

# Final Module

## Lec7

## נושאים למימוש בפרוייקט

1. התחברות
2. התנתקות
3. הסבר על קומפוננטה עוטפת ועל CHILDREN
4. מימוש אמיתי של נתיב שאי אפשר להגיע אליו בלי להתחבר
5. עבודה עם קבצי סביבה
6. גישה למשאבים מוגנים עם JWT

## קומפוננטה להתחברות – המשך בעמוד הבא:

### Routes/Login.tsx

```
import { ErrorMessage, Field, Form, Formik } from "formik";
import { useContext, useState } from "react";
import * as Yup from "yup";
import Spinner from "../components/Spinner";
import { showErrorDialog, showSuccessDialog } from "../dialogs/dialogs";
import { auth } from "../services/auth-service";
import { useNavigate } from "react-router-dom";
import { AuthContext } from "../contexts/AuthContext";

const Login = () => {
  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState<string>();
  const { login } = useContext(AuthContext);

  const navigate = useNavigate();

  const validationSchema = Yup.object({
    email: Yup.string().email("Bad Email!").required("The email is required"),

    password: Yup.string()
      .required()
      .min(8)
      .max(20)
      .matches(
        /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$$%^&*_-]).{8,30}$/ ,
        "Password must contain at least one uppercase letter, one lowercase letter, one number and one special character"
      )
  });
};
```

```
return (  
  <Formik  
    initialValues={initialValues}  
    validationSchema={validationSchema}  
    onSubmit={(o) => {  
      setIsLoading(true);  
      auth  
        .login(o.email, o.password)  
        .then((response) => {  
          showSuccessDialog("Login Successful").then(() => {  
            login(response.data.token);  
            navigate("/");  
          });  
        })  
        .catch((error) => {  
          showErrorDialog("Login Failed");  
        })  
        .finally(() => {  
          setIsLoading(false);  
        });  
    })  
  </Formik>  
>
```

```

<Form className="flex flex-col items-center">
  {isLoading && <Spinner title="WaitUp!" />}
  {error && <p className="text-red-500">{error}</p>}}

  <div className="font-extralight form-group flex flex-col gap-2 w-1/2 mx-auto text-lg my-4">
    <label htmlFor="email">Email Address</label>
    <Field
      name="email"
      type="email"
      id="email"
      className="rounded-md hover:border-2 border-2 px-2 py-2"
    />
    <ErrorMessage name="email" component="div" className="text-red-500" />
  </div>

  <div className="font-extralight form-group flex flex-col gap-2 w-1/2 mx-auto text-lg my-4">
    <label htmlFor="password">Password</label>
    <Field
      name="password"
      type="password"
      id="password"
      className="rounded-md hover:border-2 border-2 px-2 py-2"
    />
    <ErrorMessage
      name="password"
      component="div"
      className="text-red-500"
    />
  </div>

  <button
    disabled={isLoading}
    type="submit"
    className="bg-blue-500 disabled:bg-blue-500/50 w-1/2 block text-left hover:bg-blue-700 text-white font-bold py-2 px-4 rounded my-4"
  >
    Login
  </button>
</Form>
</Formik>
);
};
export default Login;

```

## Routes/Login.tsx

קומפוננטה  
להתחברות – המשך  
מהעמוד הקודם:

## Contexts/AuthContex.tsx

```
import { createContext, useState } from "react";

interface AuthContextType {
  isLoggedIn: boolean;
  token: string;
  login: (token: string) => void;
  logout: () => void;
}

const initialValues: AuthContextType = {
  isLoggedIn: !!localStorage.getItem("token"),
  token: localStorage.getItem("token") || "",
  login: () => {},
  logout: () => {},
};

const AuthContext = createContext(initialValues);

function AuthProvider(props) {
  const [isLoggedIn, setIsLoggedIn] = useState(!!localStorage.getItem("token"));
  const [token, setToken] = useState(localStorage.getItem("token") || "");

  function login(token: string) {
    setIsLoggedIn(true);
    localStorage.setItem("token", token);
    setToken(token);
  }

  function logout() {
    setIsLoggedIn(false);
    setToken("");
    localStorage.removeItem("token");
  }

  return (
    <AuthContext.Provider value={{ isLoggedIn, token, login, logout }}>
      {props.children}
    </AuthContext.Provider>
  );
}

export { AuthProvider, AuthContext };
```

שימוש ב LocalStorage לשימור מצב  
גם כשהמשתמש טוען מחדש את  
האפליקציה

Hook

לשימוש נוח יותר בContext

Hooks/useAuth.tsx

```
import { useContext } from "react";
import { AuthContext } from "../contexts/AuthContext";

export const useAuth = () => {
  const context = useContext(AuthContext);
  if (!context) {
    throw new Error('useAuthContext must be used within an AuthProvider');
  }
  return context;
};
```

דוגמא לשימוש

במקום לכתוב:

```
const authContext = useContext(AuthContext);
```

נוכל לכתוב:

```
const authContext = useAuth();
```

## Hook לשימוש נוח יותר בContext

Hooks/useDarkMode.tsx

```
import { useContext } from "react";
import { DarkModeContext } from "../contexts/DarkModeContext";

const useDarkMode = () => {
  const context = useContext(DarkModeContext);

  if (!context) {
    throw new Error("useDarkMode must be used within a DarkModeProvider");
  }

  return context;
};

export default useDarkMode;
```



## הסבר על קומפוננטה עם Children ועל הטיפוס של הקומפוננטה בTypescript

```
import { ReactNode } from "react";
```

```
type CardProps = {  
  title: string;  
  children: ReactNode;  
}
```

Typescript מאפשרת  
לנו לציין את הטיפוס של  
הפרמטרים שפונקציה  
מקבלת  
כך נקבל השלמה  
אוטומטית וtype safety

```
const Card = (props: CardProps) => {  
  return (  
    <div className="bg-white p-6 shadow-lg rounded-lg">  
      <h2 className="text-xl font-semibold mb-2">{props.title}</h2>  
      <div>{props.children}</div>  
    </div>  
  );  
};  
  
export default Card;
```

## הסבר על קומפוננטה עם Children ועל הטיפוס של הקומפוננטה בTypescript

```
import { ReactNode } from "react";
```

```
type CardProps = {  
  title: string;  
  children: ReactNode;  
}
```

Typescript מאפשרת  
לנו לציין את הטיפוס של  
פונקציה כולל טיפוס  
לארגומנטים וגם הטיפוס  
המוחזר

```
type CardType = (props: CardProps) => ReactNode;
```

```
const Card: CardType = (props) => {  
  return (  
    <div className="bg-white p-6 shadow-lg rounded-lg">  
      <h2 className="text-xl font-semibold mb-2">{props.title}</h2>  
      <div>{props.children}</div>  
    </div>  
  );  
};  
  
export default Card;
```

## Object Destructuring

```
import { ReactNode } from "react";
```

```
type CardProps = {  
  title: string;  
  children: ReactNode;  
}
```

```
type CardType = (props: CardProps) => ReactNode;
```

```
const Card: CardType = ({title, children}) => {  
  return (  
    <div className="bg-white p-6 shadow-lg rounded-lg">  
      <h2 className="text-xl font-semibold mb-2">{title}</h2>  
      <div>{children}</div>  
    </div>  
  );  
};
```

```
export default Card;
```

הפונקציה שלנו מקבלת  
אובייקט של props  
לשם נוחות אפשר לחלץ  
מהאובייקט את התכונות  
title, children

במקום props.title

נכתוב title  
(בזכות החילוץ של  
title מהprops)

## סידור מחדש של הקוד לטובת שימוש חוזר:

```
type CardProps = {  
  title: string;  
  children: ReactNode;  
}
```

```
type CardType = (props: CardProps) => ReactNode;
```

```
type FC<T> = (props: T) => ReactNode;
```

הגדרת טיפוס גנרי  
לפונקציה שמקבלת  
props ומחזירה  
ReactNode

הטיפוס של props  
יכול להיות שונה בין  
קומפוננטה לקומפוננטה

ולכן נגדיר אותו כ-`<T>`

## סידור מחדש של הקוד לטובת שימוש חוזר:

לטובת שימוש חוזר בקוד – נכתוב את זה בקובץ חדש שנוכל לייבא אחר כך מכל מקום בפרוייקט:

src/@types.d.ts

```
import { ReactNode } from "react";
```

```
//FC = FunctionComponent
```

```
export type FC<T> = (props: T) => ReactNode;
```

הגדרת טיפוס גנרי  
לפונקציה שמקבלת  
props ומחזירה  
ReactNode

הטיפוס של props  
יכול להיות שונה בין  
קומפוננטה לקומפוננטה

ולכן נגדיר אותו כ<T>

## סידור מחדש של הקוד לטובת שימוש חוזר:

לטובת שימוש חוזר בקוד – נכתוב את זה בקובץ חדש שנוכל לייבא אחר כך מכל מקום בפרוייקט:

src/components/Card.tsx

```
import { ReactNode } from "react";  
import { FC } from "../@types";
```

```
type CardProps = {  
  title: string;  
  children: ReactNode;  
}
```

```
const Card:FC<CardProps> = ({title, children}) => {  
  return (  
    <div className="bg-white p-6 shadow-lg rounded-lg">  
      <h2 className="text-xl font-semibold mb-2">{title}</h2>  
      <div>{children}</div>  
    </div>  
  );  
};
```

```
export default Card;
```

שימוש בטיפוס FC<T>

לשם נוחיות אפשר להגדיר גם טיפוס לפונקציה שמקבלת ב props רק children מסוג ReactNode

```
// a react component is a function that takes props and returns a ReactNode
```

```
import { ReactNode } from "react";
```

```
//FC = FunctionComponent
```

```
export type FC<T> = (props: T) => ReactNode;
```

```
export type FCP = FC<{ children: ReactNode }>;
```

העברה אוטומטית של לקוח מחובר – שלא יוכל לצפות בעמוד התחברות

Components/NoAuthRoute.tsx

```
import { Navigate } from "react-router-dom";
import { FCP } from "../@types";
import useAuth from "../hooks/useAuth";

const NoAuthRoute: FCP = ({ children }) => {
  const { isLoggedIn } = useAuth();

  if (isLoggedIn) {
    return <Navigate to="/" />;
  }
  return children;
};

export default NoAuthRoute;
```



## שימוש בקובץ App.tsx

App.tsx

```
<Route path="/register" element={<NoAuthRoute><Register /></NoAuthRoute>} />  
<Route path="/login" element={<NoAuthRoute><Login /></NoAuthRoute>} />
```

כך משתמש מחובר לא יכול להגיע לעמוד לוגין – זה אוטומטית יעביר אותו לעמוד הבית

## העברה אוטומטית של לקוח לא מחובר – לעמוד התחברות

### Components/ProtectedRoute.tsx

```
import { FC, ReactNode } from "react";
import useAuth from "../hooks/useAuth";
import { Navigate } from "react-router-dom";
import { FCP } from "../@types";

const ProtectedRoute: FCP = ({children}) => {
  const { isLoggedIn } = useAuth();

  if (!isLoggedIn) {
    return <Navigate to="/login" />;
  }

  return children;
};

export default ProtectedRoute;
```

## שימוש בקובץ App.tsx

### App.tsx

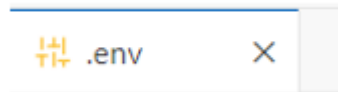
```
const App = () => {  
  return (  
    <>  
      <Navbar />  
      <Routes>  
        <Route path="/" element={<Home />} />  
        <Route path="/about" element={<About />} />  
        <Route path="/register" element={<NoAuthRoute><Register /></NoAuthRoute>} />  
        <Route path="/login" element={<NoAuthRoute><Login /></NoAuthRoute>} />  
        <Route path="/products" element={<ProtectedRoute><Products /></ProtectedRoute>} />  
  
        <Route path="*" element={<NotFound />} />  
      </Routes>  
    </>  
  );  
};  
  
export default App;
```

כך משתמש לא מחובר ישלח לעמוד ההתחברות אם הוא מנסה לגשת למשאב מוגן

## שימוש במשתני סביבה:

בתיקה הראשית של הפרוייקט (מחוץ לתיקית src) ניצור קובץ בשם .env.

Vite תומכת בקבצים הללו באופן אוטומטי ולכן שם הקובץ חשוב.



```
VITE_BASE_URL=https://localhost:7037/api  
VITE_MODE=development
```

בדיקה שזה עובד:


```
const App = () => {  
  const apiUrl = import.meta.env.VITE_BASE_URL;  
  console.log(apiUrl);  
}
```

כך נשתמש  
במשתני סביבה:

עבודה עם משתני סביבה:


אפשר ליצור מספר קבצים למשתני סביבה:


 .env.production X  package.json

 .env.production  
VITE\_BASE\_URL=https://localhost:7037/api  
VITE\_MODE=production

ניצור עוד קובץ .env.production


1

 .env.production

 package.json X

נערוך את הקובץ package.json

2

 package.json > ...

```
1  {  
2    "name": "frontend",  
3    "private": true,  
4    "version": "0.0.0",  
5    "type": "module",  
6    "scripts": {  
7      "dev": "vite",  
8      "prod": "vite --mode production",  
9      "build": "tsc -b && vite build",  
10     "lint": "eslint . --ext ts,tsx --report-unused-disable-directives --max-warnings 0",  
11     "preview": "vite preview"  
12  },
```

נוסיף סקריפט להרצת התוכנית במצב production:

נריץ את התוכנית עם  
npm run prod

3

4

```
const App = () => {  
  const env = import.meta.env.VITE_ENV;  
  console.log(env);  
}
```

בדיקה

5

## שליחת בקשות עם JWT

services/products-service.ts

```
import axios from "axios";
```

ייבוא של הכתובת מקובץ המשתנים .env

```
const url = import.meta.env.VITE_BASE_URL + "/products";
```

```
export const getProducts = (jwt: string) => {  
  return axios.get(url, {  
    headers: {  
      Authorization: `Bearer ${jwt}`,  
    },  
  });  
};
```

פונקציה שמקבלת jwt  
כפרמטר

שליחת בקשות עם JWT ב axios

שליחת בקשות עם JWT

הקומפוננטה products

```
import useAuth from "../hooks/useAuth";
import { getProducts } from "../services/products-service";

const Products = () => {
  const { token } = useAuth();
  getProducts(token)
    .then((response) => {
      console.log(response.data);
    })
    .catch((error) => {
      console.error(error);
    });
  return <div>Products</div>;
};

export default Products;
```