Vue.js router

WASA: Web and Software Architecture

Enrico Bassetti

SPA and Vue Router

In a Single-page application, views/pages are rendered **on the client**, leveraging on JavaScript and the browser itself.

The server provides the code for the SPA (JS+HTML+CSS), and APIs. The SPA will send and receive data using APIs using JavaScript.

Why a router?

Why a router? To "navigate" the application.

Views have a **unique URL**. Optionally, you can pass URL or query parameters.

How?

The router has the following parts:

- Routes (binding between paths and components)
- <RouterView />: the router view inside the Vue main component
  - it will be replaced with the component indicated by the URL when it changes
- <RouterLink to="/link1">Link</RouterLink> creates a link
  - When clicked, the router view will be replaced by the component registered for /link1

Stacked views

Router views are stacked in the history. You are pushing into the history when you use <RouterLink>.

You can control the stack programmatically with these actions:

- **Push**: new view added on top of the stack, and switch to it
- **Go**: move back/forward in the history
- **Replace**: like push, except that discard the current page from the history

Users can go back/forward using buttons in the browser.

Router JavaScript

```
import {createRouter, createWebHashHistory} from 'vue-router'
import HomeView from '../views/HomeView.vue'
import Page1View from '../views/Page1View.vue'
import Page2View from '../views/Page2View.vue'

const router = createRouter({
  history: createWebHashHistory(),
  routes: [
    {path: '/', component: HomeView},
    {path: '/link1', component: Page1View},
    {path: '/some/:id/link', component: Page2View},
  ]
});
export default router
```

Main component

```
<script setup>
import { RouterView } from 'vue-router'
</script>
<script>
export default {}
</script>

<template>
  <header>App header</header>
  <main><RouterView /></main>
</template>

<style>
</style>
```

Switch page: router link

```
<template>
  <header>App header</header>
  <main>
    <p>
      <RouterLink to="/">Home</RouterLink>
      <RouterLink to="/link1">Page 1</RouterLink>
      <RouterLink to="/some/1/link">Internal page with
                                    parameter</RouterLink>
      <RouterLink :to="variableWithLinkDest">
                                  Dynamic link</RouterLink>
    </p>
    <RouterView />
  </main>
</template>
```

Switch page from JavaScript

```
<script>
export default {
  methods: {
    doSomething() {
      // ... set the id variable ...
      this.$router.push('/some/' + id + '/link');
    }
  }
}
</script>
<template>
  <button @click="doSomething">
    Do something and change page</button>
  <button @click="$router.push('/');">Back to index</button>
</template>
```

Get route parameters

Given the route /some/:id/link, its component can access the parameter using $route.params:

```
<script>
export default {
  methods: {
    doSomething() {
      console.log(this.$route.params.id);
    }
  }
}
</script>
<template>
  {{ $route.params.id }}
</template>
```

Get route parameters

There is one caveat: switching to the same link with different values (e.g., /some/1/link -> /some/2/link) **won't reload the component**.

You can watch for that event:

```
export default {
  created() {
    this.$watch(
      () => this.$route.params,
      (toParams, previousParams) => {
        // react to route changes...
      }
    )
  }
}
```