API Documentation#

Enumerations#

NavigationFailureType

Interfaces#

HistoryState

NavigationFailure

NavigationGuard

NavigationGuardNext

NavigationGuardWithThis

NavigationHookAfter

RouteLocation

RouteLocationMatched

RouteLocationNormalized

RouteLocationNormalizedLoaded

RouteLocationOptions

RouteMeta

RouteRecordNormalized

Router

RouterHistory

RouterLinkProps

RouterOptions

RouterScrollBehavior

RouterViewProps

Type Aliases#

LocationQuery#

Ƭ LocationQuery: Record<string, LocationQueryValue | LocationQueryValue[]>

Normalized query object that appears in RouteLocationNormalized

LocationQueryRaw#

Ƭ LocationQueryRaw: Record<string | number, LocationQueryValueRaw | LocationQueryValueRaw[]>

Loose LocationQuery object that can be passed to functions like push and replace or anywhere when creating a RouteLocationRaw

PathParserOptions#

Ƭ PathParserOptions: Pick<\_PathParserOptions, "end" | "sensitive" | "strict">

RouteComponent#

Ƭ RouteComponent: Component | DefineComponent

Allowed Component in RouteLocationMatched

RouteLocationRaw#

Ƭ RouteLocationRaw: string | RouteLocationPathRaw | RouteLocationNamedRaw

User-level route location

RouteParams#

Ƭ RouteParams: Record<string, RouteParamValue | RouteParamValue[]>

RouteParamsRaw#

Ƭ RouteParamsRaw: Record<string, RouteParamValueRaw | Exclude<RouteParamValueRaw, null | undefined>[]>

RouteRecord#

Ƭ RouteRecord: RouteRecordNormalized

Normalized version of a route record.

RouteRecordName#

Ƭ RouteRecordName: string | symbol

Possible values for a user-defined route record's name

RouteRecordRaw#

Ƭ RouteRecordRaw: RouteRecordSingleView | RouteRecordSingleViewWithChildren | RouteRecordMultipleViews | RouteRecordMultipleViewsWithChildren | RouteRecordRedirect

UseLinkOptions#

Ƭ UseLinkOptions: VueUseOptions<RouterLinkOptions>

Variables#

RouterLink#

• Const RouterLink: \_RouterLinkI

Component to render a link that triggers a navigation on click.

RouterView#

• Const RouterView: () => { $props: AllowedComponentProps & ComponentCustomProps & VNodeProps & RouterViewProps ; $slots: { default: (arg: { Component: VNode<RendererNode, RendererElement, { [key: string]: any; }> ; route: RouteLocationNormalizedLoaded }) => VNode<RendererNode, RendererElement, { [key: string]: any; }>[] } }

Type declaration#

• new RouterView()

Component to display the current route the user is at.

START\_LOCATION#

• Const START\_LOCATION: RouteLocationNormalizedLoaded

Initial route location where the router is. Can be used in navigation guards to differentiate the initial navigation.

Example

import { START\_LOCATION } from 'vue-router'

router.beforeEach((to, from) => {

if (from === START\_LOCATION) {

// initial navigation

}

})

Functions#

createMemoryHistory#

▸ createMemoryHistory(base?): RouterHistory

Creates an in-memory based history. The main purpose of this history is to handle SSR. It starts in a special location that is nowhere. It's up to the user to replace that location with the starter location by either calling router.push or router.replace.

Parameters#

Name Type Default value Description

base string '' Base applied to all urls, defaults to '/'

Returns#

RouterHistory

a history object that can be passed to the router constructor

createRouter#

▸ createRouter(options): Router

Creates a Router instance that can be used by a Vue app.

Parameters#

Name Type Description

options RouterOptions RouterOptions

Returns#

Router

createWebHashHistory#

▸ createWebHashHistory(base?): RouterHistory

Creates a hash history. Useful for web applications with no host (e.g. file://) or when configuring a server to handle any URL is not possible.

Example

// at https://example.com/folder

createWebHashHistory() // gives a url of `https://example.com/folder#`

createWebHashHistory('/folder/') // gives a url of `https://example.com/folder/#`

// if the `#` is provided in the base, it won't be added by `createWebHashHistory`

createWebHashHistory('/folder/#/app/') // gives a url of `https://example.com/folder/#/app/`

// you should avoid doing this because it changes the original url and breaks copying urls

createWebHashHistory('/other-folder/') // gives a url of `https://example.com/other-folder/#`

// at file:///usr/etc/folder/index.html

// for locations with no `host`, the base is ignored

createWebHashHistory('/iAmIgnored') // gives a url of `file:///usr/etc/folder/index.html#`

Parameters#

Name Type Description

base? string optional base to provide. Defaults to location.pathname + location.search If there is a <base> tag in the head, its value will be ignored in favor of this parameter but note it affects all the history.pushState() calls, meaning that if you use a <base> tag, it's href value has to match this parameter (ignoring anything after the #).

Returns#

RouterHistory

createWebHistory#

▸ createWebHistory(base?): RouterHistory

Creates an HTML5 history. Most common history for single page applications.

Parameters#

Name Type

base? string

Returns#

RouterHistory

isNavigationFailure#

▸ isNavigationFailure(error, type?): error is NavigationRedirectError

Check if an object is a NavigationFailure.

Example

import { isNavigationFailure, NavigationFailureType } from 'vue-router'

router.afterEach((to, from, failure) => {

// Any kind of navigation failure

if (isNavigationFailure(failure)) {

// ...

}

// Only duplicated navigations

if (isNavigationFailure(failure, NavigationFailureType.duplicated)) {

// ...

}

// Aborted or canceled navigations

if (isNavigationFailure(failure, NavigationFailureType.aborted | NavigationFailureType.canceled)) {

// ...

}

})

Parameters#

Name Type Description

error any possible NavigationFailure

type? NAVIGATION\_GUARD\_REDIRECT optional types to check for

Returns#

error is NavigationRedirectError

▸ isNavigationFailure(error, type?): error is NavigationFailure

Parameters#

Name Type

error any

type? ErrorTypes | NavigationFailureType

Returns#

error is NavigationFailure

loadRouteLocation#

▸ loadRouteLocation(route): Promise<RouteLocationNormalizedLoaded>

Ensures a route is loaded, so it can be passed as o prop to <RouterView>.

Parameters#

Name Type Description

route RouteLocationNormalized resolved route to load

Returns#

Promise<RouteLocationNormalizedLoaded>

onBeforeRouteLeave#

▸ onBeforeRouteLeave(leaveGuard): void

Add a navigation guard that triggers whenever the component for the current location is about to be left. Similar to beforeRouteLeave but can be used in any component. The guard is removed when the component is unmounted.

Parameters#

Name Type Description

leaveGuard NavigationGuard NavigationGuard

Returns#

void

onBeforeRouteUpdate#

▸ onBeforeRouteUpdate(updateGuard): void

Add a navigation guard that triggers whenever the current location is about to be updated. Similar to beforeRouteUpdate but can be used in any component. The guard is removed when the component is unmounted.

Parameters#

Name Type Description

updateGuard NavigationGuard NavigationGuard

Returns#

void

useLink#

▸ useLink(props): Object

Parameters#

Name Type

props VueUseOptions<RouterLinkOptions>

Returns#

Object

Name Type

href ComputedRef<string>

isActive ComputedRef<boolean>

isExactActive ComputedRef<boolean>

navigate (e: MouseEvent) => Promise<void | NavigationFailure>

route ComputedRef<RouteLocation & { href: string }>

useRoute#

▸ useRoute(): RouteLocationNormalizedLoaded

Returns the current route location. Equivalent to using $route inside templates.

Returns#

RouteLocationNormalizedLoaded

useRouter#

▸ useRouter(): Router

Returns the router instance. Equivalent to using $router inside templates.

Returns#

Router