Vue.js Стартовий

Маршрутизація у Vue.js

Vue.js Стартовий

Introduction

Кінаш Станіслав

Front-end dev

stanislav.kinash

stanislav.kinash

MCID: 9210561

Тема уроку

Маршрутизація у Vue.js

План уроку

1. Як працює маршрутизація в SPA.
2. Встановлення та налаштування vue-router.
3. Методи vue-router.
4. Приклад роботи роутингу vue-router.

Як працює маршрутизація в SPA

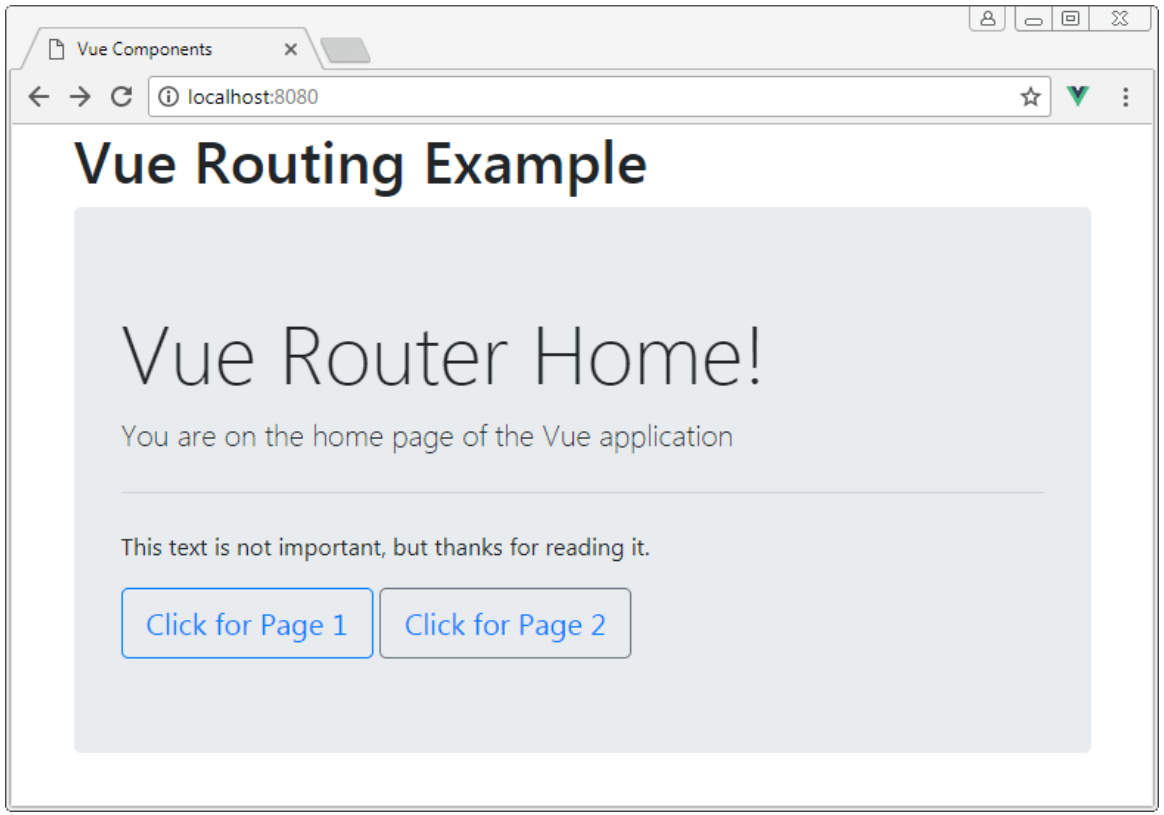
Маршрутизація — це процес зіставлення URL-адрес із різними представленнями чи компонентами у вашій програмі, що дозволяє користувачам переходити між різними сторінками чи розділами вашої програми без необхідності повного перезавантаження сторінки.

Як працює маршрутизація в SPA

У **Vue 3** маршрутизація на стороні клієнта обробляється бібліотекою **vue-router.**

Він надає спосіб зіставляти URL-шляхи з різними компонентами у вашій програмі Vue, дозволяючи вам створювати односторінкові програми (SPA) із кількома представленнями.

Як працює маршрутизація в SPA



Як працює маршрутизація в SPA

Server-Side Routing

Маршрутизація на стороні сервера означає, що сервер надсилає відповідь на основі URL-шляху, який відвідує користувач. Коли ми клацаємо посилання в традиційній веб-програмі, що відтворюється на сервері, браузер отримує відповідь HTML від сервера та перезавантажує всю сторінку з новим HTML.

Client-Side Routing

Проте в односторінковій програмі (SPA) клієнтський JavaScript може перехоплювати навігацію, динамічно отримувати нові дані та оновлювати поточну сторінку без повного перезавантаження сторінки. Це зазвичай призводить до більш швидкої взаємодії з користувачем, особливо для випадків використання, які більше схожі на справжні «додатки», де користувач повинен виконувати багато взаємодій протягом тривалого періоду часу.

Встановлення та налаштування vue-router.

По-перше, вам потрібно буде встановити **vue-router** за допомогою обраного менеджера пакетів:

```
npm install vue-router
```

Далі вам потрібно буде створити новий екземпляр маршрутизатора у вашому файлі main.js, який є точкою входу для вашої програми Vue 3:

```
import { createApp } from 'vue'
import { createRouter, createWebHistory } from 'vue-router'
import App from './App.vue'
import Home from './views/Home.vue'
import About from './views/About.vue'

const router = createRouter({
history: createWebHistory(),
routes: [
    { path: '/', component: Home },
    { path: '/about', component: About }
]
})

const app = createApp(App)
app.use(router)
app.mount('#app')
```

Встановлення та налаштування vue-router.

У цьому прикладі ми створюємо новий екземпляр маршрутизатора за допомогою функції `createRouter` із `vue-router`. Ми вказуємо режим історії як `createWebHistory()`, який використовує API історії HTML5 для маніпулювання URL-адресою в адресному рядку.

Ми також визначаємо два маршрути: один для домашньої сторінки (`/`) і один для сторінки about (`/about`). Кожен маршрут відображається на іншому компоненті, який ми імпортували.

Нарешті, ми використовуємо метод `app.use`, щоб додати екземпляр маршрутизатора до нашої програми Vue, а потім монтуємо програму до елемента #app.

```
import { createApp } from 'vue'
import { createRouter, createWebHistory } from 'vue-router'
import App from './App.vue'
import Home from './views/Home.vue'
import About from './views/About.vue'

const router = createRouter({
history: createWebHistory(),
routes: [
    { path: '/', component: Home },
    { path: '/about', component: About }
]
})

const app = createApp(App)
app.use(router)
app.mount('#app')
```

Встановлення та налаштування vue-router.

Тепер, коли ми налаштували наш маршрутизатор, ми можемо використовувати компонент `<router-link>` для створення зв'язків між нашими різними представленнями:

```
<template>
    <div>
        <h1>My Vue App</h1>
        <router-link to="/">Home</router-link>
        <router-link to="/about">About</router-link>
        <router-view></router-view>
    </div>
</template>
```

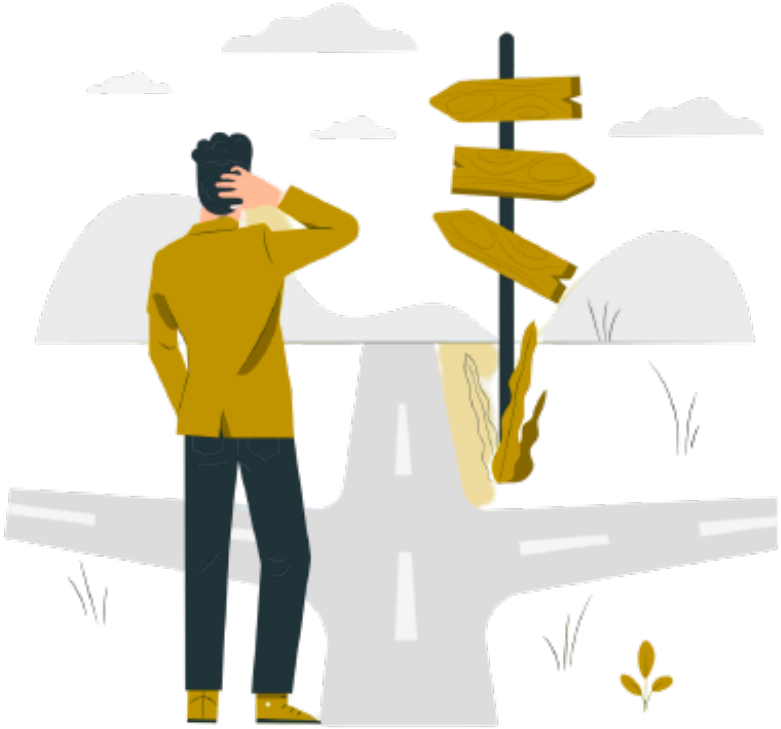
У цьому прикладі ми використовуємо компонент `<router-link>` для створення посилань на домашню сторінку та сторінку `About`.
Ми також використовуємо компонент `<router-view>` для відтворення відповідного компонента на основі поточного маршруту.

Коли користувач натискає посилання, створене `<router-link>`, URL-адреса в адресному рядку буде оновлена, щоб відобразити новий маршрут, і відповідний компонент буде відображено.

Встановлення та налаштування vue-router.

На додачу до відображення URL-шляхів до компонентів, vue-router також підтримує:

- параметри маршруту
- вкладені маршрути
- захист маршрутів, що дозволяють контролювати доступ до різних частин вашої програми на основі автентифікації користувача чи інших факторів.



Методи vue-router

Бібліотека **vue-router** надає кілька методів, які дозволяють вам переходити між різними маршрутами у вашій програмі.

Ось деякі з найбільш часто використовуваних методів:

router.push(location, onComplete?, onAbort?): надсилає новий маршрут до стеку історії та здійснює навігацію до нього. Параметром розташування може бути рядок, що представляє шлях нового маршруту, або об'єкт із такими властивостями, як ім'я, параметри та запит.
Ви також можете надати необов'язкові зворотні виклики `onComplete` і `onAbort`, які будуть викликані, коли навігація буде успішною або невдалою відповідно.

```
// Navigates to the /about page
router.push('/about')

// Navigates to the route with name 'user' and params { id: 123 }
router.push({ name: 'user', params: { id: 123 } })
```

Методи vue-router

Бібліотека **vue-router** надає кілька методів, які дозволяють вам переходити між різними маршрутами у вашій програмі.

router.replace(location, onComplete?, onAbort?): замінює поточний маршрут у стеку історії новим маршрутом і здійснює навігацію до нього. Параметри такі ж, як і для `router.push`.

Ось деякі з найбільш часто використовуваних методів:

```
// Replaces the current route with the /about page
router.replace('/about')
```

Методи vue-router

Бібліотека **vue-router** надає кілька методів, які дозволяють вам переходити між різними маршрутами у вашій програмі.

Ось деякі з найбільш часто використовуваних методів:

router.go(n):
переходить до маршруту в стеку історії відносно поточного маршруту. Параметр n може бути додатним або від'ємним цілим числом, що представляє кількість кроків, які потрібно пройти вперед або назад в історії.

```
// Navigates one step back in the history
router.go(-1)
```

Методи vue-router

Бібліотека **vue-router** надає кілька методів, які дозволяють вам переходити між різними маршрутами у вашій програмі.

Ось деякі з найбільш часто використовуваних методів:

router.back():

еквівалент `router.go(-1).` Переміщується на крок назад в історії.

```
// Navigates one step back in the history
    router.back()
```

Методи vue-router

Бібліотека **vue-router** надає кілька методів, які дозволяють вам переходити між різними маршрутами у вашій програмі.

Ось деякі з найбільш часто використовуваних методів:

router.forward():

еквівалент `router.go(1).` Переходить на крок вперед в історії.

```
// Navigates one step forward in the history
router.forward()
```

Приклад роботи роутингу vue-router.

Наведемо приклад того, як можна використовувати Vue Router для реалізації маршрутизації на стороні клієнта в програмі TODO.

Спочатку створіть нову програму Vue 3 за допомогою Vue CLI:

```
vue create todo-app
```

Далі встановіть бібліотеку vue-router за допомогою обраного менеджера пакетів, використовуючи npm, потрібно виконати таку команду:

```
npm install vue-router
```

Приклад роботи роутингу vue-router.

У каталозі `src` створіть новий каталог під назвою `views`. У каталозі представлень створіть два нових компоненти Vue під назвою `Home.vue` та `About.vue`. Наприклад:

```
<!-- Home.vue -->
<template>
    <div>
      <h1>Home</h1>
      <ul>
        <li v-for="task in tasks" :key="task.id">
          {{ task.title }}
        </li>
      </ul>
    </div>
  </template>

  <script>
  export default {
    data() {
      return {
        tasks: [
          { id: 1, title: 'Do laundry' },
          { id: 2, title: 'Buy groceries' },
          { id: 3, title: 'Walk the dog' },
        ]
      }
    }
  }
  </script>
```

```
<!-- About.vue -->
<template>
    <div>
        <h1>About</h1>
        <p>This is a simple
            TODO app built with
            Vue 3 and Vue Router.
        </p>
    </div>
</template>

<script>
    export default {}
</script>
```

Приклад роботи роутингу vue-router.

У каталозі `src` створіть новий файл під назвою `router.js`. У цей файл імпортуйте функції `createRouter` і `createWebHistory` з бібліотеки `vue-router`, а також компоненти `Home` і `About`. Потім створіть новий екземпляр класу `VueRouter` і налаштуйте його за допомогою масиву маршрутів:

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from './views/Home.vue'
import About from './views/About.vue'

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

const router = createRouter({
  history: createWebHistory(),
  routes
})

export default router
```

Приклад роботи роутингу vue-router.

У файл `src/main.js` імпортуйте об'єкт маршрутизатора та скористайтеся методом `app.use`, щоб додати його до програми Vue:

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

const app = createApp(App)
app.use(router)
app.mount('#app')
```

Приклад роботи роутингу vue-router.

У компоненті `App.vue` використовуйте компонент `router-link` для створення зв'язків між різними видами інтерфейсу. Наприклад:

Це воно! Тепер, коли ви запустите `npm run serve` і перейдете до http://localhost:8080, ви побачите програму TODO з посиланнями на вікна Home і About. Коли ви натискаєте на посилання, відповідні компоненти мають відображатися без необхідності повного перезавантаження сторінки.

```
<template>
  <div>
    <h1>TODO App</h1>
    <nav>
      <router-link to="/">Home</router-link>
      <router-link to="/about">About</router-link>
    </nav>
    <router-view></router-view>
  </div>
</template>

<script>
export default {}
</script>
```

Інформаційний відеосервіс для розробників програмного забезпечення

TestProvider Pearson CyberBionic systematics Microsoft Partner Windows Azure EBA

Перевірка знань

TestProvider.com

TestProvider – це online-сервіс перевірки знань з інформаційних технологій. За його допомогою и можете оцінити свій рівень та виявити слабкі місця. Він буде корисним як у процесі вивчення технології, так і для загальної оцінки знань IT-спеціаліста.

Перевірте, як ви засвоїли даний матеріал на TestProvider.com

Успішне проходження фінального тестування дозволить отримати відповідний Сертифікат.

Дякую за увагу! До нових зустрічей!

Кінаш Станіслав
Front-end dev

MCID: 9210561