Chapter 8 – Vue Router

Asst.Prof. Dr. Umaporn Supasitthimethee
ผศ.ดร.อุมาพร สุภสิทธิเมธี

Vue-Router

- What and Why
- Setting Up the Router and Route Config
- Catch All Not found Routes
- `UseRoute` and `UseRouter` Functions
- Styling Active Links

What and Why

- Vue is powerful for creating Single Page Applications: highly interactive webpages that don’t refresh when you change from page to page.
- Single page application shares the same URL for all area of the page
- If your website has multiple pages (or “views”), you would like to share to another user, your always end up with the starting page and cannot share anything else than the starting page of this application
- because the change that happens when clicking this button is driven by JavaScript and has nothing to do with the URL.

https://www.vuemastery.com/blog/vue-router-a-tutorial-for-vue-3/

Setting Up the Router and Route Config

1. Install Vue Router to your project

Go to your project, install router (need version 4 for Vue 3)

```
> npm install vue-router@4
```

```sh
npm install vue-router@4
```

https://router.vuejs.org/

2. Add a Routing Directory & Configuration File

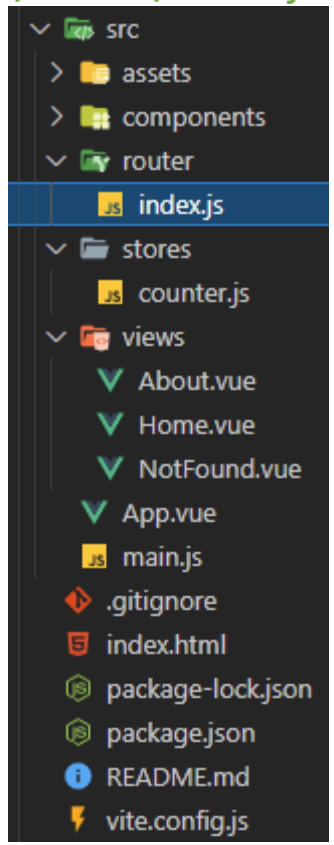
2.1 Importing `createRouter` and `createWebHistory` from the `vue-router` library.

2.2 Creating `routes` in an array, where we specify for each route a few important items:

- `path` - the URL path where this route can be found.
- `name` - An optional name to use when we link to this route.
- `component` - Which component to load when this route is called.

src/router/index.js

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../views/Home.vue'
import About from '../views/About.vue'
const history = createWebHistory()
const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/about',
    name: 'About',
    component: About
  }
]
const router = createRouter({ history, routes })
export default router
```



https://next.router.vuejs.org/installation.html

3. Importing our Routes & using Vue Router

//main.js

```
import { createApp } from 'vue'
import router from './router'
import App from './App.vue'

const app = createApp(App)

app.use(router)

app.mount('#app')
```

4. Using <router-link> to navigate inside the page with routes

to – path you want to go as one of registered in the routes

src/App.vue

```
<template>
  <div id="nav">

    <router-link to="/">Home</router-link> |
    <router-link :to="{name:'About'}">About Us</router-link>

  </div>
  <router-view/>
</template>
```

You can pass text or any HTML Tags

src/router/index.js

```
const routes = [
 //routes registered
 {
   path: '/',
   name: 'Home',
   component: Home
 },

 {
   path: '/about',
   name: 'About',
   component: About
 }
]
```

router-link will not load a different page and reload the entire app and hence loose the current state but instead router-link prevents the browser default loading and the Vue router will take over and download the link that user clicked and updates the URL.

5. Using <router-view> to display the component that corresponds to the URL. This is the place where selected component should be loaded

App.vue

```
<template>
  <div id="nav">
    <router-link to="/">Home</router-link> |
    <router-link to="/about">About</router-link>
  </div>
  <router-view> </router-view>
</template>
```

src/router/index.js

```
const routes = [
  //routes registered
{
   path: '/',
   name: 'Home',
   component: Home
 },
 {
   path: '/about',
   name: 'About',
    component: About
  }
]
```

Redirect

- Redirecting is also done in the routes configuration. To redirect from `/home` to `/`:

```
const routes = [{ path: '/home', redirect: '/' }]
```

- The redirect can also be targeting a named route:

```
const routes = [{ path: '/home', redirect: { name: 'homepage' } }]
```

https://router.vuejs.org/guide/essentials/redirect-and-alias.html#redirect-and-alias

Catch All Not found Routes

Catch All Not found Routes

src/router/index.js

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../views/Home.vue'
import About from '../views/About.vue'
import NotFound from '../views/NotFound.vue'
const history = createWebHistory()
const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/about',
    name: 'About',
    component: About
  },

  {
    path: '/:catchNotMatchPath(.*)',
    name: 'NotFound',
    component: NotFound
  }
]
const router = createRouter({ history, routes })
export default router
```

any name allowed

(.*) Regular expression- any character combination

NotFound.vue

```
<script setup></script>

<template>
  <h2>Page Not Found!</h2>
</template>

<style></style>
```

Regular params (denoted by a colon :) will only match characters in between url fragments, separated by / . If we want to match **anything**, we can use a custom *param* regexp by adding the regexp inside parentheses right after the *param*:

In this example, `catchNotMatchPath` is the dynamic segment and (.*) is a regular expression, which essentially matches against anything. So if none of the routes above this one match, it will use this one.

`UseRoute` and `UseRouter` Functions

`useRoute` : Accessing the current Route inside setup

We use `useRoute` function inside of `setup()` to return the current route location. (it is equivalent to using `$route` inside templates.)

We can use params in `useRoute()` to accept parameter on router path

```
<script setup>
import { useRoute } from 'vue-router'
const { params } = useRoute()
console.log(params.noteId)
</script>
```

Using `$route` inside templates

```
<template>
  <h1>The user is {{ $route.params.noteId }}</h1>
</template>
```

`useRouter`: Accessing the Router inside setup

- We use the `useRouter` function in `<script>` to return the router instance (it is equivalent to using `$router` inside templates.)
- To navigate to a different URL, use `router.push`. This method pushes a new entry into the history stack, so when the user clicks the browser back button they will be taken to the previous URL.
- This is the method called internally when you click a `<router-link>`, so clicking `<router-link :to="...">` is the equivalent of calling `router.push(...)`

Declarative	Programmatic
`<router-link :to="...">`	`router.push(...)`

https://router.vuejs.org/guide/essentials/navigation.html

useRouter

```
const router = createRouter({
  history: createWebHistory(),
  routes: [
    {
      path: '/users/:username',
      name: 'user',
      component: User
    }
  ]
})
```

All params must be declared in the path of the route record and can only be strings or array of strings for repeatable params.

Convert to JSON String and then Parse back to Javascript object with JSON.parse()

```
import {useRouter} from 'vue-router'
const router = useRouter()

// literal string path
router.push('/users/eduardo')

const username = 'eduardo'
// we can manually build the url but we will have to handle the encoding ourselves
router.push(`/users/${username}`) // -> /user/eduardo

// object with path
router.push({ path: '/users/eduardo' }

// named route with params to let the router build the url route
router.push({ name: 'user', params: { username: 'eduardo' } })
```

Named Routes

In the router, we specify in our route with the : denoting the variable.

```
const routes = [
  {
    path: '/user/:username',
    name: 'user',
    component: User
  }
]
```

Named Routes can be useful if our URL paths change in the future. We wouldn't have to change all the router-links if the path changes, since they're referencing using name.

In both cases, the router will navigate to the path /user/erina

1. To link to a named route, you can pass an object to the router-link component's to prop:

```
<router-link :to="{ name: 'user', params: { username: 'erina' }}">
  User
</router-link>
```

2. This is the exact same object used programmatically with `router.push()`

```
router.push({ name: 'user', params: { username: 'erina' } })
```

Styling Active Links

Styling Active Links

- By default, Vue Router adds two active classes i.e. "`router-link-active`" and "`router-link-exact-active`" to the links that we create using the "`router-link`" component.
- `router-link-active` class applied to any navigation item which contains a part of the currently active route. Note the default value can also be configured globally via the `linkActiveClass` router constructor option.
- `router-link-exact-active` class applied when the page URL matches exactly with the path we set in the `<router-link>` component. Note the default value can also be configured globally via the `linkExactActiveClass` router constructor option.

For example, If you set `/hello` to the router-link as the path `<router-link to="/hello">`, the **router-link-active** class will be applied when the current page URL starts with "`/hello`". If you visit two pages with the URLs `/hello` and `/hello/someone`, in both cases that link will contain the **router-link-active** class because both URLs start with `/hello`.

For example, if we create two links `<router-link to="/hello">` and `<router-link to="/hello/someone">`, the **router-link-exact-active** class will be applied to `<router-link to="/hello/someone">` only when the path is `/hello/someone`.

router constructor option

```
linkExactActiveClass: 'text-blue-500',
linkActiveClass: 'text-green-500'
```

```
import { createRouter, createWebHistory } from 'vue-router'
import QuestionsView from '../views/QuestionsView.vue'
import NotFound from '../views/NotFoundView.vue'
import OneQuestionManagement from '../components/OneQuestionManagement.vue'
import AboutView from '../views/AboutView.vue'
const router = createRouter({
  history: createWebHistory(),
  routes: [
    {
      path: '/',
      redirect: '/questions'
    },
    {
      path: '/questions',
      name: 'QuestionsView',
      component: QuestionsView
    },
    {
      path: '/question/:id',
      name: 'question',
      component: OneQuestionManagement
    },
    {
      path: '/aboutus',
      name: 'About',
      component: AboutView
    },
    {
      path: '/:catchNotMathPath(.*)',
      name: 'NotFound',
      component: NotFound
    }
  ],

  linkExactActiveClass: 'text-blue-500',
  linkActiveClass: 'text-green-500'
})
```

https://v3.router.vuejs.org/api/#linkactiveclass

TailwindCSS Class

```
<router-link active-class="text-purple-500" :to="{ name: 'Home' }"
  >Home |
</router-link>
<router-link exact-active-class="text-purple-500" :to="{ name: 'About' }"
  >About
</router-link>
```

Navigation Guards

https://router.vuejs.org/guide/advanced/navigation-guards.html#Navigation-Guards

Navigation Guards

- As the name suggests, the navigation guards provided by Vue router are primarily used to guard navigations either by redirecting it or canceling it.
- You can register global before guards using `router.beforeEach`:

```
router.beforeEach(async (to, from) => {
  // canUserAccess() returns `true` or `false`
  const canAccess = await canUserAccess(to)
  if (!canAccess) return '/login'
})
```

- Global before guards are called in creation order, whenever a navigation is triggered. Guards may be resolved asynchronously, and the navigation is considered pending before all hooks have been resolved.
- Every guard function receives two arguments:
  - `to`: the target route location being navigated to.
  - `from`: the current route location being navigated away from.