

Frontend With React

עבודה על פרוייקט צד לקוח

TomerBu

עבודה עם Tailwind css

Css framework

<https://tailwindcss.com/docs/guides/vite>

```
npm install -D tailwindcss postcss autoprefixer
```

התקנה של הספרייה:
-D מתקין את הספרייה
כDev Dependency

1

```
npx tailwindcss init -p
```

יוצר קובץ בשם tailwind.config.js

קובץ הגדרות לספרייה

2

עבודה עם Tailwind css

Css framework

 `tailwind.config.js` U X

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}

// הגדרות לספרייה
```

עבודה עם Tailwind css

התקנה של 3 רכיבים מהספרייה:

index.css

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```


4

תוסף להשלמה אוטומטית:

EXTENSIONS: MARKETPLACE


tailwind


5




Tailwind CSS IntelliSense

Intelligent Tailwind CSS tooling for VS Code

 Tailwind Labs

 52ms



עבודה עם Tailwind

```
import { NavLink } from "react-router-dom";
const Navbar = () => {
  return (
    <nav className="shadow-2xl p-8 gap-5 flex bg-fuchsia-900 text-fuchsia-50">
      <NavLink className="border-white border-2 p-2 rounded-lg" to="/home">Home</NavLink>
      <NavLink className="border-white border-2 p-2 rounded-lg" to="/about">About</NavLink>
      <NavLink className="border-white border-2 p-2 rounded-lg" to="/products">Products</NavLink>
      <NavLink className="border-white border-2 p-2 rounded-lg" to="/login">Login</NavLink>
      <NavLink className="border-white border-2 p-2 rounded-lg" to="/register">Register</NavLink>
    </nav>
  );
};

export default Navbar;
```

CSS
עבור מובייל

עבודה עם Tailwind - עיצוב למסכים שונים ומצב לילה

```
import { NavLink } from "react-router-dom";
const Navbar = () => {
  return (
    <nav className="sm:gap-10 shadow-2xl p-8 gap-4 flex bg-fuchsia-50 text-fuchsia-900 dark:bg-fuchsia-900 dark:text-fuchsia-50">
      <NavLink to="/home">Home</NavLink>
      <NavLink to="/about">About</NavLink>
      <div className="flex-1"></div>
      <div className="hidden sm:flex sm:gap-10">
        <NavLink to="/products">Products</NavLink>
        <NavLink to="/login">Login</NavLink>
      </div>
      <NavLink to="/register">Register</NavLink>
    </nav>
  );
};

export default Navbar;
```

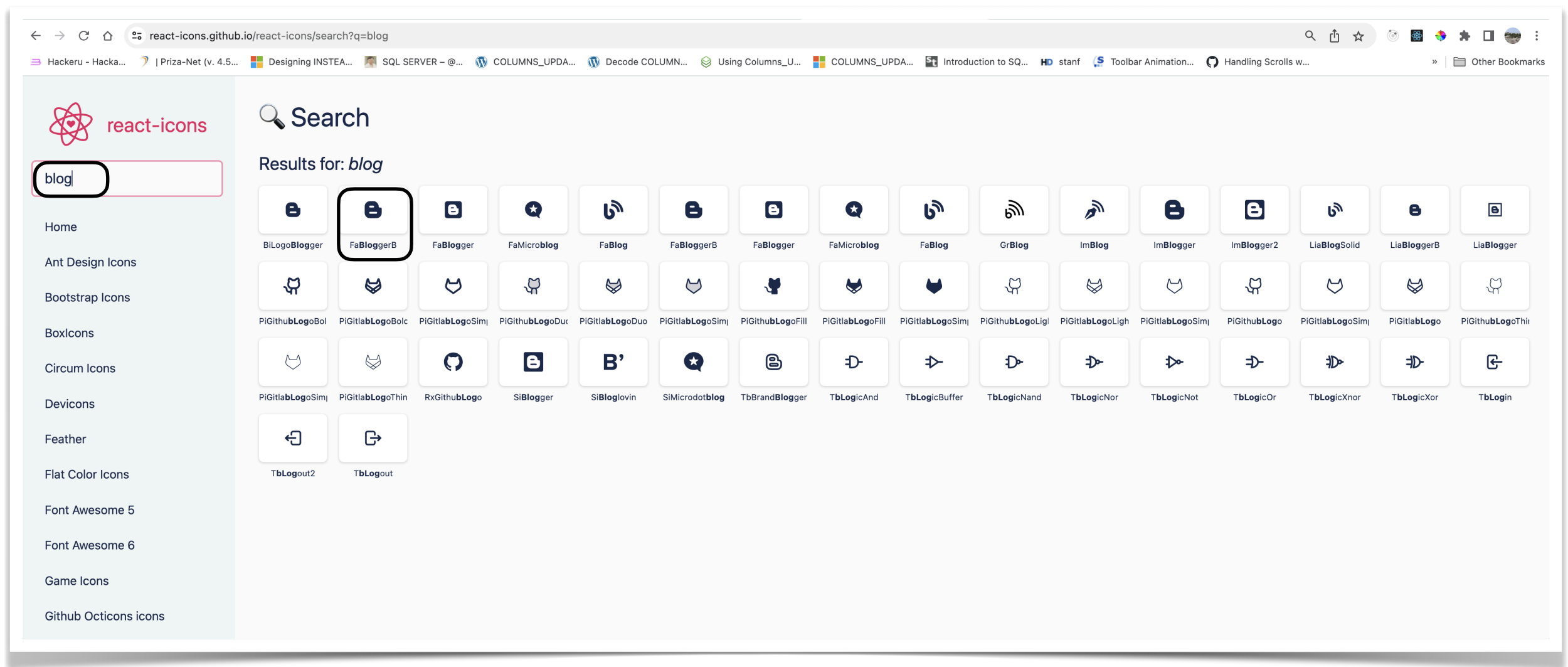
CSS
עבור כל היתר:



אחרי ההפסקה - נוסף לוגו
ונתייחס למצב active

ספריית אייקונים לראקט:

npm i react-icons



תגובה ל isActive של NavLink מתוך הספרייה React-Router-Dom

```
<NavLink className={ (o) => (o.isActive ? "text-black" : "") } to="/home">  
  <FaBloggerB />  
</NavLink>
```

```
import { NavLink } from "react-router-dom";  
import { FaBloggerB } from "react-icons/fa";  
import { AiFillGithub } from "react-icons/ai";  
import './Navbar.scss'  
const Navbar = () => {  
  
  return (  
    <nav  
      id="app-nav"  
      className="sm:gap-10 shadow-2xl 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">  
        <NavLink to="/posts">Posts</NavLink>  
        <NavLink to="/login">Login</NavLink>  
        <a href="https://github.com/">  
          <AiFillGithub />  
        </a>  
      </div>  
      <NavLink to="/register">Register</NavLink>  
    </nav>  
  );  
};  
  
export default Navbar;
```

עיצוב של active.

2

Navbar.scss U

```
#app-nav .active {  
  color: black;  
  font-size: larger;  
}
```

ספרייה לעבודה עם scss/sass

npm install sass

1

3

Navbar.tsx U × Navbar.scss U

src > components > Navbar.tsx > ...

```
1 import { NavLink } from "react-router-dom";  
2 import { FaBloggerrB } from "react-icons/fa";  
3 import { AiFillGithub } from "react-icons/ai";  
4 import './Navbar.scss'  
5 const Navbar = () => {  
6  
7   return (
```

כתיבת חוקי ולידציה עם הספרייה **yup**:

```
npm i yup
```

עבודה עם טפסים - הספרייה **formik**:

```
npm i formik
```

```
npm i yup formik
```

כתיבת חוקי ולידציה עם הספרייה yup:

npm i yup

```
import * as Yup from "yup";

const Register = () => {

  const validationSchema = Yup.object({
    username: Yup.string().min(2).required(),
    email: Yup.string().email().required(),
    password: Yup.string()
      .min(6)
      .matches(/^(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])(?=.*\W){8,20}$/)
      .required(),
  });

  return <div>Register</div>;
};

export default Register;
```

עבודה עם Formik:

מגדירים ערכים התחלתיים, ופונקציה שתגיב ללחיצה על Submit

```
import { Formik } from "formik";
import * as Yup from "yup";

const Register = () => {
  const validationSchema = Yup.object({
    username: Yup.string().min(2).required(),
    email: Yup.string().email().required(),
    password: Yup.string()
      .min(6)
      .matches(/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?\W){8,20}$/)
      .required(),
  });

  const initialValues = {
    username: "",
    email: "",
    password: "",
  };

  return (
    <Formik
      validationSchema={validationSchema}
      initialValues={initialValues}
      onSubmit={() => {}}
    >
      <Form></Form>
    </Formik>
  );
};

export default Register;
```

```
import { ErrorMessage, Field, Formik } from "formik";
import * as Yup from "yup";
```

```
const Register = () => {
  const validationSchema = Yup.object({
    username: Yup.string().min(2).required(),
    email: Yup.string().email().required(),
    password: Yup.string()
      .min(6)
      .matches(/^(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])(?=.*\W){8,20}$/)
      .required(),
  });
```

```
  const initialValues = {
    username: "",
    email: "",
    password: "",
  };
```

```
  return (
    <Formik
      validationSchema={validationSchema}
      initialValues={initialValues}
      onSubmit={() => {}}
    >
      <Form>
```

```
        <div className="form-group">
          <label htmlFor="username">User Name</label>
          { /* label that describes the input */ }
          { /* the input */ }
          <Field name="username" type="text" id="username" />
          { /* error message for the input */ }
          <ErrorMessage name="username" component="div" className="text-red-500" />
        </div>
      </Form>
    </Formik>
```

```
  );
```

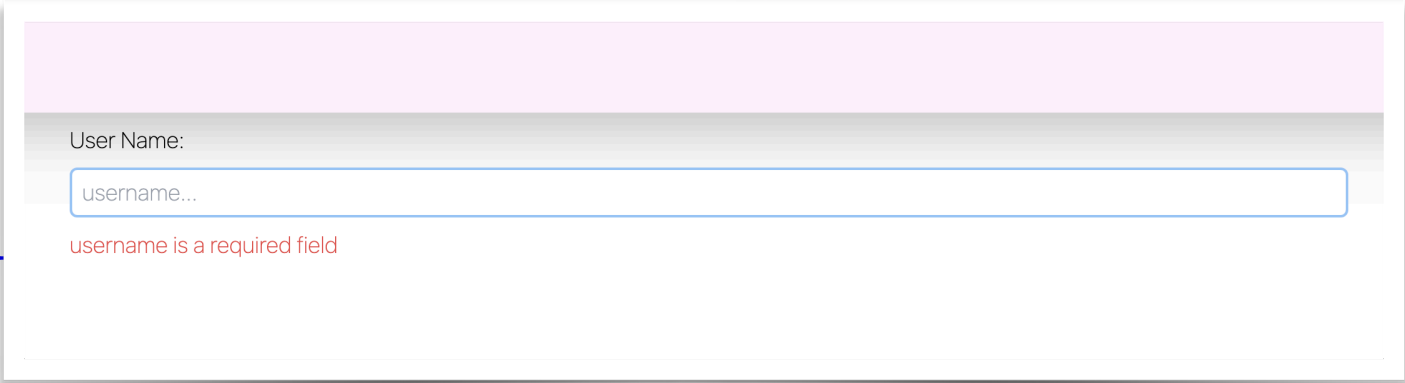
```
};
```

לכל קלט בטופס צריך:
label+input+error message

label
יש תכונה htmlFor
שמקבלת את הid של הinput

עיצוב של הקלט

```
<Formik
  validationSchema={validationSchema}
  initialValues={initialValues}
  onSubmit={() => {}}
>
  <Form>
    <div className="font-extralight text-lg w-1/2 mx-auto my-2 form-group gap-2 flex flex-col">
      <label htmlFor="username">your name:</label>
      { /* label that describes the input */ }
      { /* the input */ }
      <Field
        className="px-2 py-1 rounded-md border-blue-300 border-2"
        placeholder="username..."
        name="username"
        type="text"
        id="username"
      />
      { /* error message for the input */ }
      <ErrorMessage
        name="username"
        component="div"
        className="text-red-500"
      />
    </div>
  </Form>
</Formik>
```



label
htmlFor תכונה

שמקבלת את הid של הinput

```
<Formik
  validationSchema={validationSchema}
  initialValues={initialValues}
  onSubmit={() => {alert("Hello")}}
>
<Form className="bg-white shadow-md rounded-lg my-2 w-1/2 mx-auto p-4 flex flex-col gap-2">
  <div className="font-extralight text-lg my-2 form-group gap-1 flex flex-col">
    <label htmlFor="username">User name:</label>
    <Field
      className="px-2 py-1 rounded-md border-blue-300 border-2"
      placeholder="username..."
      name="username"
      type="text"
      id="username"
    />
    <ErrorMessage
      name="username"
      component="div"
      className="text-red-500"
    />
  </div>

  <div className="font-extralight text-lg my-2 form-group gap-2 flex flex-col">
    <label htmlFor="email">Email address:</label>
    <Field
      className="px-2 py-1 rounded-md border-blue-300 border-2"
      placeholder="email..."
      name="email"
      type="email"
      id="email"
    />
    <ErrorMessage name="email" component="div" className="text-red-500" />
  </div>

  <div className="font-extralight text-lg my-2 form-group gap-2 flex flex-col">
    <label htmlFor="password">Password:</label>
    <Field
      className="px-2 py-1 rounded-md border-blue-300 border-2"
      placeholder="password..."
      name="password"
      type="password"
      id="password"
    />
    <ErrorMessage
      name="password"
      component="div"
      className="text-red-500"
    />
  </div>

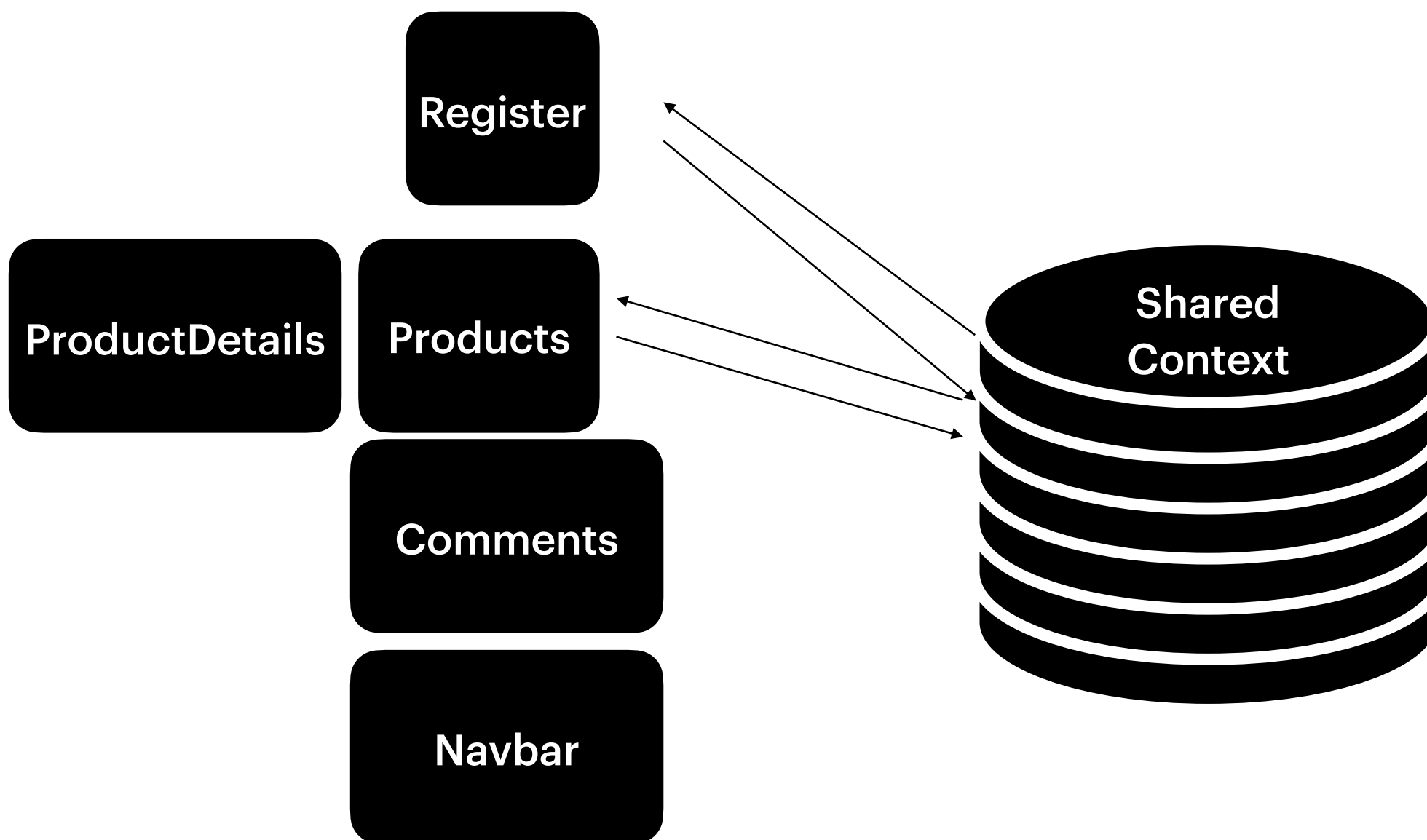
  <button className="rounded text-white px-3 py-2 w-full bg-fuchsia-700">Register</button>
</Form>
</Formik>
```

עיצוב של הטופס:

Context API

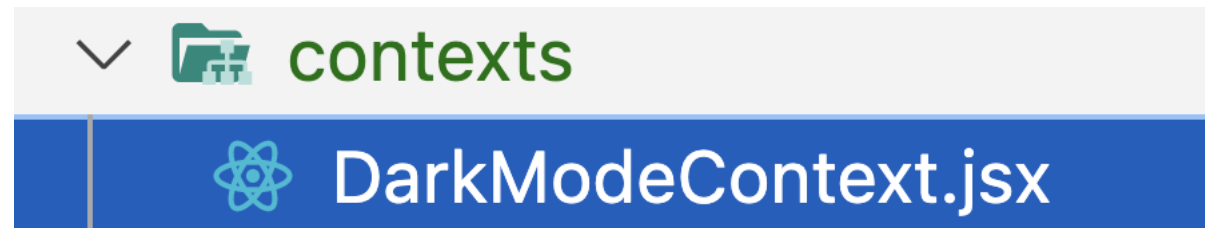
שמירת מידע בזכרון - כך שיהיה נגיש עבור כל קומפוטננטה באפליקציה

אחרת כל משתנה שנגדיר הוא לוקלי לקומפוטננטה שבה הוא מוגדר



Context API

ניצור קובץ חדש שבו נגדיר את "חנות המידע"



Context API

לפני שנגדיר את הcontext:
הגדרות לפרוייקט בנושא typescript



```
{  
  "compilerOptions": {  
    "isolatedModules": true,  
    "moduleDetection": "force",  
    "noEmit": true,  
    "jsx": "react-jsx",  
  
    /* Linting */  
    "noImplicitAny": false,  
    "strict": false,  
    "noUnusedLocals": false,  
    "noUnusedParameters": false,  
    "noFallthroughCasesInSwitch": true  
  },  
  "include": ["src"]  
}
```

מכיוון שעדיין לא למדנו על הגדרות טיפוסים בtypescript
"נכבה" את הבדיקות של הגדרת טיפוסים strict mode

(נחזור לזה בסוף הפרוייקט ונראה איך אפשר להשאיר את הבדיקות ואיתן ליצור קוד טוב/מתועד יותר).

Context API

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

//1) initial state:

```
const initialState = {  
  darkMode: false,  
};
```

1

מגדירים אובייקט עם מצב התחלתי

//2) create the context "STORE"

```
const DarkModeContext = createContext(initialState);
```

2

יוצרים מופע של "חנות"

//3) create a wrapper component

```
const DarkModeContextWrapper = ({ children }) => {  
  return (  
    <DarkModeContext.Provider>  
      {children}  
    </DarkModeContext.Provider>  
  );  
};
```

3

יצירה של קומפוננטה עוטפת
שתפקידה להנגיש את החנות לקומפוננטות האחרות

```
export {DarkModeContext, DarkModeContextWrapper}
```

Context Api

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

```
//1) initial state:
```

```
const initialState = {  
  darkMode: false,  
  toggleDarkMode: () => {},  
};
```

(א) מה שרוצים לשתף:
תכונות
פעולות

```
//2) create the context "STORE"
```

```
const DarkModeContext = createContext(initialState);
```

```
//3) create a wrapper component
```

```
const DarkModeContextWrapper = ({children}) => {  
  const [darkMode, setDark] = useState(false);  
  
  const toggleDarkMode = () => {  
    setDark((d) => !d);  
  };  
};
```

(ב)
ניצור משתני מצב עבור התכונות
ופונקציות לשינוי המצב

```
return (  
  <DarkModeContext.Provider  
    value={{ darkMode: darkMode, toggleDarkMode: toggleDarkMode }}  
  >  
    {children}  
  </DarkModeContext.Provider>  
>);  
};
```

```
export {DarkModeContext, DarkModeContextWrapper}
```

הנגשת Context 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";

const root = ReactDOM.createRoot(
  document.getElementById("root") as HTMLElement
);
root.render(
  <BrowserRouter>
    <DarkModeContextWrapper>
      <App />
    </DarkModeContextWrapper>
  </BrowserRouter>
);

reportWebVitals();
```

שימוש בContext

```
import { Route, Routes } from "react-router-dom";
import Home from "../routes/Home";
import About from "../routes/About";
import Register from "../routes/Register";
import Login from "../routes/Login";
import Posts from "../routes/Posts";
import Navbar from "../components/Navbar/Navbar";
import NotFound from "../routes/NotFound";
import { useContext } from "react";
import DarkModeContext from "../contexts/DarkModeContext";
```

```
const App = () => {
```

```
  const { darkMode, toggleDarkMode } = useContext(DarkModeContext);
```

```
  return (
```

```
    <>
```

```
    <h1 onClick={()=>{toggleDarkMode()}}>{darkMode ? "Dark" : "Not Dark"}</h1>
```

```
    <Navbar />
```

```
    <Routes>
```

```
      <Route path="/" element={<Home />} />
```

```
      <Route path="/home" element={<Home />} />
```

```
      <Route path="/about" element={<About />} />
```

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

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

```
      <Route path="/posts" element={<Posts />} />
```

```
      <Route path="*" element={<NotFound />} />
```

```
    </Routes>
```

```
  </>
```

```
);
```

```
};
```

```
export default App;
```

קיצורי מקשים:

alt shift f

פרמוט המסמך

alt ↓

הורדת שורה למטה

alt shift ↓

שכפול שורה

ctrl .

quick fix

(מקביל לalt enter)

ctrl + d

מסמנים ביטוי או שורה עם העכבר ואז לוחצים

ctrl+d

וזה מביא לסימון של הערך הזהה הבא במסמך