

WMNWA 2210

WEB APP DEVELOPMENT

React 8 | Routes





ΠΕΡΙΕΧΟΜΕΝΑ



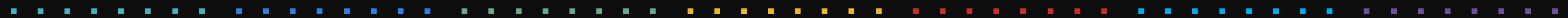


Περιεχόμενα

- Installation
- Basic Setup
 - NoMatch
- Navigation
 - **Link & NavLink**
 - **useNavigate & Navigate**
 - **useParams & useSearchParams**
 - **useLocation**
- Nested routes
 - **Outlet** component
- Lazy routes



ROUTES





Routes

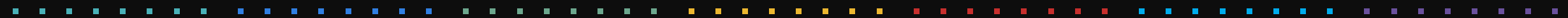
What's a route?

Στη γλώσσα των *Single-page apps*, η έννοια της σελίδας (*page*), αντικαθίσταται από αυτή του *route*.

Αν σε μια κλασική εφαρμογή *Web*, κάνουμε κλικ σε συνδέσμους (*links*) για να πλοηγηθούμε σε σελίδες, σε ένα *Single-page app*, παραμένουμε στην ίδια σελίδα, αλλά εναλλάσσονται τα **Component** που εμφανίζονται σε αυτή.

Τα **Component** αυτά μπορεί να εκτείνονται, είτε σε όλο το σώμα της σελίδας, είτε σε ένα μέρος της, μιας και αρκετά τμήματα μιας σελίδας παραμένουν κοινά με των υπολοίπων.

Το όφελος αυτής της προσέγγισης είναι πως, μιας και δεν αλλάζει ποτέ η σελίδα, διατηρείται με ευκολία το *state* της εφαρμογής.





Routes

React Router

Η **React** συχνά αναφέρεται ως μία βιβλιοθήκη για τη δημιουργία *Single-page apps*, αλλά αυτό προϋποθέτει την ύπαρξη ενός κατάλληλου *router* για την διαχείριση των *routes* στον *browser*.

Η **React** δεν περιλαμβάνει τέτοια λειτουργικότητα, αλλά υπάρχει η πλέον διαδεδομένη βιβλιοθήκη **React Router** για το σκοπό αυτό.

Η **React Router** είναι μια πλήρως εξοπλισμένη βιβλιοθήκη δρομολόγησης για τη **React**.

Η **React Router** εκτελείται οπουδήποτε εκτελείται και η **React**. Στον ιστό (*client*), στον διακομιστή (*server*) με **node.js**, και στο **React Native**.



REACT ROUTER



React Router

Installation

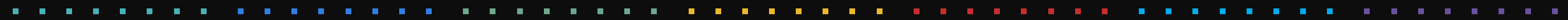
Αρχικά, μιας και η **React Router** είναι μια "εξωτερική" βιβλιοθήκη θα πρέπει να φροντίσουμε να την προσθέσουμε στο *project* μας.

```
npm install react-router-dom
```

```
yarn add react-router-dom
```




BASIC SETUP





Basic Setup

Router

Για να ξεκινήσουμε να δουλεύουμε με τη **React Router** θα πρέπει να συμπεριλάβουμε έναν από τους **Router** που υπάρχουν στη βιβλιοθήκη.

```
import { BrowserRouter as Router } from 'react-router-dom';

ReactDOM.render(
  <React.StrictMode>
    <Router>
      <App />
    </Router>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Ο **Router** θα πρέπει να βρίσκεται στη "ρίζα" του *app*.



Basic Setup

Router options

Οι επιλογές για **Router** (σε *browser*) είναι οι εξής.

- **BrowserRouter**: αντανακλά το *state* του *routing* στο *URL*, μέσω του *History API*. Είναι το πιο κοντινό σε ένα "παραδοσιακό" *Web app*. Χρειάζεται κατάλληλες ρυθμίσεις στο *server* για να λειτουργήσει προβλέψιμα.
- **HashRouter**: αντανακλά το *state* του *routing* μέσω του *hash / fragment* τμήματος του *URL*. Λειτουργεί χωρίς καμία ιδιαίτερη ρύθμιση στο *server*.
- **MemoryRouter**: το *state* του *routing* υπάρχει μόνο στη μνήμη. Σε πιθανό *refresh* θα χαθεί.



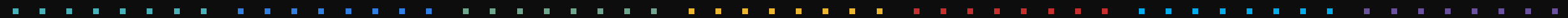
Basic Setup

Routes component

Παρόλο που ο **Router** πρέπει να είναι μοναδικός σε μία εφαρμογή, μπορούν να υπάρχουν περισσότερα του ενός σημεία στα οποία τα **Component** θα εναλλάσσονται, βάσει του *path*.

Το πιο συνηθισμένο, βέβαια, είναι το **Component** αυτό να εμφανίζεται μία φορά, μέσα στο "βασικό" **Component**.

Τα σημεία αυτά ορίζονται με το **Routes component**.





Basic Setup

Route component

Το **Routes component** λειτουργεί σε συνεργασία με το **Route component**.

Με αυτό ορίζονται οι "κανόνες" για το ποιο *element / component* θα εμφανίζεται κάθε φορά.

```
<Route path="[path]" element={ [jsx] } />
```

- **path**: ορίζει το *matching*
- **element**: ορίζει τον **JSX** κώδικα που θα εμφανιστεί μέσα στο **<Routes>**, αν ταιριάζει το **path**.



Basic Setup

Παράδειγμα

```
import { Routes, Route } from 'react-router-dom';

const App = () => (
  <>
    <h1>Welcome to my site!</h1>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/about" element={<About />} />
    </Routes>
  </>
);

export default App;
```



Basic Setup

NoMatch (404)

Το **path property** μπορεί να πάρει και την τιμή *****.

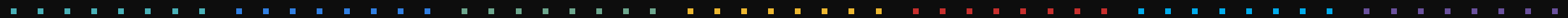
Αυτή η τιμή είναι χρήσιμη στο τελευταίο **Route component**, για να εμφανίσει κάποιο **element** όταν δεν βρεθεί κάποιο **path**.

```

<Routes>
  <Route path="/" element={<Home />} />
  ...
  <Route path="*" element={<NoMatch />} />
</Routes>
  
```



NAVIGATION





Navigation

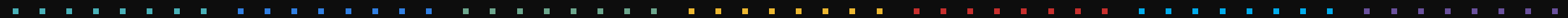
Link component

Για να λειτουργήσει σωστά το *routing*, θα πρέπει να αντικαταστήσουμε τα στοιχεία `<a>` με το *Link component* που μας παρέχει η **React Router**.

```
<Link to="[path]">[Link text]</Link>
```

Υπάρχει περίπτωση, αν συνεχίσουμε τα χρησιμοποιούμε στοιχεία `<a>`, η εφαρμογή να λειτουργεί φαινομενικά σωστά.

Αλλά, ένα στοιχείο `<a>` θα προκαλεί *refresh* της σελίδας, που, για πολλούς λόγους, δεν είναι επιθυμητό.





Navigation

Παράδειγμα

```
import { Link } from 'react-router-dom';

const Menu = () => (
  <nav>
    <Link to="/">Home</Link>
    <Link to="/about">About</Link>
  </nav>
);

export default Menu;
```



Navigation

NavLink component

Πολύ συχνά, υπάρχει η ανάγκη να εφαρμόσουμε κάποιο συγκεκριμένο στυλ στα *link* που αναφέρονται στο "ενεργό" *route*.

Αυτό μπορεί να επιτευχθεί με το **NavLink component**, όπου στο **className property**, μπορεί να γίνει χρήση μιας **function** που καθορίζει μια κλάση, βάσει του αν το συγκεκριμένο *route* είναι ενεργό ή όχι.

```
<NavLink
  to="[path]"
  className={({ isActive }) => isActive ? [active-class-name] : ''}
>
  [Link text]
</NavLink>
```



Navigation

NavLink component

Το ίδιο μπορεί να γίνει και στο **style property**.

```
<NavLink
  to="[path]"
  style={({ isActive }) => ({
    css-property-1: isActive ? [value-1] : [value-2],
    css-property-2: isActive ? [value-3] : [value-4],
    ...
  })}
>
  [Link text]
</NavLink>
```



Navigation

useNavigate

Εκτός των **Link** & **NavLink** *component*, υπάρχει η δυνατότητα "πλοήγησης", είτε με κάποιο άλλο στοιχείο, είτε με *imperative* τρόπο.

Για το σκοπό αυτό, υπάρχει το **useNavigate** *hook*, το οποίο μας παρέχει μια συνάρτηση η οποία μπορεί να κληθεί με ένα *path* ή με μια αριθμητική τιμή, π.χ. με *-1* για προσομοίωση του *back button*.

```
import { useNavigate } from 'react-router-dom';  
...  
const navigate = useNavigate();
```



Navigation

Navigate

Ίδια λειτουργικότητα μπορεί να επιτευχθεί και με *declarative* τρόπο, με τη βοήθεια του **Navigate** *component*.

```
import { useState, useEffect } from 'react';
import { Navigate } from 'react-router-dom';

const LoginForm = () => {
  const { user, setUser } = useState();
  ...
  return (
    <>
      {user && (
        <Navigate to="/dashboard" replace={ true } />
      )}
      ...
    </>
  );
}
```



Navigation

Παράδειγμα

```
import { useNavigate } from 'react-router-dom';
...
export default function Invoice({ invoice }) {
  let navigate = useNavigate();

  return (
    <h2>Total Due: {invoice.amount}</h2>
    <p>
      {invoice.name}: {invoice.number}
    </p>
    <p>Due Date: {invoice.due}</p>
    <p>
      <button
        onClick={() => {
          deleteInvoice(invoice.number);
          navigate('/invoices');
        }}
      >
        Delete
      </button>
    </p>
  );
}
```



Navigation

useParams

Το **path** σε ένα **Route component** συχνά περιλαμβάνει και κάποιο δυναμικό κομμάτι, ως παράμετρο. Π.χ., το **path** που αφορά στο προφίλ ενός χρήστη ή στις λεπτομέρειες ενός προϊόντος.

Ένα τέτοιο δυναμικό κομμάτι ενός **path** ορίζεται με το πρόθεμα ':', π.χ., **/users/:id** ή **/products/:id**.

Το **Component** μπορεί να έχει πρόσβαση στην παράμετρο αυτή, με το **useParams hook**.

```
import { useParams } from 'react-router-dom';
...
const params = useParams();
```




Navigation

Παράδειγμα

```
// App.jsx

export default App = () => (
  <Routes>
    ...
    <Route path="/product/:id" element={<Product />} />
  </Routes>
);
```

```
// Product.jsx

import { useParams } from 'react-router-dom';

export default Product = () => {
  const { id } = useParams();
  return <h2>Product: #{id}</h2>;
}
```



Navigation

useSearchParams

Εκτός του **useParams** *hook*, υπάρχει και το **useSearchParams** *hook*, το οποίο παρέχει πρόσβαση στις παραμέτρους που περιέχονται στο **search property** του **location object**, π.χ. στο **/search?q=react-router**.

Πέρα από την πρόσβαση, το *hook* αυτό μας παρέχει και τη δυνατότητα **τροποποίησης** του **search property**, π.χ. για να αποτυπώσουμε την κατάσταση των φίλτρων μιας αναζήτησης.

```
import { useSearchParams } from 'react-router-dom';
...
const [searchParams, setSearchParams] = useSearchParams({});
```



Navigation

Παράδειγμα

```
import { useSearchParams } from 'react-router-dom';

export default Filters = () => {
  const [searchParams, setSearchParams] = useSearchParams({});

  return (
    <div className="filters">
      <button type="button" onClick={() => { setSearchParams({}); }}>
        Όλα
      </button>
      <button type="button" onClick={
        () => { setSearchParams({ type: 'pc' }); }
      }>
        PC
      </button>
      ...
    </div>
  );
}
```



Navigation

useLocation

Το *hook* αυτό επιστρέφει το τρέχων **location object**.

Χρήσιμο για *side effects* που πρέπει να εκτελούνται όταν αλλάζει η διεύθυνση.

```
import { useEffect } from 'react';
import { useLocation } from 'react-router-dom';

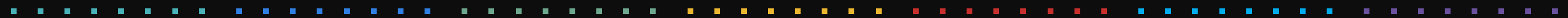
const App = () => {
  const location = useLocation();

  useEffect(() => {
    ga('send', 'pageview');
  }, [location]);

  return ( ... );
};
```



NESTED ROUTES





Nested routes

Nesting

Υπάρχουν πολλοί τρόποι να οργανώσει κανείς τα *routes* μιας εφαρμογής.

Σε αυτό μπορεί να βοηθήσει η δυνατότητα να οριστούν *nested routes*.

Σε ένα *nested route* το *path* προσαρτάται σε αυτό του *parent route*.

Το κύριο *route* σε μια σειρά από *sibling routes* ορίζεται με το *keyword index*.



Nested routes

Παράδειγμα #1

```
import { Route, Routes } from 'react-router-dom';

...

const App = () => (
  <>
    <Routes>
      <Route path="/">
        <Route index element={<Home />} />
        <Route path="about" element={<About />} />
      </Route>
    </Routes>
  </>
);

export default App;
```



Nested routes

Παράδειγμα #2

```
import { Route, Routes } from 'react-router-dom';

...

const App = () => (
  <>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/about" element={<About />} />
      <Route path="/products" >
        <Route index element={<List />} />
        <Route path=":id" element={<Details />} />
      </Route>
    </Routes>
  </>
);

export default App;
```




Nested routes

Outlet component

Αν σε ένα *parent route* δεν έχει οριστεί το **element property**, τότε το **route** αυτό απλά προσφέρει το **path** του ως πρόθεμα στα *child routes*.

Αντίθετα, αν έχει οριστεί το **element property**, τότε θα πρέπει να γίνει χρήση του **Outlet component** σε κάποιο στοιχείο που εμφανίζεται στο **element property**.

Στη θέση του **Outlet component**, εμφανίζεται το περιεχόμενο του **element property** από το *child route*, που έχει "ταιριιάξει" με το τρέχων **path**.

Αν δεν υπάρχει **Outlet component**, τότε το περιεχόμενο των *child routes* δεν θα εμφανιστεί πουθενά.



Nested routes

Παράδειγμα

```
// App.jsx
import { Route, Routes } from 'react-router-dom';

...

const App = () => (
  <>
    <Routes>
      <Route path="/" element={<Layout />}>
        <Route index element={<Home />} />
        <Route path="about" element={<About />} />
      </Route>
    </Routes>
  </>
);

export default App;
```



Nested routes

Παράδειγμα (cont.)

```
// Layout.jsx
...

const Layout = () => (
  <>
    <Header />
    <main>
      <Outlet />
    </main>
    <aside>
      <SideBar />
    </aside>
    <Footer />
  </>
);

export default Layout;
```



Nested routes

Lazy routes

Μιας και όλα τα *components* που αφορούν σε **Routes** δηλώνονται, συνήθως, στο κεντρικό *component*, π.χ., **<App />**, ο κώδικας που περιέχουν θα πρέπει να φορτωθεί εξ' αρχής.

Αν η εφαρμογή έχει αρκετά **Routes**, αυτό μπορεί να αυξάνει το χρόνο απόκρισης, γεγονός μη επιθυμητό.

Η χρήση της μεθόδου **React.lazy** επιτρέπει να φορτώνονται τα *components*, στο πρώτο *mount* και όχι εξ' αρχής.

Πρέπει, όμως, να γίνει χρήση και του **React.Suspense component**, αλλιώς η χρήση του **React.lazy** οδηγεί σε σφάλμα.



Nested routes

Παράδειγμα

```
import React, { Suspense } from 'react';
import { Routes, Route } from 'react-router-dom';

import Layout from './Layout';
import Home from './Home';

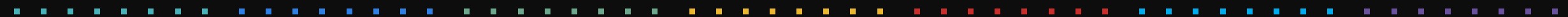
const About = React.lazy(() => import('./About'));

const App = () => (
  <>
    <Routes>
      <Route path="/" element={<Layout />}>
        <Route index element={<Home />} />
        <Route path="about" element={
          <React.Suspense fallback={<>...</>}>
            <About />
          </React.Suspense>
        } />
      </Routes>
    </Routes>
  </>
);

export default App;
```



HOMework





Homework

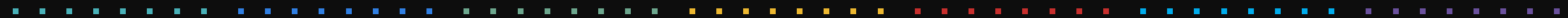
Homework #1

Φτιάξτε ένα **React app** για το "*sitemap*" ενός e-shop. Θα πρέπει να υπάρχουν, κατ' ελάχιστον, τα εξής *routes*:

Home, About, Contact, Product List, Product Details

Χρησιμοποιήστε το **React Router** για την διαχείριση των **Routes** και ένα **Layout component** για τη βασική δομή.

Προαιρετικά, ορίστε κάποια **Routes** ως *lazy*.





Χρήσιμα links

› React Router | Installation

<https://reactrouter.com/docs/en/v6/getting-started/ins...>

› React Router | Overview

<https://reactrouter.com/docs/en/v6/getting-started/ov...>

› React Router | Tutorial

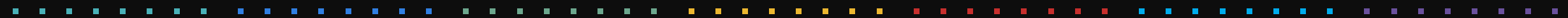
<https://reactrouter.com/docs/en/v6/getting-started/tut...>

DEV Create a React App with React Router Dom v6 - DEV...

<https://dev.to/salehmubashar/react-router-dom-36a2>

› React Router 6 Tutorial

<https://www.robinwieruch.de/react-router/>





Extra info

📄 How to Pass Props Through React Router's Link Comp...
<https://ui.dev/react-router-pass-props-to-link/>

📄 What's New in React Router 6?. A Quick Overview of ...
<https://enlear.academy/whats-new-in-react-router-6-e...>

📺 React Router 6 - What Changed & Upgrading Guide ...
<https://www.youtube.com/watch?v=zEQiNFAwDGo>

📄 Amazing New Features In React & React Router v6 | ...
<https://medium.com/age-of-awareness/amazing-new-...>

📄 Code-Splitting – React
<https://reactjs.org/docs/code-splitting.html>



THANK YOU!