## **🧑‍🏫 מערך שיעור: יצירת קובץ UserContext.jsx לניהול משתמשים ב־React**

### **🎯 מטרות השיעור**

* להבין את עקרון הקונטקסט (Context API) בריאקט.
* ליצור UserContext עם מידע על המשתמש המחובר.
* לשמור את המשתמש ב־localStorage ולשחזר אותו.
* לספק פעולות גלובליות: login, logout, isAuthenticated, updateUser, isAdmin ועוד.

## **📁 שלב 1: מיקום הקובץ בפרויקט**

יש למקם את הקובץ בתוך:

src/contexts/UserContext.jsx

## **🧱 שלב 2: מבנה כללי של הקובץ**

הקובץ כולל 3 חלקים עיקריים:

1. יצירת הקונטקסט: createContext()
2. יצירת hook מותאם: useUser()
3. יצירת Provider: UserProvider

## **🧩 שלב 3: הסבר חלhttps://braintop.vercel.app/general-displayקי הקוד**

### **✅ 1. יצירת הקונטקסט**

const UserContext = createContext();

🔹 זהו האובייקט שנשתמש בו כדי לשתף מידע גלובלי בין קומפוננטות.

### **✅ 2. יצירת hook מותאם אישית לשימוש נוח בקונטקסט**

export const useUser = () => {

const context = useContext(UserContext);

if (!context) {

throw new Error('useUser must be used within a UserProvider');

}

return context;

};

🔹 מספק דרך נוחה לקרוא ל־context בלי צורך ב־useContext(UserContext) בכל פעם.  
 🔹 מגן מפני שימוש לא חוקי מחוץ ל־Provider.

### **✅ 3. יצירת UserProvider – מעטפת גלובלית לאפליקציה**

export const UserProvider = ({ children }) => {

...

return (

<UserContext.Provider value={value}>

{children}

</UserContext.Provider>

);

};

🔹 עוטף את כל האפליקציה ונותן לכל קומפוננטה גישה ל־user, login, logout וכו’.

### **✅ 4. ניהול המשתמש – סטייטים ופעולות**

#### **🎮 סטייטים:**

const [user, setUser] = useState(null);

const [isAuthenticated, setIsAuthenticated] = useState(false);

const [loading, setLoading] = useState(true);

#### **🔐 טעינה מ־localStorage:**

useEffect(() => {

const savedUser = localStorage.getItem('user');

...

}, []);

#### **✔️ התחברות:**

const login = (userData) => {

setUser(userData);

setIsAuthenticated(true);

localStorage.setItem('user', JSON.stringify(userData));

};

#### **❌ התנתקות:**

const logout = () => {

setUser(null);

setIsAuthenticated(false);

localStorage.removeItem('user');

};

#### **✏️ עדכון משתמש:**

const updateUser = (newData) => {

const updatedUser = { ...user, ...newData };

setUser(updatedUser);

localStorage.setItem('user', JSON.stringify(updatedUser));

};

#### **📦 פונקציות עזר:**

* getUserData() – מחזירה את המשתמש מ־localStorage.
* hasRole(roleId) – בדיקת תפקיד.
* isAdmin() – בדיקת האם המשתמש אדמין.

| **מושג** | **הסבר** |
| --- | --- |
| createContext | יוצר אובייקט קונטקסט לשיתוף מידע |
| useContext | קריאה לערך מתוך הקונטקסט |
| Provider | מעטפת שמספקת ערכים גלובליים לכל הרכיבים שבתוכה |
| children | כל מה שנמצא בין תגי פתיחה וסגירה של קומפוננטה |
| localStorage | אחסון נתונים מקומי בדפדפן שנשמר בין רענונים |

דוגמא

import React, { createContext, useContext, useState, useEffect } from 'react';

*// Create the context*

const UserContext = createContext();

*// Custom hook to use the UserContext*

export const **useUser** = () => {

const context = useContext(UserContext);

if (!context) {

throw new Error('useUser must be used within a UserProvider');

}

return context;

};

*// UserProvider component*

export const **UserProvider** = ({ *children* }) => {

const [user, **setUser**] = useState(null);

const [isAuthenticated, **setIsAuthenticated**] = useState(false);

const [loading, **setLoading**] = useState(true);

*// Check if user is already logged in (e.g., from localStorage)*

useEffect(() => {

const savedUser = localStorage.getItem('user');

if (savedUser) {

try {

const userData = JSON.parse(savedUser);

setUser(userData);

setIsAuthenticated(true);

} catch (error) {

console.error('Error parsing saved user data:', error);

localStorage.removeItem('user');

}

}

setLoading(false);

}, []);

*// Login function (without token support)*

const **login** = (*userData*) => {

setUser(*userData*);

setIsAuthenticated(true);

*// Save to localStorage for persistence*

localStorage.setItem('user', JSON.stringify(userData));

};

*// Logout function*

const **logout** = () => {

setUser(null);

setIsAuthenticated(false);

*// Clear localStorage*

localStorage.removeItem('user');

};

*// Update user data*

const **updateUser** = (*newUserData*) => {

const updatedUser = { ...user, ...*newUserData* };

setUser(updatedUser);

localStorage.setItem('user', JSON.stringify(updatedUser));

};

*// Get user data from localStorage*

const **getUserData** = () => {

const savedUser = localStorage.getItem('user');

if (savedUser) {

try {

return JSON.parse(savedUser);

} catch (error) {

console.error('Error parsing user data:', error);

return null;

}

}

return null;

};

*// Check if user has specific role*

const **hasRole** = (*roleId*) => {

return user && user.role\_id === *roleId*;

};

*// Check if user is admin*

const **isAdmin** = () => {

return hasRole(1); *// Assuming role\_id 1 is admin*

};

*// Context value*

const value = {

user,

isAuthenticated,

loading,

**login**,

**logout**,

**updateUser**,

**getUserData**,

**hasRole**,

**isAdmin**

};

return (

<UserContext.Provider *value*={value}>

{*children*}

</UserContext.Provider>

);

};

export default UserContext;