VueRouter. Advanced

Динамічне завантаження компонентів (lazy loading)

Компонент завантажується тільки за потреби (коли переходимо за маршрутом)

Загальна форма	Приклад

`{`
`  path:'/шлях',`
`  component: () => import('шлях_до_компонента'),`
`}`
	`//`
	`{`
	`  path:'/about',`
	`  component: () => import('../views/AboutView.vue')`
	`}`

`//--- задання назви файлу(у коментарі) ---`
`{`
`  path:'/шлях',`
`  component: () => import(`
`  /* webpackChunkName: “назва” */`
`  'шлях_до_компонента'`
`  ),`
`}`
	`//--- задання назви файлу(у коментарі) ---`
	`{`
	`  path:'/шлях',`
	`  component: () => import(`
	`  /* webpackChunkName: "about" */`
	`  '../views/AboutView.vue'`
	`  ),`
	`}`

Динамічне завантаження компонентів (lazy loading)

Компонент завантажується тільки за потреби (коли переходимо за маршрутом)

```
{
    path:'/шлях',
    component: () => import('шлях_до_компонента'),
}
```

```
{
    path:'/шлях',
    component: () => import(
        /* webpackChunkName: “назва” */
        'шлях_до_компонента'
    ),
}
```

```
const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },
    {
        path: '/books/:id',
        name: 'books',
        component: () => import('../views/BooksView.vue'),
    },
    {
        path: '/authors/:id',
        name: 'authors',
        component: () => import(/* webpackChunkName: "authors" */
        '../views/AuthorsView.vue'),        You, now • Uncommitted
    },
]
```

Завантажується завжди

Завантажується тільки за потреби (коли переходимо за маршрутом)

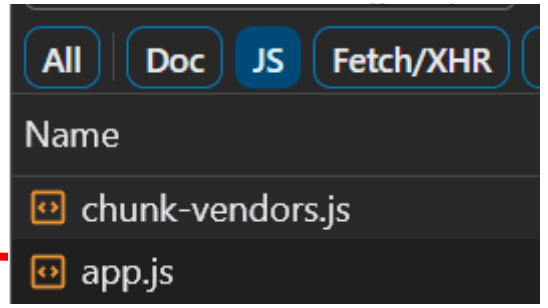
Динамічне завантаження компонентів (lazy loading)

Компонент завантажується тільки за потреби (коли переходимо за маршрутом)

```
{
    path:'/шлях',
    component: () => import('шлях_до_компонента'),
}
```

```
{
    path:'/шлях',
    component: () => import(
        /* webpackChunkName: "about" */
        'шлях_до_компонента'
    ),
}
```

```
const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },
    {
        path: '/books/:id',
        name: 'books',
        component: () => import('../views/BooksView.vue'),
    },
    {
        path: '/authors/:id',
        name: 'authors',
        component: () => import(/* webpackChunkName: "authors" */
        '../views/AuthorsView.vue'),        You, now • Uncommitted
    },
]
```



Завантажується завжди

src_views_BooksView_vue.js

Завантажується тільки за потреби (коли переходимо за маршрутом)

authors.js

Дочірні маршрути

index.js router M

src > router > JS index.js > routes

You, 2 minutes ago | 1 author (You)

```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'
import LibraryView from '../views/LibraryView.vue'
import BooksView from '../views/BooksView.vue'
import AuthorsView from '../views/AuthorsView.vue'

const routes = [
    {
        path: '/',        You, 3 minutes ago • Uncommitted ch
        name: 'home',
        component: HomeView,
    },
    {
        path: '/library',
        name: 'library',
        component: LibraryView,
    },
    {
        path: '/library/books',
        name: 'books',
        component: BooksView,
    },
    {
        path: '/library/authors',
        name: 'authors',
        component: AuthorsView,
    },
]
```

Хоча маршрути є частиною єдиного цілого, але описані як окремі незалежні маршрути

Дочірні маршрути

index.js router M

src > router > index.js > routes

You, 2 minutes ago | 1 author (You)

```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'
import LibraryView from '../views/LibraryView.vue'
import BooksView from '../views/BooksView.vue'
import AuthorsView from '../views/AuthorsView.vue'

const routes = [
    {
        path: '/',          You, 3 minutes ago • Uncommitted ch
        name: 'home',
        component: HomeView,
    },
    {
        path: '/library',
        name: 'library',
        component: LibraryView,
    },
    {
        path: '/library/books',
        name: 'books',
        component: BooksView,
    },
    {
        path: '/library/authors',
        name: 'authors',
        component: AuthorsView,
    },
]
```

LibraryView.vue views U

src > views > LibraryView.vue > {} "LibraryView.vue" > te

```
<template>
    <hr />
    <router-link :to="{ name: 'books' }
    ">Books</router-link> |
    <router-link :to="{ name: 'authors' }
    ">Authors</router-link>
</template>
<script>
export default {
    name: 'LibraryView',
}
</script>
```

BooksView.vue views U

src > views > BooksView.vue > {} "BooksView.vue" > script

```
<template>
    <h1>Books :</h1>
    <div v-for="book in books" :key="book.
    id">{{ book.title }} - {{ book.price }}
    </div>
</template>
<script>
import { mapState } from 'vuex'

export default {
    name: 'BooksView',
    computed: {
        ...mapState(['books']),
    },
}
</script>
```

Books | Authors

AuthorsView.vue views U

ews > AuthorsView.vue > {} "AuthorsView.vue" > scrip

```
<template>
    <h1>Authors :</h1>
    <div v-for="author in authors"
    :key="author.id">{{ author.
    name }} - {{ author.genre }}</div>
</template>
<script>
import { mapState } from 'vuex'

export default {
    name: 'BooksView',
    computed: {
        ...mapState(['authors']),
    },
}
</script>
```

Дочірні маршрути. Опис як дочірніх маршрутів

index.js router M

src > router > index.js > routes

You, 2 minutes ago | 1 author (You)

```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'
import LibraryView from '../views/LibraryView.vue'
import BooksView from '../views/BooksView.vue'
import AuthorsView from '../views/AuthorsView.vue'

const routes = [
  {
    path: '/',        You, 3 minutes ago • Uncommitted ch
    name: 'home',
    component: HomeView,
  },
  {
    path: '/library',
    name: 'library',
    component: LibraryView,
  },
  {
    path: '/library/books',
    name: 'books',
    component: BooksView,
  },
  {
    path: '/library/authors',
    name: 'authors',
    component: AuthorsView,
  },
]
```

```
const routes = [
  {
    path: '/',
    name: 'home',
    component: HomeView,
  },
  {
    path: '/library',
    name: 'library',
    component: LibraryView,
    children: [
      {
        path: '~~/library/~~books',
        name: 'books',
        component: BooksView,
      },
      {
        path: '~~/library/~~authors',
        name: 'authors',
        component: AuthorsView,
      },
    ],
  },
]
```

Частину “/library” можна не вказувати

Дочірні маршрути. Використання вкладеного <router-view/> для дочріних компонентів

LibraryView.vue views U

src > views > LibraryView.vue > { } "LibraryView.vue"

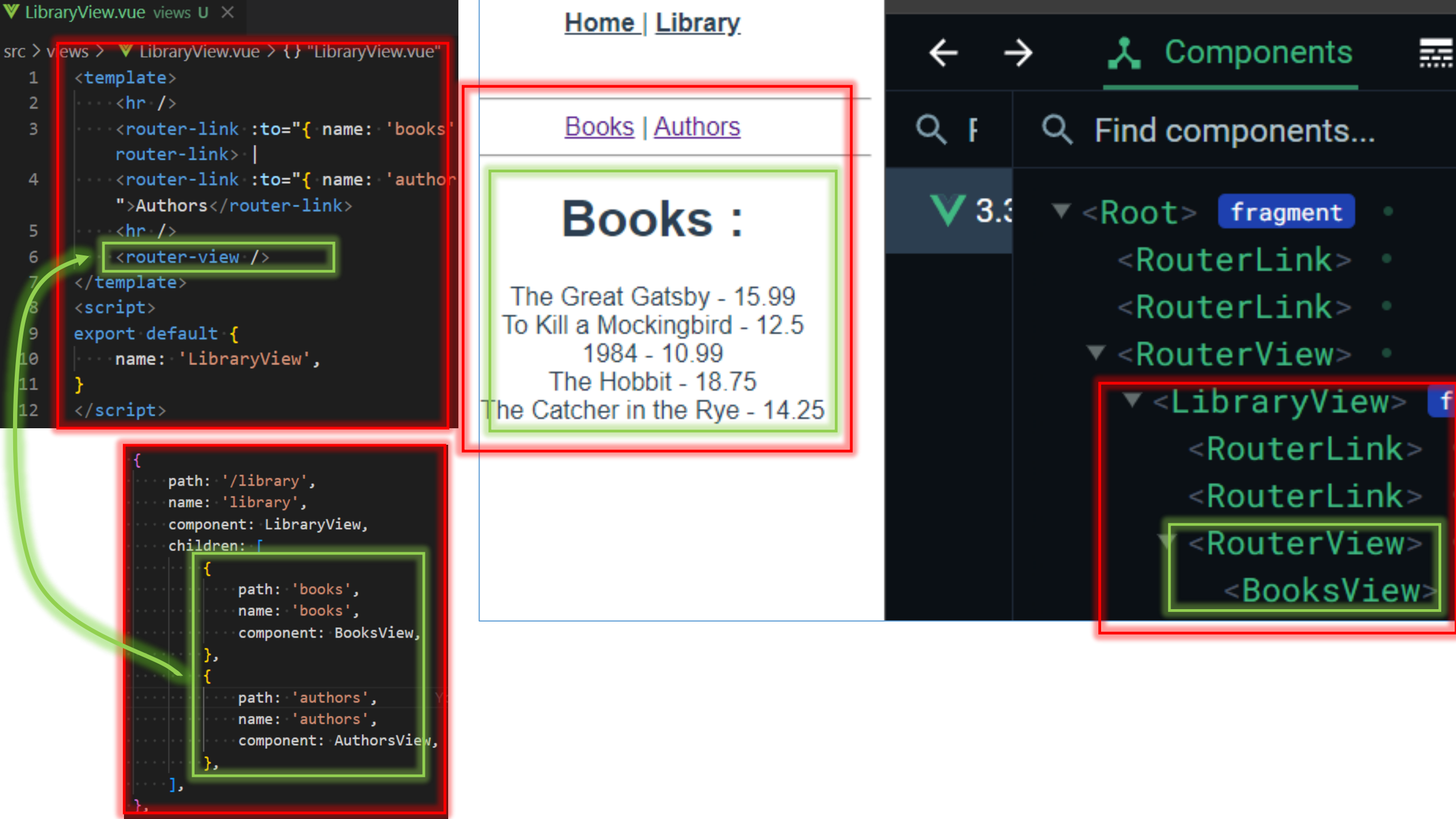
```vue
<template>
    <hr />
    <router-link :to="{ name: 'books' }">Books</
    router-link> |
    <router-link :to="{ name: 'authors' }
    ">Authors</router-link>
    <hr />
    <router-view />
</template>
<script>
export default {
    name: 'LibraryView',
}
</script>
<style lang="scss" scoped></style>
```

Місце для монтування дочірніх компонентів

index.js router M

src > router > index.js > routes > children

```js
import LibraryView from '../views/LibraryView.vue'
import BooksView from '../views/BooksView.vue'
import AuthorsView from '../views/AuthorsView.vue'

const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },
    {
        path: '/library',
        name: 'library',
        component: LibraryView,
        children: [
            {
                path: 'books',
                name: 'books',
                component: BooksView,
            },
            {
                path: 'authors',        You, now • Un
                name: 'authors',
                component: AuthorsView,
            },
        ],
    },
]
```


LibraryView.vue views U
src > views > LibraryView.vue > {} "LibraryView.vue"
<template>
<hr />
<router-link :to="{ name: 'books'
router-link> |
<router-link :to="{ name: 'author
">Authors</router-link>
<hr />
<router-view />
</template>
<script>
export default {
name: 'LibraryView',
}
</script>
{
path: '/library',
name: 'library',
component: LibraryView,
children: [
{
path: 'books',
name: 'books',
component: BooksView,
},
{
path: 'authors',
name: 'authors',
component: AuthorsView,
},
],
},
Home | Library
Books | Authors
Books :
The Great Gatsby - 15.99
To Kill a Mockingbird - 12.5
1984 - 10.99
The Hobbit - 18.75
The Catcher in the Rye - 14.25
Components
Find components...
<Root> fragment
<RouterLink>
<RouterLink>
<RouterView>
<LibraryView>
<RouterLink>
<RouterLink>
<RouterView>
<BooksView>

Приклад.
За маршрутами відображаємо таку інформацію
/teachers -- відображаємо картки з вчителями, за маршрутом
/teachers/:id -- відображаємо гуртки, які веде вчитель

Динамічний роутинг

Задання параметрів

Загальна форма	Приклад

`{ path: '/ шлях / : параметр' }`	`// /:orderId -> ідентифікатор продукту`
	
	`{ path: '/products/:orderId' },`

- Значення параметру є обов’язковим
- є рядком
- рядок може містити значення рядкове представння довільного типу (число, текст, ..)

Динамічний роутинг

Задання параметрів

Загальна форма	Приклад

`{ path: '/ шлях / : параметр' }`	`// /:orderId -> ідентифікатор продукту`
	
	`{ path: '/products/:orderId' },`

Регулярні вирази для параметрів

Загальна форма	Приклад

`{ path:'/шлях/ : параметр ( регулярний_вираз )' }`	`// /:orderId -> тільки цифри`
	
	`{ path: '/products/ :orderId(\\d+)' }`

- Значення параметру є обов’язковим
- є рядком
- рядок може містити тільки значення, що відповідає регурялному виразу (потрібно екранувати символ “\”, тобто, наприклад, для чила вираз не “\d” а “\\d”)

Приклад. Дано список книг і авторів. Організувати перехід за такими маршрутами:

- вибірка книги за ***числовим id***
  /books/2
- вибірка авторів за жанром
  /books/Classic

```
books: [
    { id: 1, title: 'The Great Gatsby', author:
    'F. Scott Fitzgerald', price: 15.99 },
    { id: 2, title: 'To Kill a Mockingbird',
    author: 'Harper Lee', price: 12.5 },
    { id: 3, title: '1984', author: 'George
    Orwell', price: 10.99 },
    { id: 4, title: 'The Hobbit', author: 'J.R.R.
    Tolkien', price: 18.75 },
    { id: 5, title: 'The Catcher in the Rye',
    author: 'J.D. Salinger', price: 14.25 },
],
authors: [
    { id: 1, name: 'Jane Austen', genre:
    'Classic' },
    { id: 2, name: 'George R.R. Martin', genre:
    'Fantasy' },
    { id: 3, name: 'Haruki Murakami', genre:
    'Magical' },
    { id: 4, name: 'Agatha Christie', genre:
    'Mystery' },
    { id: 5, name: 'J.K. Rowling', genre:
    'Fantasy' },
],
```

Приклад. Дано список книг і авторів. Організувати перехід за такими маршрутами:

- вибірка книги за ***числовим id***
  /books/2
- вибірка авторів за жанром
  /books/Classic

```
books: [
    { id: 1, title: 'The Great Gatsby', author:
    'F. Scott Fitzgerald', price: 15.99 },
    { id: 2, title: 'To Kill a Mockingbird',
    author: 'Harper Lee', price: 12.5 },
    { id: 3, title: '1984', author: 'George
    Orwell', price: 10.99 },
    { id: 4, title: 'The Hobbit', author: 'J.R.R.
    Tolkien', price: 18.75 },
    { id: 5, title: 'The Catcher in the Rye',
    author: 'J.D. Salinger', price: 14.25 },
],
authors: [
    { id: 1, name: 'Jane Austen', genre:
    'Classic' },
    { id: 2, name: 'George R.R. Martin', genre:
    'Fantasy' },
    { id: 3, name: 'Haruki Murakami', genre:
    'Magical' },
    { id: 4, name: 'Agatha Christie', genre:
    'Mystery' },
    { id: 5, name: 'J.K. Rowling', genre:
    'Fantasy' },
],
```

Такий спосіб

```
{ path: '/books/ :orderId' }
```

не підходить, бо немає можливості визначити тип параметра

Приклад. Дано список книг і авторів. Організувати перехід за такими маршрутами:

- вибірка книги за ***числовим id***
  /books/2
- вибірка авторів за жанром
  /books/Classic

```
{
  path: '/ books / :id(\\d+)‘,
  component : BooksView
},

{
   path: '/ books / :id'
   component: AuthorsView
}
```

Тільки цифри

Всі інші значення

```
books: [
    { id: 1, title: 'The Great Gatsby', author:
    'F. Scott Fitzgerald', price: 15.99 },
    { id: 2, title: 'To Kill a Mockingbird',
    author: 'Harper Lee', price: 12.5 },
    { id: 3, title: '1984', author: 'George
    Orwell', price: 10.99 },
    { id: 4, title: 'The Hobbit', author: 'J.R.R.
    Tolkien', price: 18.75 },
    { id: 5, title: 'The Catcher in the Rye',
    author: 'J.D. Salinger', price: 14.25 },
],
authors: [
    { id: 1, name: 'Jane Austen', genre:
    'Classic' },
    { id: 2, name: 'George R.R. Martin', genre:
    'Fantasy' },
    { id: 3, name: 'Haruki Murakami', genre:
    'Magical' },
    { id: 4, name: 'Agatha Christie', genre:
    'Mystery' },
    { id: 5, name: 'J.K. Rowling', genre:
    'Fantasy' },
],
```

Приклад. Дано список книг і авторів. Організувати перехід за такими маршрутами:

- вибірка книги за ***числовим id***

  /books/2

```
{
    path: '/books/:id(\\d*)',
    name: 'books',
    component: BooksView,
},
```

- вибірка авторів за жанром

  /books/Classic

```
{
    path: '/books/:id',
    name: 'authors',
    component: AuthorsView,
},
```

```
books: [
    { id: 1, title: 'The Great Gatsby', author:
    'F. Scott Fitzgerald', price: 15.99 },
    { id: 2, title: 'To Kill a Mockingbird',
    author: 'Harper Lee', price: 12.5 },
    { id: 3, title: '1984', author: 'George
    Orwell', price: 10.99 },
    { id: 4, title: 'The Hobbit', author: 'J.R.R.
    Tolkien', price: 18.75 },
    { id: 5, title: 'The Catcher in the Rye',
    author: 'J.D. Salinger', price: 14.25 },
],
authors: [
    { id: 1, name: 'Jane Austen', genre:
    'Classic' },
    { id: 2, name: 'George R.R. Martin', genre:
    'Fantasy' },
    { id: 3, name: 'Haruki Murakami', genre:
    'Magical' },
    { id: 4, name: 'Agatha Christie', genre:
    'Mystery' },
    { id: 5, name: 'J.K. Rowling', genre:
    'Fantasy' },
],
```

Динамічний роутинг. Використання мультрипараметрів

- використовується коли параметр може повторюватись
- задається з використанням квантифікаторів регулярних виразів
  «+» - кількість параметрів >= 1
  «*» - кількість параметрів >= 0
  «?» - необов’язковий параметр
- параметр є масивом

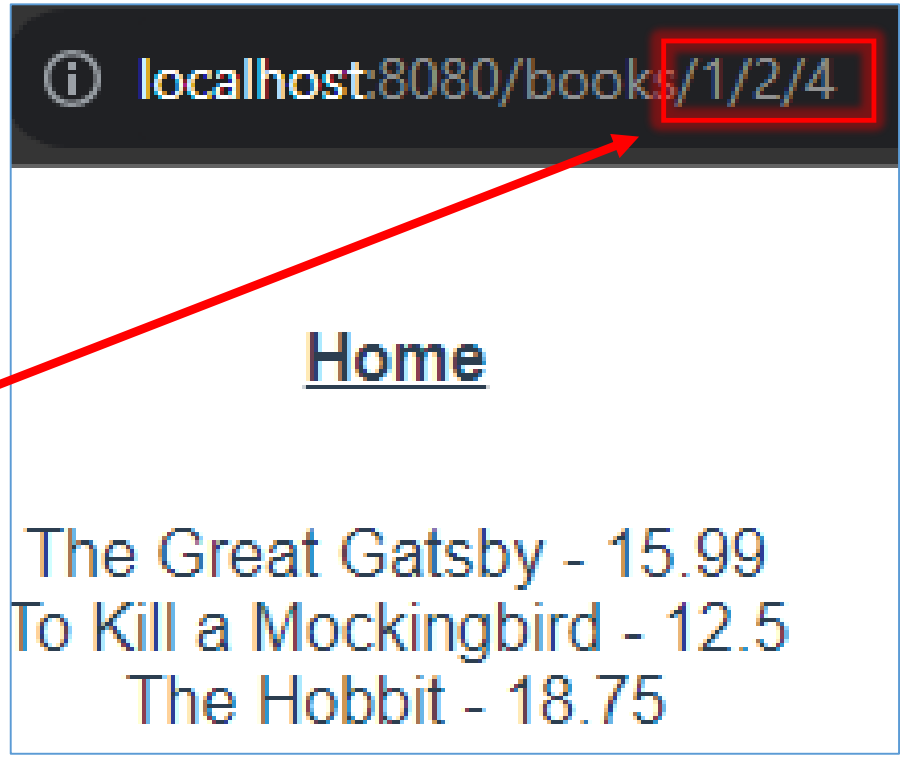
Загальна форма	Приклад

`//--- кількість параметрів >=1 ---`
`{`
`  path: '/ шлях / : параметр+'`
`}`
	`//--- кількість параметрів >=1 ---`
	
	`{`
	`  path: '/products/:orderId+'`
	`}`

`//--- кількість параметрів >=0 ---`
`{`
`  path: '/ шлях / : параметр*'`
`}`
	`//--- кількість параметрів >=0 ---`
	`{`
	`  path: '/products/:orderId*'`
	`}`

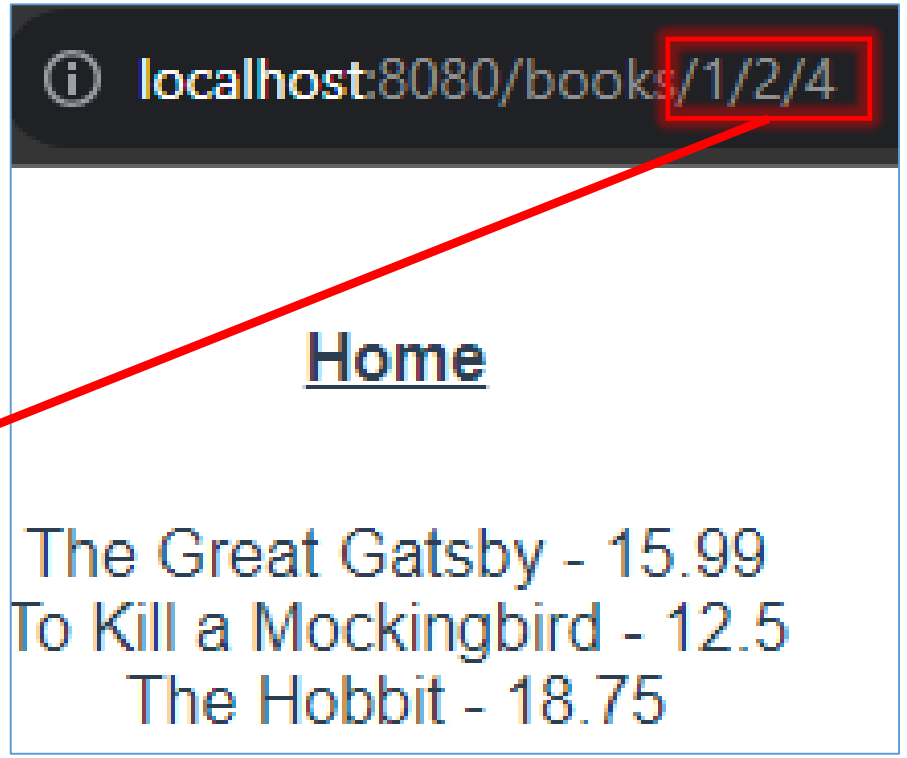
Приклад. Розробити маршрут, який може приймати довільну кількість ідентифікаторів книг

Відображаємо книги з переданими через маршрут id

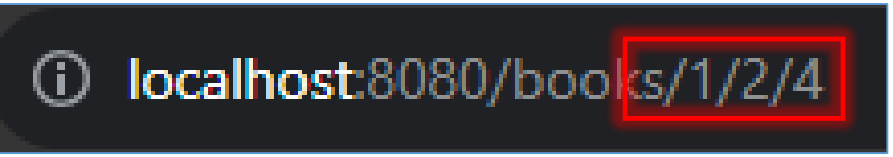


Приклад. Розробити маршрут, який може приймати довільну кількість ідентифікаторів книг

```
import { createRouter, createWebHistory } from
'vue-router'
import HomeView from '../views/HomeView.vue'
import BooksView from '../views/BooksView.vue'
import AuthorsView from '../views/AuthorsView.vue'

const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },      You, 50 seconds ago • Uncommitted chan
    {
        path: '/books/:id(\\d*)+',
        name: 'books',
        component: BooksView,
    },
    {
        path: '/books/:id',
        name: 'authors',
        component: AuthorsView,
    },
]
```



Приклад. Розробити маршрут, який може приймати довільну кількість ідентифікаторів книг



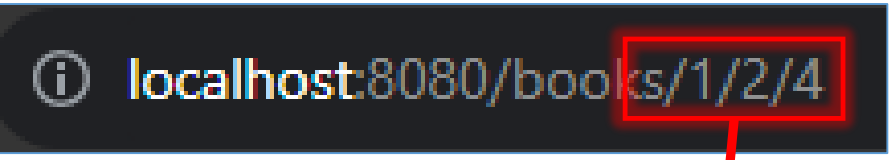

```
import HomeView from '../views/HomeView.vue'
import BooksView from '../views/BooksView.vue'
import AuthorsView from '../views/AuthorsView.vue'

const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },      You, 50 seconds ago • Uncommitted chan
    {
        path: '/books/:id(\\d*)+',
        name: 'books',
        component: BooksView,
    },
    {
        path: '/books/:id',
        name: 'authors',
        component: AuthorsView,
    },
]
```

BooksView.vue views U

src > views > BooksView.vue > {} "BooksView.vue" > template >

```
1   <template>
2       <book-item v-for="bookId in booksIdList"
            :key="bookId" :book-id="bookId" />
3   </template>
4   <script>
5   import BookItem from '@/components/BookItem.vue'
6
7   export default {
8       name: 'BooksView',
9
10      components: { BookItem },
11
12      computed: {
13          booksIdList() {
14              return this.$route.params.id
15          },
16      },
17  }
18  </script>
```

Приклад. Розробити маршрут, який може приймати довільну кількість ідентифікаторів книг



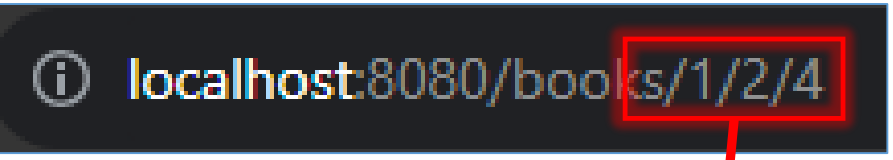

```
import HomeView from '../views/HomeView.vue'
import BooksView from '../views/BooksView.vue'
import AuthorsView from '../views/AuthorsView.vue'

const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },      You, 50 seconds ago • Uncommitted chan
    {
        path: '/books/:id(\\d*)+',
        name: 'books',
        component: BooksView,
    },
    {
        path: '/books/:id',
        name: 'authors',
        component: AuthorsView,
    },
]
```

BooksView.vue views U

src > views > BooksView.vue > {} "BooksView.vue" > template >

```
1   <template>
2       <book-item v-for="bookId in booksIdList"
            :key="bookId" :book-id="bookId" />
3   </template>
4   <script>
5   import BookItem from '@/components/BookItem.vue'
6
7   export default {
8       name: 'BooksView',
9
10      components: { BookItem },
11
12      computed: {
13          booksIdList() {
14              return this.$route.params.id
15          },
16      },
17  }
18  </script>
```

Приклад. Розробити маршрут, який може приймати довільну кількість ідентифікаторів книг




```
import HomeView from '../views/HomeView.vue'
import BooksView from '../views/BooksView.vue'
import AuthorsView from '../views/AuthorsView.vue'

const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },      You, 50 seconds ago • Uncommitted chan
    {
        path: '/books/:id(\\d*)+',
        name: 'books',
        component: BooksView,
    },
    {
        path: '/books/:id',
        name: 'authors',
        component: AuthorsView,
    },
]
```

BooksView.vue views U

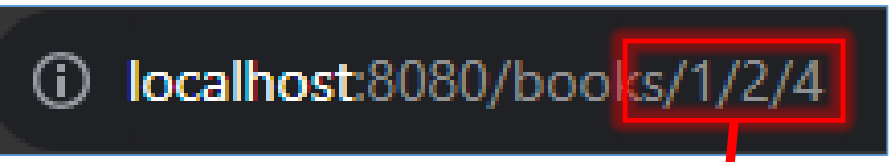
src > views > BooksView.vue > {} "BooksView.vue" > template >

```
1   <template>
2       <book-item v-for="bookId in booksIdList"
            :key="bookId" :book-id="bookId" />
3   </template>
4   <script>
5   import BookItem from '@/components/BookItem.vue'
6
7   export default {
8       name: 'BooksView',
9
10      components: { BookItem },
11
12      computed: {
13          booksIdList() {
14              return this.$route.params.id
15          },
16      },
17  }
18  </script>
```

['1', '2', '4']

параметр є масивом

Приклад. Розробити маршрут, який може приймати довільну кількість ідентифікаторів книг




```
import HomeView from '../views/HomeView.vue'
import BooksView from '../views/BooksView.vue'
import AuthorsView from '../views/AuthorsView.vue'

const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },      You, 50 seconds ago • Uncommitted chan
    {
        path: '/books/:id(\\d*)+',
        name: 'books',
        component: BooksView,
    },
    {
        path: '/books/:id',
        name: 'authors',
        component: AuthorsView,
    },
]
```

BooksView.vue views U

src > views > BooksView.vue > {} "BooksView.vue" > template >

```
1   <template>
2       <book-item v-for="bookId in booksIdList"
            :key="bookId" :book-id="bookId" />
3   </template>
4   <script>
5   import BookItem from '@/components/BookItem.vue'
6
7   export default {
8       name: 'BooksView',
9
10      components: { BookItem },
11
12      computed: {
13          booksIdList() {
14              return this.$route.params.id
15          },
16      },
17  }
18  </script>
```

['1', '2', '4']

параметр є масивом

Приклад. На одній сторінці вибираємо декілька авторів, інформацію про яких треба відбразити на іншій сторінці.

Vue route. Props. Дозволяє передавати параметри маршрута через props:

- булевий режим (***props:true***)
- об’єктний режим
- функціональний режим

```
import BookItem from '@/components/
BookItem.vue'

const routes = [
    {…
    },
    {
        path: '/books/:bookId',
        name: 'books',
        component: BookItem,
    },
```

Зазвичай у компоненті ми отримували значення параметра через об’єкт $route

this.$route.params. параметр

BookItem.vue components U

src > components > BookItem.vue > {} "BookItem.vue"

```
1   <template>
2       <div v-if="book">{{ book.title }} - {{ book.price }}</div>
3   </template>
4
5   <script>
6   import { mapGetters } from 'vuex'
7
8   export default {
9       name: 'BookItem',
10
11      computed: {
12          ...mapGetters(['getBookById']),
13          bookId() {
14              return this.$route.params.bookId
15          },
16          book() {
17              return this.getBookById(this.bookId)
18          },
19      },
20  }
21  </script>
```

Vue route. Props. Дозволяє передавати параметри маршрута через props:

- булевий режим (***props:true***)
- об'єктний режим
- функціональний режим

```
import BookItem from '@/components/
BookItem.vue'

const routes = [
    {...
    },
    {
        path: '/books/:bookId',
        name: 'books',
        component: BookItem,
        props: true,
    },
```

Зараз значення параметра передається як властивість. Якщо значення
props:true
то, в якості властивостей передається
$route.params

BookItem.vue components U

src > components > BookItem.vue > {} "BookItem.vue" > script > default

```
1   <template>
2       <div v-if="book">{{ book.title }} - {{ book.price }}</div>
3   </template>
4
5   <script>
6   import { mapGetters } from 'vuex'
7
8   export default {
9       name: 'BookItem',
10
11      props: {
12          bookId: {
13              type: [String, Number],
14              required: true,
15          },
16      },
17
18      computed: {
19          ...mapGetters(['getBookById']),
20          book() {
21              return this.getBookById(this.bookId)
22          },
23      },
24  }
25  </script>
```

Vue route. Props. Дозволяє передавати параметри маршрута через props:

- булевий режим (***props:true***)
- об’єктний режим
- функціональний режим

props : { власт.1:знач.1, власт.2:знач.2, …}

```
{
    path: '/test',
    name: 'test',
    component: TestView,
    props: { userName: 'Ivan', userAge: 21 },
},
```

Знаення властивості props передається як об’єкт вхідних параметів props для компонента

TestView.vue views U

src > views > TestView.vue > {} "TestView.vue"

```
<template>
    <div>{{ userName }} - {{ userAge }}</div>
</template>

<script>
export default {
    name: 'TestView',

    props: {
        userName: {
            type: String,
            required: true,
        },
        userAge: {
            type: Number,
            required: true,
        },
    },
}
</script>
```

Vue route. Props. Дозволяє передавати параметри маршрута через props:

- булевий режим (***props:true***)
- об'єктний режим
- функціональний режим

props : ***route*** => (*об'єкт_props_для_компонента*)

```
{
    path: '/books/:id',        You, 3 minutes ago • Uncommitted
    name: 'books',
    component: BooksView,
    props: (route) => ({ bookId: parseInt(route.params.id) }),
},
```

BooksView copy.vue views U

src > views > BooksView copy.vue > {} "BooksView copy.vue"

```
<template>
    <book-item :book-id="bookId" />
</template>
<script>
import BookItem from '@/components/BookItem.vue'

export default {
    name: 'BooksView',

    components: { BookItem },

    props: {
        bookId: {
            type: Number,
            required: true,
        },
    },
}
</script>
<style lang="scss" scoped></style>
```

Зараз значення bookId передається як властивість типу Number.
Функція як параметр отримує ***route*** і повертає об'єкт, який використовується як об'єкт віхдних даних ***props*** у компоненті (можуть бути не тільки значення - параметри)

PageNotFound (404)

Потрібно відбражати коли:

- не існує маршруту з таким шаблоном
- шаблон існує, але відповідних даних не існує

Обробка неіснуючих маршрутів. PageNotFound. *Не існує шаблону для співпадіння*

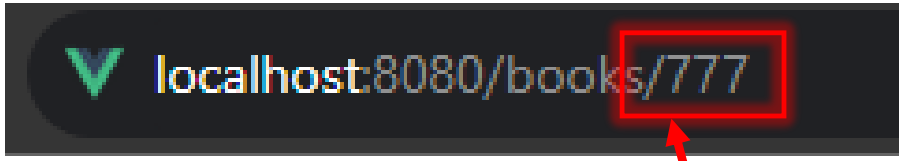
```
import NotFound from '@/components/NotFound.vue'

const routes = [
    {...
    },
    {...
    },
    {
        path: '/:pathMatch(.*)*',
        name: 'NotFound',
        component: NotFound,
    },
]
```

Додаємо в кінець списку

Обробка неіснуючих маршрутів. PageNotFound. ***<u>Шаблон існує, але не існує відповідних даних</u>***

```vue
import BookItem from '@/components/BookItem.vue'

const routes = [
    {…
    },
    {
        path: '/books/:id',
        name: 'books',
        component: BookItem,
    },
```

```vue
state: {
    books: [
        { id: 1, title: 'The Great Gatsby', author:
        'F. Scott Fitzgerald', price: 15.99 },
        { id: 2, title: 'To Kill a Mockingbird',
        author: 'Harper Lee', price: 12.5 },
        { id: 3, title: '1984', author: 'George
        Orwell', price: 10.99 },
        { id: 4, title: 'The Hobbit', author: 'J.R.R.
        Tolkien', price: 18.75 },
        { id: 5, title: 'The Catcher in the Rye',
        author: 'J.D. Salinger', price: 14.25 },
    ],
```



Книги з таким id немає

Хуки роутів на компонентах. In-Component Guards

Можна додати *до комонента* або *до опцій роутів* (у *router.js*):

beforeRouteEnter(to, from, next) { . . . . . }
- викликається коли маршрут, за яким треба перйти визначено
- не має доступу до this, бо компонент ще не згенерований
- якщо потірбно щось з стора, то треба його явно імпортувати !!!

//----------

beforeRouteUpdate(to, from , next) { . . . . . }
- використовується коли відбувається перехід до нового маршруту, який використовє цей же компонент(пр. “/book/1” => “/book/2”)
- має доступ до this

beforeRouteLeave(to, from , next) { . . . . . }
- використовується коли необхідно перейти на інший маршрут
- має доступ до this

Властивості ***to(цільовий маршрут), from(поточний маршрут)***:
- **fullPath**
- **path**
- **name**
- **meta**
- **params**

https://vue-router-ru.netlify.app/api/interfaces/RouteLocationNormalized.html

```
<template>…
<script>
export default {

  beforeRouteEnter(to, from, next) {
  },
  beforeRouteUpdate(to, from, next) {
  },
  beforeRouteLeave(to, from, next) {
  },

  data() {…
  },
  methods: {…
  },

}
</script>
```

```
const routes = [
  {…
  },
  {
    path: '/books/:id',
    name: 'books',
    component: BookItem,
    beforeEnter(to) {
      const bookExists = store.state.books.find
      ((book) => book.id == to.params.id)
      if (!bookExists) return { name: 'NotFound' }
    },
  },
  {…
  },
```

Обробка неіснуючих маршрутів. PageNotFound. Шаблон існує, але не існує відповідних параметрів

```
import store from '@/store'

const routes = [
    {…
    },
    {
        path: '/books/:id',
        name: 'books',
        component: BookItem,
        beforeEnter(to) {
            const bookExists = store.state.books.find
            ((book) => book.id == to.params.id)
            if (!bookExists) return { name: 'NotFound' }
        },
    },
    {…
    },
    {
        path: '/:pathMatch(.*)*',
        name: 'NotFound',
        component: NotFound,
    },
]
```

```
state: {
    books: [
        { id: 1, title: 'The Great Gatsby', author:
        'F. Scott Fitzgerald', price: 15.99 },
        { id: 2, title: 'To Kill a Mockingbird',
        author: 'Harper Lee', price: 12.5 },
        { id: 3, title: '1984', author: 'George
        Orwell', price: 10.99 },
        { id: 4, title: 'The Hobbit', author: 'J.R.R.
        Tolkien', price: 18.75 },
        { id: 5, title: 'The Catcher in the Rye',
        author: 'J.D. Salinger', price: 14.25 },
    ],
```

Перевіряємо чи коректний параметр id. Якщо некоректний – переходимо до сторінки NotFound

Мета-дані маршрутів. Властивість meta

- використовуються для додавання деяких додаткових даних для маршруту
- додаємо/отримуємо через властивість “route.meta”

https://router.vuejs.org/guide/advanced/meta

Мета-дані маршрутів. Властивість meta

- використовуються для додавання деяких довільних додаткових даних для маршруту
- додаємо/отримуємо через властивість “route.meta”

```
{
    path: '/students',
    name: 'students',
    component: StudentsView,
    meta: {
        requiresAuth: true,
        menuOptions: {
            iconClass: 'fa-video'
        }
        displayedActiveMenuOptName: 'save'
    },
},
```

```
const router = createRouter({
    history: createWebHistory(),
    mode: 'history',
    routes,
})      You, 3 minutes ago • Uncommitted changes

router.beforeEach((to) => {
    const isAuthenticated = window.userName
    if (to.meta.requiresAuth && !isAuthenticated) {
        return {
            name: 'LoginPage',
            query: { redirect: to.fullPath },
        }
    }
})
```

https://router.vuejs.org/guide/advanced/meta

Приклад. Виводити навігаційне меню тільки для деяких маршрутів

Глобальні навігаційні хуки

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

- викликається перед початком переходу за маршрутом

router.beforeResolve ((to, from) => { …})

- викликається перед підтвердженням переходу за маршрутом
- спрацьовує після хуків навігації на компоненті

router.afterEach ((to, from) => { …})

- викликається після завершення переходу за маршрутом
- додатково може отримувати failure – індикатор невдалого переходу

Приклад:

```
router.afterEach((to, from, failure) => {
  if (!failure) sendToAnalytics(to.fullPath)
})
```

Можуть бути описані після створення об’єкта роутера

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

const routes = [
  {…
  },
  {…
  },
  {…
  },
  {…
  },
]

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

router.beforeEach((to) => {…
})
      You, 11 hours ago • Uncommitted changes
export default router
```

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

Обмеження доступу до сторінок, що потребують авторизації

1)Створюємо LoginPage
2)Додаємо атрибут meta до роутів з властивістю requiresAuth
3) Додаємо хук router.beforeEach, у якому робимо первірку чи авторизований

LoginPage.vue views U

src > views > LoginPage.vue > {} "LoginPage.vue" > template > div

```vue
<template>
    <div> User name
        <input v-model="userName" type="text" />
    </div>
    <div> Password
        <input v-model="password" type="password" />
    </div>
    <button @click="onLogin">Login</button>
</template>
<script>
export default {
    name: 'LoginPage',
    data() {
        return {
            userName: null,
            password: null,
        }
    },
    methods: {
        onLogin() {
            //--- перевірка правильності ----
            window.userName = this.userName
            if (this.$route.query.redirect)
                this.$router.push({
                    path: this.$route.query.redirect,
                })
            else this.$router.push({ path: '/' })
        },
    },
}
</script>
```

Обмеження доступу до сторінок, що потребують авторизації

1)Створюємо LoginPage

2)Додаємо атрибут meta до роутів з властивістю requiresAuth

3) Додаємо хук router.beforeEach, у якому робимо первірку чи авторизований

```
const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },
    {
        path: '/books/:id',
        name: 'books',
        component: () => import('../views/BooksView.vue'),
        meta: { requiresAuth: false },
    },
    {
        path: '/authors/:id',
        name: 'authors',
        component: () => import('../views/AuthorsView.vue'),
        meta: { requiresAuth: true },
    },
    {
        path: '/login',
        name: 'LoginPage',
        component: () => import('../views/LoginPage.vue'),
        meta: { requiresAuth: false },
    },
]
```

LoginPage.vue views U

src > views > LoginPage.vue > {} "LoginPage.vue" > template > div

```
<template>
    <div> User name
        <input v-model="userName" type="text" />
    </div>
    <div> Password
        <input v-model="password" type="password" />
    </div>
    <button @click="onLogin">Login</button>
</template>
<script>
export default {
    name: 'LoginPage',
    data() {
        return {
            userName: null,
            password: null,
        }
    },
    methods: {
        onLogin() {
            //--- перевірка правильності ----
            window.userName = this.userName
            if (this.$route.query.redirect)
                this.$router.push({
                    path: this.$route.query.redirect,
                })
            else this.$router.push({ path: '/' })
        },
    },
}
</script>
```

Обмеження доступу до сторінок, що потребують авторизації

1)Створюємо LoginPage
2)Додаємо атрибут meta до роутів з властивістю requiresAuth
3) Додаємо хук router.beforeEach, у якому робимо первірку чи авторизований

```
const routes = [
    {
        path: '/',
        name: 'home',
        component: HomeView,
    },
    {
        path: '/books/:id',
        name: 'books',
        component: () => import('../views/BooksView.vue'),
        meta: { requiresAuth: false },
    },
    {
        path: '/authors/:id',
        name: 'authors',
        component: () => import('../views/AuthorsView.vue'),
        meta: { requiresAuth: true },
    },
    {
        path: '/login',
        name: 'LoginPage',
        component: () => import('../views/LoginPage.vue'),
        meta: { requiresAuth: false },
    },
]
```

```
const router = createRouter({
    history: createWebHistory(),
    mode: 'history',
    routes,
})      You, 3 minutes ago • Uncommitted changes

router.beforeEach((to) => {
    const isAuthenticated = window.userName
    if (to.meta.requiresAuth && !isAuthenticated) {
        return {
            name: 'LoginPage',
            query: { redirect: to.fullPath },
        }
    }
})
```