiển thị

## Setup

* Khởi Tạo Dữ Liệu
* Khởi tạo hàm xử lý
* Sử Dụng Hàm Lifecycle Hooks
* Tái Sử Dụng Logic
* Trả về Dữ Liệu cho Template

## ref, isRef, unRef, toRef, toRefs ( API của Refs )

* + Ref object
    - Giá trị ban đầu (Initial Value): Ref object được tạo bằng ref() có giá trị ban đầu là undefined.
    - .value Property: Thuộc tính .value của ref object chứa giá trị reactivity.
  + ref : Tạo một Ref để bọc một giá trị reactivity cơ bản
    - trả về một Ref object chứa giá trị ban đầu là undefined
  + isRef : Kiểm tra xem giá trị có phải là một ref hay ko trả về true , flase
  + unRef : Trích xuất giá trị gốc từ một Ref k phải ref thì trả về chính nó giống ref.value
  + toRef : chuyển đổi một trường từ một đối tượng có khả năng reactivity (như một đối tượng reactive) thành một Ref
    - A computer screen with text on it

      Description automatically generatedCũng có thể chuyển đổi một Ref object thành một Ref object khác như sao chép

chuyển đổi trường name của data thành 1 Ref

Bất cứ t/đ nào trong nameRef đều a/h đến

data và ngược lại

* toRefs : Chuyển đổi một reactive object thành một đối tượng chứa các Ref objects tương ứng với từng thuộc tính.

## reactive

* được sử dụng để tạo một đối tượng reactive có Reactivity
* ref cũng có thể đc tạo đối tượng như khi truy xuất phải .value ( khi tạo đối tượng dùng reactive còn ref để tạo giá trị nguyên thủy )

## computed

* A computer code with white text

  Description automatically generatedtạo một giá trị tính toán dựa trên các thuộc tính hoặc biến khác

## watchEffect

* được sử dụng để theo dõi các biến hoặc giá trị và thực hiện lại mỗi khi nó thay đổi
* A computer code on a black background

  Description automatically generatedtự động phát hiện các biến reactivity (ref objects và reactive objects) mà hàm callback bên trong nó sử dụng

## watch

* Cho phép bạn theo dõi và phản ứng khi các biến reactivity thay đổi
* Khác với watchEffect tự động phát hiện còn watch phải truyền các biến vào
* A screenshot of a computer

  Description automatically generatedTruyền nhiều biến
* Có trả về giá trị cũ và mới
* Có thể hủy watch

A screen shot of a computer

Description automatically generated

## Provide / Inject

Chia sẽ dữ liệu và logic giữa component cha/con ở xa

* Chia sẻ Logic Hoặc Dữ Liệu Qua Nhiều Cấp Độ Component so với props
* Giảm Sự Lặp Lại Code so với props
* Thao Tác Với Dữ Liệu Global
  + Component cha

<script setup>

import { provide } from 'vue';

const sharedData = 'This is shared data from the parent';

provide('sharedData', sharedData);

</script>

* Provide(“key”,value)
  + Component con

<script setup>

import { inject } from 'vue';

// Inject dữ liệu từ component cha

const sharedData = inject('sharedData');

</script>

* ịnect(“key”)

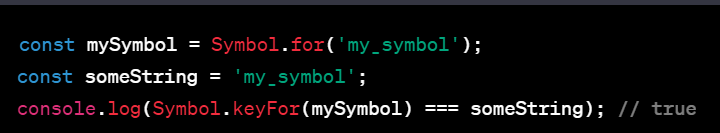
## Symbol

Là một kiểu dữ liệu nguyên thủy (primitive data type) dùng để đại diện cho một giá trị duy nhất không thể thay đổi và được sử dụng để định danh một cách độc nhất trong một ngữ cảnh cụ thể.

A black background with red and green text

Description automatically generated

Lưu ý khi dùng Symbol

* Mô Tả Rõ Ràng
* Không Sử Dụng Cùng Mô Tả Cho Nhiều Symbol
  + Cho dù 2 Symbol cùng mô tả cũng ko bh bằng nhau
* Sử Dụng Symbol.for Nếu Cần So Sánh Với Chuỗi
* Khi Truy Cập Symbol
  + A black background with white text

    Description automatically generatedhãy sử dụng cùng tên Symbol đã được định nghĩa.