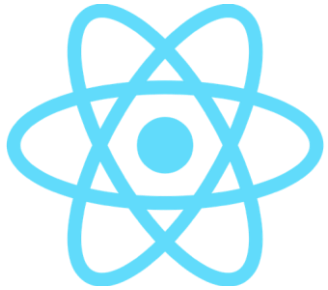


React Базовий

Роутинг та навігація в React

Тема уроку

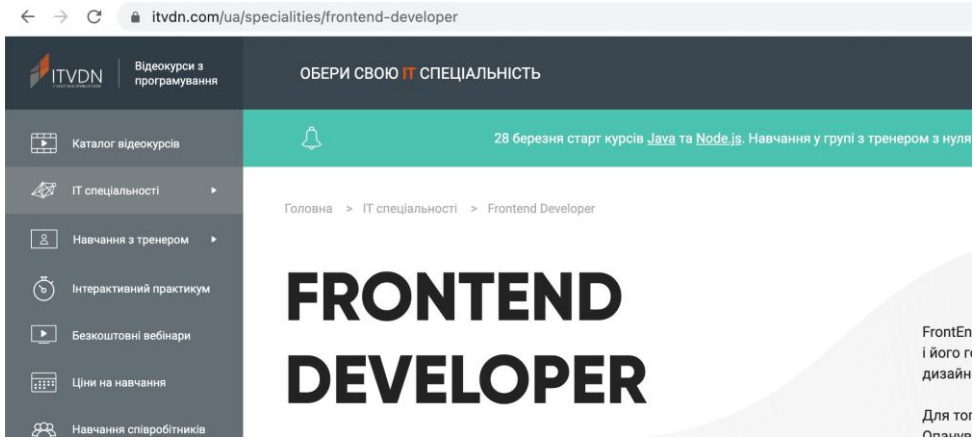
Роутинг та навігація в React

План уроку

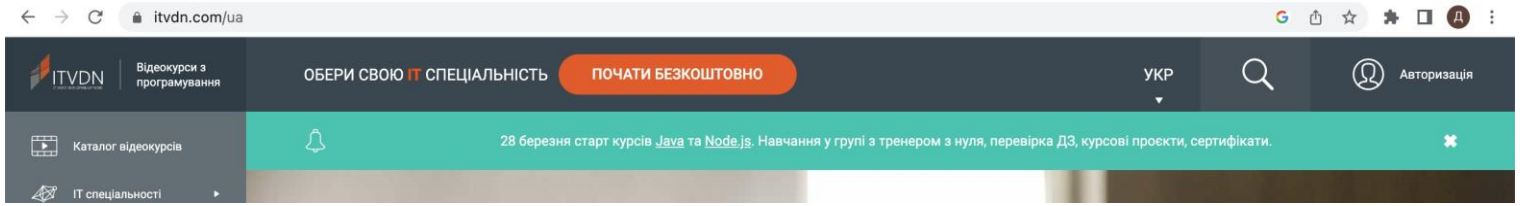
1. Що таке роутинг?
2. Механізм роутингу в React.
3. Схема маршрутизації createBrowserRouter.
4. Обробка сторінки з помилкою Error 404.
5. Елементи Link та NavLink.
6. Вкладені маршрути та метод loader.

Що таке роутинг?

Роутинг або маршрутизація - це процес організації маршрутів між посиланнями. У вебзастосунках роутингом називають зміну URL адреси в момент переходу по різним частинам інтернет-ресурсу. Роутинг дозволяє користувачу визначати, в якій частині вебдодатку він знаходиться в даний момент. Також завдяки маршрутизації у користувача є можливість відслідковувати власну історію користування вебдодатком.



Адреса Front-end розділу ITVDN



Адреса головної сторінки ITVDN

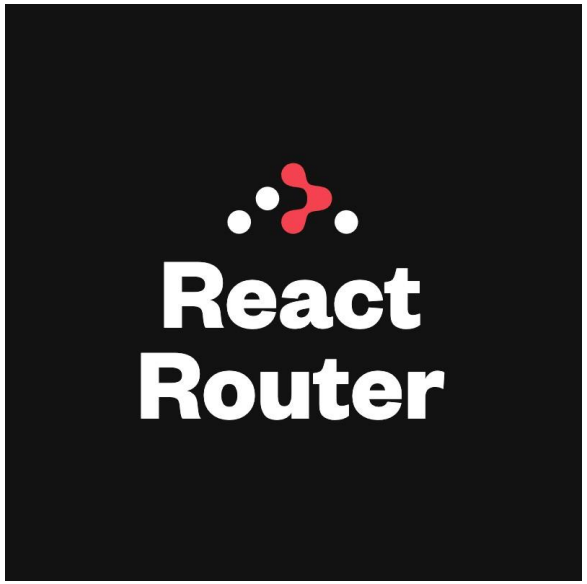
Механізм роутингу в React

За замовчуванням бібліотека React не надає інструментів для побудови маршрутизації, але ці задачі цілком покривають сторонні бібліотеки.

Сама потужна з них - це React Router.

Щоб працювати з даною бібліотекою, її потрібно встановити в проєкт командою:

```
npm i react-router-dom
```



Посилання на бібліотеку:
https://reactrouter.com/en/main

Метод createBrowserRouter

Для побудови маршрутизації React застосунка використовується метод createBrowserRouter. Щоб передавати атрибути та адреси у вигляді JSX-компонентів, використовується допоміжний метод createRoutesFromElements.
Для роботи з даними методами їх необхідно імпортувати з 'react-router-dom'.

```
const router = createBrowserRouter(
    createRoutesFromElements(
        <Route path='/' element={ <Root /> } >
            <Route index element ={ <Welcome /> } />
            <Route path='statehooks' element={ <StateHooks /> } />
            <Route path='community' element={ <Community /> } />
            <Route path='resources' element={ <Resources /> } />
            <Route path='about' element={ <About /> } />
            <Route path='users' element={<Users/>} />
            <Route path='users/:userId' loader={loader} element={<UserPage />} errorElement={<ErrorPage/>} />
            <Route path='*' element={ <ErrorPage /> } />
        </Route>
    )
);
```

Приклад схеми маршрутизації createBrowserRouter

Сторінка Error 404

Якщо за посиланням вебдодатку такої сторінки не передбачено, необхідно реалізувати шаблон сторінки з помилкою.
Бібліотека React Router надає можливість детальної обробки помилок та може відображати код помилки, яка трапилась за допомогою методу useRouteError.

Error.jsx U

src > Pages > Error > Error.jsx > ErrorPage

```jsx
import { useRouteError } from "react-router-dom";

export default function ErrorPage(){
    const error = useRouteError();
    console.log(error);
```

Метод useRouteError для ідентифікації помилки

```jsx
  <Route path='about' element={ <About /> } />
  <Route path='users' element={<Users/>} />
  <Route path='users/:userId' loader={loader} element={<UserPage />} errorElement={<ErrorPage/>} />
  <Route path='*' element={ <ErrorPage /> } />
Route>
```

обробка помилок для неіснуючого юзеру

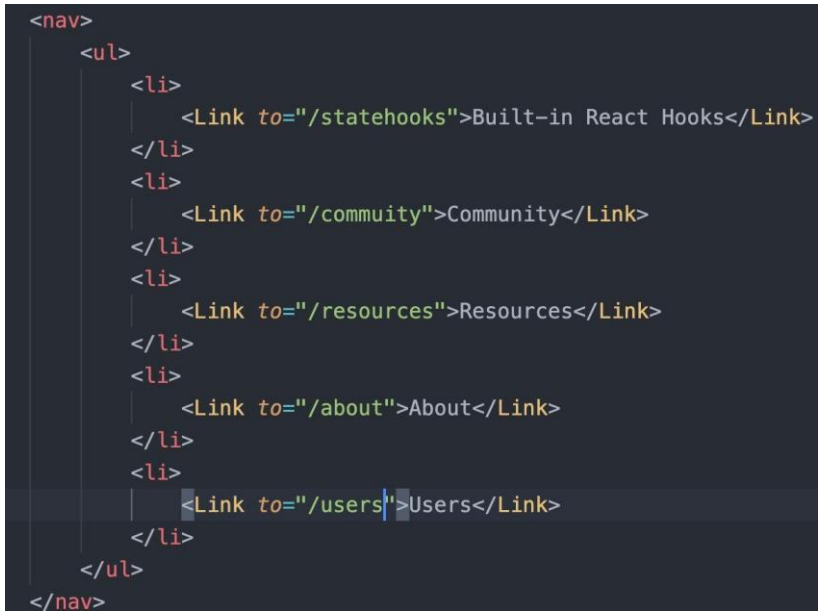
неіснуюча сторінка

Встановлення шаблону компоненту Error, якщо сторінки за посиланням немає в схемі

Елементи Link та NavLink

Щоб використовувати всі переваги реакту, замість звичайного посилання через елемент <a> використовуйте елемент з бібліотеки React Router <Link>. Даний елемент дозволяє «підгружати» тільки ті компоненти, які потрібно, при цьому загального перезавантаження сторінки не відбувається. Елемент <NavLink> аналогічний елементу <Link>, тільки додає атрибут class="active" до посилання, яке активно в даний момент.

Щоб використовувати Link та NavLink, їх потрібно імпортувати в компонент із 'react-router-dom'.



Навігація додатку через Link

Вкладені маршрути та loader

Якщо в вашому застосунку є сторінки, які продовжують гілку посилань, наприклад, профіль конкретного юзера в адресі /users/ , то для маршрутизації таких сторінок в атрибуті path після головної гілки потрібно вказати двокрапку та типізовану назву такої сторінки.

Приклад:

```
<Route path="users" element={ <Users />} />
<Route path="users/:userId" loader={loader} element={<UserPage />} errorElement={ <Error /> }/>
```

Метод loader дозволяє зробити запит до зовнішньої системи (API) та передати отримані дані компоненту, який викликається. Для цього створюється окрема функція, яка передається в атрибут loader. За аргумент функція loader може приймати ім’я сторінки, для якої необхідно зробити запит. Наприклад, якщо викликається сторінка /users/john, функція loader отримає аргумент john.

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 ви 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 вам отримати відповідний сертифікат.