

# Frontend With React

עבודה על פרוייקט צד לקוח

TomerBu

# התקנת ספריות

תיאור	התקנה
ספריה להצגת "התקדמות/טעינה"	<code>npm i react-loader-spinner</code>
ספריה להצגת דיאלוגים מודליים	<code>npm i sweetalert2</code>
ספריה לבקשות http	<code>npm i axios</code>

פקודה אחת להתקנת כל הספריות:

```
npm i axios react-loader-spinner sweetalert2
```

# יישום DarkMode

tailwind.config.js

מגדיר לספריה לעבור למצב dark אם יש class="dark" על האלמנט של הbody (ולהיפך)

```
export default {  
  darkMode: "class",  
  content: ["../index.html", "./src/**/*.{js,ts,jsx,tsx}"],  
  theme: {  
    extend: {},  
  },  
  plugins: [],  
};
```

# יישום DarkMode

contexts/DarkModeContext.tsx

```
function toggle() {  
  //calculate the new state as a string: "dark" or "light":  
  const newMode = !darkMode ? "dark" : "light";  
  
  //save the new state to local storage:  
  localStorage.setItem("darkMode", newMode);  
  
  setDarkMode((prev) => !prev);  
  
  document.body.classList.toggle("dark");  
}
```

בפונקציה שמגיבה לכפתור שינוי מצב: נבדוק מה המצב החדש (ההיפך מהמצב הקודם בעת הלחיצה)

נשמור את המצב החדש בlocalstorage

נעדכן את הclass בbody בהתאם למצב החדש - הוספת class="dark" או הסרת class="dark"

# יישום DarkMode

contexts/DarkModeContext.tsx

```
function toggle() {  
  //calculate the new state as a string: "dark" or "light":  
  const newMode = !darkMode ? "dark" : "light";  
  
  //save the new state to local storage:  
  localStorage.setItem("darkMode", newMode);  
  
  setDarkMode((prev) => !prev);  
  
  document.body.classList.toggle("dark");  
}
```

בפונקציה שמגיבה לכפתור שינוי מצב: נבדוק מה המצב החדש (ההיפך מהמצב הקודם בעת הלחיצה)

נשמור את המצב החדש בlocalstorage

נעדכן את הclass בbody בהתאם למצב החדש - הוספת class="dark" או הסרת class="dark"

# יישום DarkMode

contexts/DarkModeContext.tsx

```
function DarkModeProvider({ children }) {  
  
  useEffect(() => {  
    const mode = localStorage.getItem("darkMode");  
    if (mode === "dark") {  
      setDarkMode(true);  
      document.body.classList.toggle("dark");  
    }  
  }, []);  
  
  const [darkMode, setDarkMode] = useState(false);  
  
  //functions:  
  function toggle() {  
    //calculate the new state as a string: "dark" or "light":  
    const newMode = !darkMode ? "dark" : "light";  
  
    //save the new state to local storage:  
    localStorage.setItem("darkMode", newMode);  
  
    setDarkMode((prev) => !prev);  
  
    document.body.classList.toggle("dark");  
  }  
  
  return (  
    <DarkModeContext.Provider value={{ darkMode, toggle }}>  
      {children}  
    </DarkModeContext.Provider>  
  );  
}
```

useEffect מאפשר לנו לכתוב קוד שרץ פעם אחת כשהקומפוננטה עולה

useEffect(()=>{//code that runs on Mount}, []);

כשהקומפוננטה עולה - נבדוק מה שמרנו בlocalStorage  
אם שמור dark:

א) נעדכן את משתנה המצב darkMode לtrue  
ב) נעדכן את הclass בbody לdark

# יישום DarkMode

היישום של darkmode עם הספרייה :tailwind

```
<nav  
  id="app-nav"  
  className="shadow-2xl p-8 flex gap-3 bg-fuchsia-50 text-fuchsia-900 dark:bg-fuchsia-900 dark:text-fuchsia-50"  
>
```

צבע רקע למצב רגיל

צבע רקע למצב dark

כלומר אפשר להגדיר css אחד למצב הרגיל וcss שונה למצב dark  
וזה תקף לגבי כל חוק css שנרצה

לדוגמא:

```
class="p-2 dark:p-10"
```

יוסיף יותר padding במצב dark

## יצירת Context לשמירת מצב ההתחברות וביצוע התנתקות

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

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

```
const initialValues:AuthContextType = {  
  isLoggedIn: false,  
  token: "",  
  login: () => {},  
  logout: () => {},  
};
```

```
const AuthContext = createContext(initialValues);
```

```
function AuthProvider(props) {
```

```
  const [isLoggedIn, setIsLoggedIn] = useState(false);  
  const [token, setToken] = useState("");
```

```
  function login(token: string) {  
    setIsLoggedIn(true);  
    setToken(token);  
  }
```

```
  function logout() {  
    setIsLoggedIn(false);  
    setToken("");  
  }
```

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

```
export { AuthProvider, AuthContext };
```

הגדרת טיפוסים של הContext שלנו:  
(שלב אופציונלי - הגדרות typescript)

הגדרת ערכים התחלתיים (מבנה של הContext)

משתני מצב שאנחנו רוצים לחלוק עם כל הקומפוננטות באפליקציה  
(זה מה שcontext מאפשר)

פונקציות לשינוי משתני המצב

הספריה מספקת לנו קומפוננה עוטפת  
שמנגישה את הcontext לקומפוננטות בת  
שלה (Children)

הקומפוננטה שלנו מחזירה את הקומפוננטה  
העוטפת שיוצרת עבורנו הספריה.



# הנגשה של AuthContext לכל האפליקציה:

main.tsx

```
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { BrowserRouter } from "react-router-dom";
import { DarkModeContextWrapper } from "../contexts/DarkModeContext";
import { AuthContextProvider } from "../contexts/AuthContext";
```

```
const root = ReactDOM.createRoot(
  document.getElementById("root") as HTMLElement
);
```

```
root.render(
```

```
  <BrowserRouter>
```

```
    <AuthContextProvider>
```

```
      <DarkModeContextWrapper>
```

```
        <App />
```

```
      </DarkModeContextWrapper>
```

```
    </AuthContextProvider>
```

```
  </BrowserRouter>
```

```
);
```

```
reportWebVitals();
```

כאן רואים שמדובר בקומפוננטה עוטפת  
כי יש לה ילדים:

הילדים הם מה שהיא עוטפת:  
DarkModeContextWrapper  
App

## בסרגל הניווט - הצגה בתנאי - אם מחובר - נציג קישור לעמוד products אם לא מחובר - נציג קישור להתחברות/הרשמה

components/Navbar.tsx

```
import { NavLink } from "react-router-dom";
import { FaBloggerB } from "react-icons/fa";
import { AiFillGithub } from "react-icons/ai";
import { BsMoonFill, BsSunFill } from "react-icons/bs";
import "./Navbar.scss";
import { useContext } from "react";
import DarkModeContext from "../../contexts/DarkModeContext";
import AuthContext from "../../contexts/AuthContext";
const Navbar = () => {

  const {isLoggedIn} = useContext(AuthContext);
  const { darkMode, toggleDarkMode } = useContext(DarkModeContext);
  return (
    <nav
      id="app-nav"
      className="sm:gap-10 shadow-lg p-8 gap-4 flex align-middle items-center justify-center bg-fuchsia-50 text-fuchsia-900 dark:bg-fuchsia-900 dark:text-fuchsia-50"
    >
      <NavLink to="/home">
        <FaBloggerB />
      </NavLink>

      <NavLink to="/about">About</NavLink>
      <div className="flex-1"></div>
      <div className="hidden sm:flex sm:items-center sm:gap-10">
        {isLoggedIn && <NavLink to="/products">Products</NavLink>}
        {!isLoggedIn && <NavLink to="/login">Login</NavLink>}
        {!isLoggedIn && <NavLink to="/register">Register</NavLink>}
        <a href="https://github.com/">
          <AiFillGithub />
        </a>
        <button
```

# בסרגל הניווט - הוספת כפתור התנתקות

components/Navbar.tsx

```
import { BiLogout } from "react-icons/bi";
```

```
const Navbar = () => {
```

```
  const { isLoggedIn, logout } = useContext(AuthContext);
```

```
  {isLoggedIn && (  
    <button onClick={logout} className="rounded-lg p-2">  
      <BiLogout aria-description="Logout" />  
    </button>  
  )}
```

```
  </div>
```

```
</nav>
```

```
);
```


```
};
```

```
export default Navbar;
```

# ספרייה להצגת Spinners

התקנה:

```
npm i react-loader-spinner
```

 **React Spinners**

Documentation

Getting Started

Components

Radio

Audio

Ball Triangle

Bars

Blocks

Circles With Bar

Circles

Color Ring

Comment

Discuss

DNA

Falling Lines

Fidget Spinner

Grid


Hearts

Infinity Spin

[⬅](#) > Components > **Circles With Bar**

## Circles With Bar

RESULT



LIVE EDITOR

```
<CirclesWithBar
  height="100"
  width="100"
  color="#4fa94d"
  wrapperStyle={{}}
  wrapperClass=""
  visible={true}
  outerCircleColor=""
  innerCircleColor=""
  barColor=""
  ariaLabel='circles-with-bar-loading'
/>
```

<https://mhnpd.github.io/react-loader-spinner/docs/components/circles-with-bar>

# קומפוננטה להצגת Spinner (מצב טעינה) שימוש בספרייה react-loader-spinner

components/Spinner.tsx

```
import { CirclesWithBar } from "react-loader-spinner";
```

```
const Spinner = (props) => {
```

כל קומפוננטה בראקט יכולה לקבלת פרמטרים מבחון

```
  return (
```

לדוגמא - title - כותרת שאפשר להציג בקומפוננטת הטעינה

```
    <>
```

```
      <p className="text-center">{props.title ?? "Loading please wait"}</p>
```

```
      <CirclesWithBar
```

```
        wrapperClass="flex justify-center items-center"
```

```
        height={100}
```

```
        color="#ffdd00"
```

```
        outerCircleColor="#ff00ff"
```

```
        innerCircleColor="#dd00dd"
```

```
        barColor="#dd88dd"
```

```
      />
```

הצגת אנימצית טעינה

```
    </>
```

```
  );
```

```
};
```

```
export default Spinner;
```

# הצגת הSpinner במצב טעינה בעמוד Register

routes/Register.tsx

```
import Spinner from "../components/Spinner/Spinner";
```

```
<Form>  
  {loading && <Spinner />}
```

```
<Formik  
  initialValues={initialValues}  
  validationSchema={validationSchema}  
  onSubmit={(o) => {  
    setIsLoading(true);  
  }}  
>
```

## הצגת הודעת שגיאה:

השגיאה תהיה מוצגת רק אם error לא יהיה undefined

```
const [error, setError] = useState<string>();
```

<Form>

```
{error && <p className="text-red-500 flex justify-center w-fit mx-auto px-10 py-5 mt-4 rounded-3xl italic shadow-md">{error}</p>}
```

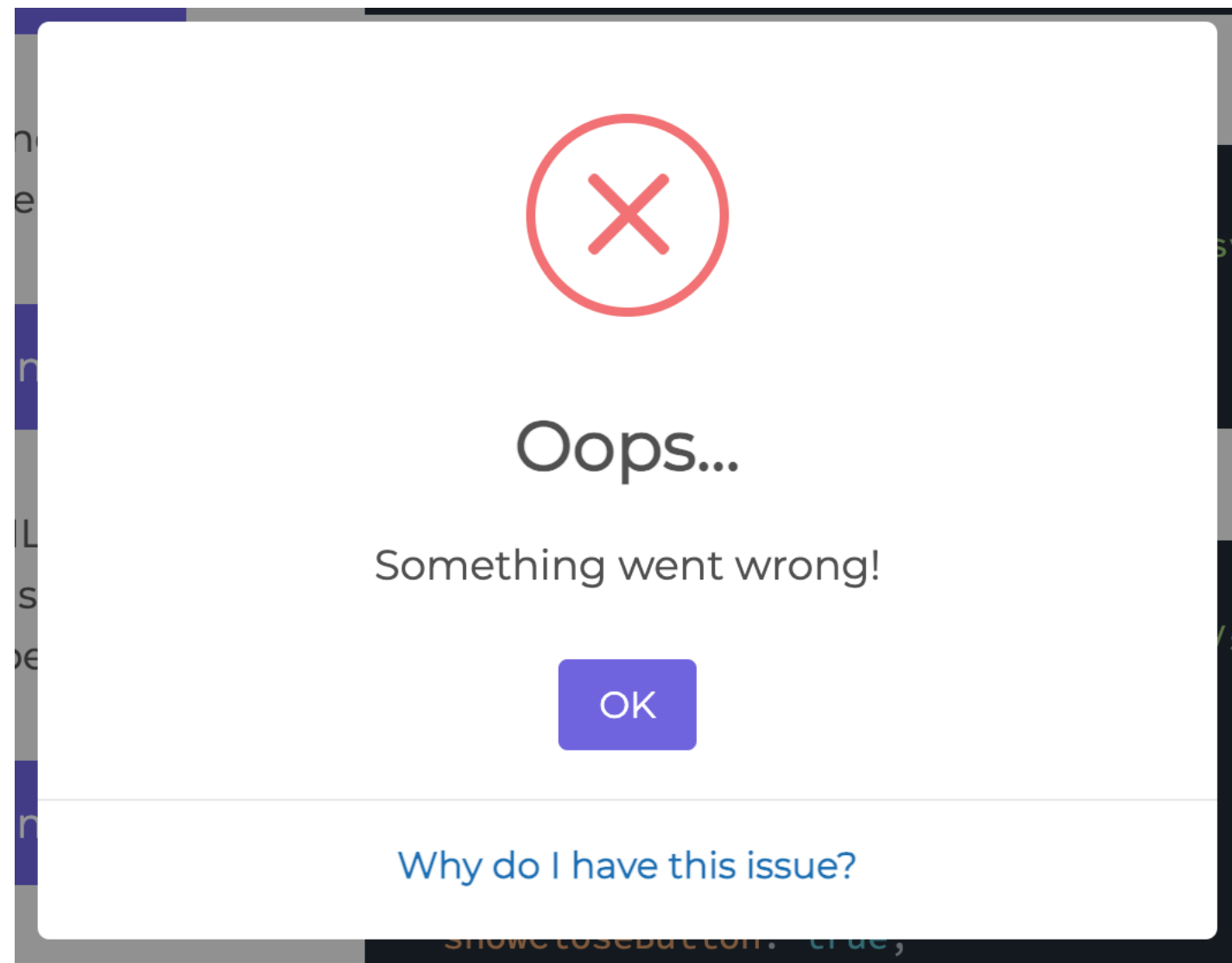
<Formik

```
  initialValues={initialValues}  
  validationSchema={validationSchema}  
  onSubmit={(o) => {  
    setError("Something went wrong");  
  }}  
>
```

# הצגת דיאלוג של הצלחה/כשלון: שימוש בספריה SweetAlert2

<https://sweetalert2.github.io/#examples>

`npm i sweetalert2`





# הצגת דיאלוג של הצלחה/כשלון: שימוש בספריה SweetAlert2

src/dialogs/dialogs.ts

```
import Swal from "sweetalert2";
```

```
const showSuccessDialog = (message) =>
```

```
  Swal.fire({  
    icon: "success",  
    title: "Success",  
    text: message,  
    timer: 2000,  
  });
```

הפונקציה שלנו: showSuccessDialog מחזירה promise  
וכך ניתן לדעת מתי הדיאלוג הסתיים (אינו מוצג)

כלומר אפשר לשרשר then כדי להפעיל קוד שכהדיאלוג מוסר מהמסך

(הפונקציה Swal.fire מחזירה promise)

והפונקציה שלנו showSuccessDialog

מחזירה את Swal.Fire כלומר מחזירה את אותו הpromise

```
const showErrorDialog = (message) =>
```

```
  Swal.fire({  
    icon: "error",  
    title: "Error",  
    text: message,  
    timer: 2000,  
  });
```

```
export { showSuccessDialog, showErrorDialog };
```

```
export const dialog = { success: showSuccessDialog, error: showErrorDialog };
```

# Auth Service

## קובץ לבקשות HTTP עבור התחברות / הרשמה / התנתקות

ספרייה נחמדה לעבודה עם http:

npm i axios

ביצוע בקשת http עם axios:

```
import axios from "axios";

const baseUrl = "https://localhost:7037/api/auth";

const register = (email: string, username: string, password: string) =>
  axios.post(`${baseUrl}/register`, { email, username, password });

//after successful login, the server will return a token
//and we will store it in the local storage
const login = (email: string, password: string) =>
  axios.post(`${baseUrl}/login`, { email, password }).then((response) => {
    if (response.data.token) {
      localStorage.setItem("token", JSON.stringify(response.data));
    }
    return response;
  });

export { register, login };

export const auth = { register, login };
```

# תגובה ללחיצה על כפתור Submit

```
import auth from "services/auth-service";
```

```
import dialog from "dialogs/dialogs";
```

```
<Formik
  validationSchema={validationSchema}
  initialValues={initialValues}
  onSubmit={({ username, email, password }) => {
    setLoading(true); //show progress spinner
    auth
      .register(username, email, password)
      .then((res) => {
        showSuccessDialog("Registered Successfully");
      })
      .catch((e) => {
        showErrorDialog("Something went wrong...");
      })
      .finally(() => {
        setLoading(false);
      });
  }}
>
```

# ביטול כפתור השליחה - כל עוד יש בקשה יוצאת:

```
<button
  disabled={loading}
  className="disabled:bg-fuchsia-700/50 rounded text-white px-3 py-2 w-full bg-fuchsia-700"
  >
  Register
</button>
```

# ניווט לעמוד התחברות - ברגע שהמשתמש נרשם בהצלחה

```
import { useNavigate } from "react-router-dom";

const Register = () => {
  const nav = useNavigate();
```

```
authService
  .register(username, email, password)
  .then((res) => {
    showSuccessDialog(
      "Registered successfully"
    );
    //navigate
    nav("/login");
  })
  .catch((e) => {
    console.log(e.response.data);
    showErrorDialog("Something went wrong...");
  })
  .finally(() => {
    setLoading(false);
  });
```

# קיצורי מקשים:

alt shift f

פרמוט המסמך

ctrl + d

מסמנים ביטוי או שורה עם העכבר ואז לוחצים  
ctrl+d

זה מביא לסימון של הערך הזהה הבא במסמך

alt ↓

הורדת שורה למטה

alt shift ↓

שכפול שורה

ctrl + p

חיפוש של קובץ בפרוייקט

ctrl .

quick fix

(מקביל לalt enter)

ctrl + shift + p

חיפוש של הגדרות או תוספים בvscode