# **React**

**GitHub:** <https://github.com/AssafFink/ofek.git>

**Common Naming Conventions**

* Lowercase Camel: myBestFriend
* Uppercase Camel: MyBestFriend
* Kebab: my-best-friend
* Snake: my\_best\_friend

**React Directory Structure**

Assets

Images

Videos

Fonts

…

Services

Calc.ts

ColorPicker.ts

…

Components

LayoutArea

Layout

Header

Footer

Menu

…

ProductsArea

ProductList

ProductList.tsx

ProductList.css

AddProduct

AddProduct.tsx

AddProduct.css

…

SuppliersArea

…

SharedArea

…

Models

…

Utils

…

Redux

…

…

**Destructuring Assignment**

const [first, second, third] = [10, 20, 30]

const {name, color, age} = {name: "Mitsi", age: 4, color: "Green"}

**React Hooks**

פונקציות מיוחדות המכילות מספר אילוצים:

1. ניתן להפעיל אותן אך ורק ישירות בתוך Functional Component או בתוך React Hooks אחרים. אי אפשר להפעיל אותן מחוץ ל-FC או בתוך CC.
2. ניתן להפעיל אותן אך ורק בקוד הקיים ישירות ב-FC. לא בתוך תנאים או לולאות, לא בתוך פונקציות פנימיות אחרות.
3. פונקציות כאלו חייבות להתחיל בתחילית use.

**Side-Effect**

פעולה שקומפוננטה מבצעת שיוצאת מגבולות הקומפוננטה ומשפיעה על מידע חיצוני כלשהו.

דומאות: פתיחת טיימר, שינוי title של דף, גישה לשרת מרוחק...

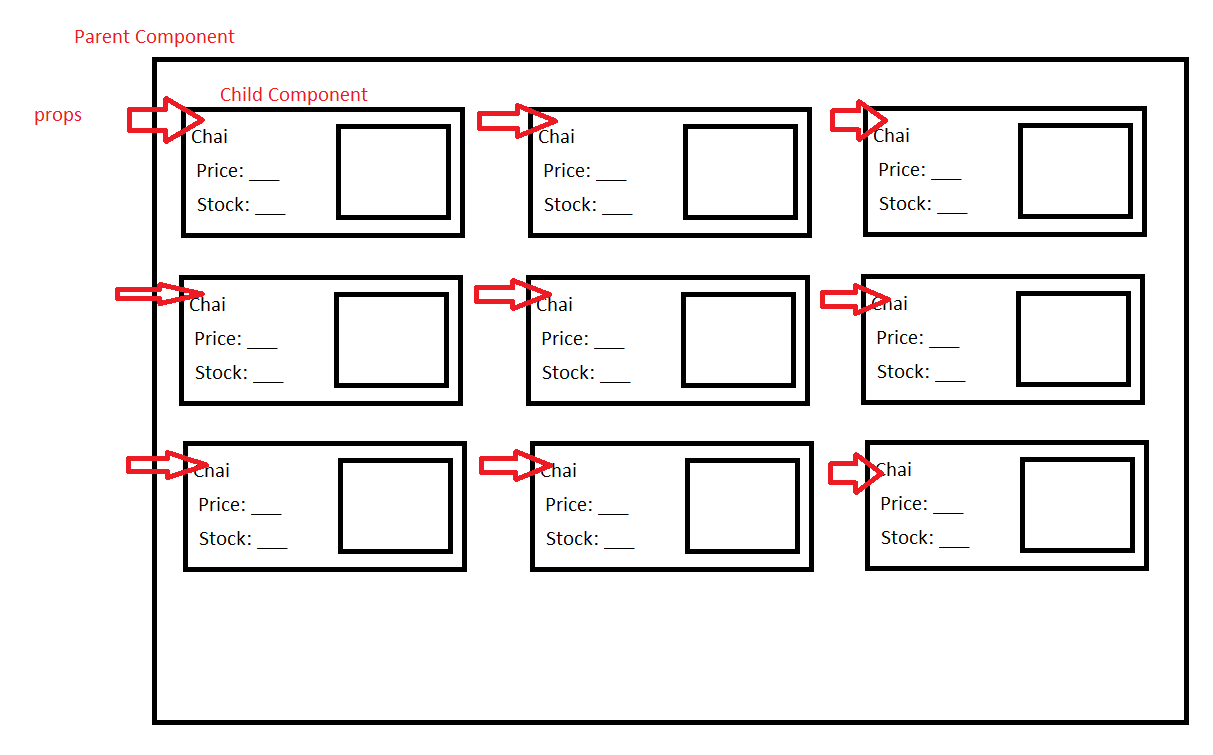
אסור לבצע Side-Effect ישירות בתוך קוד הקומפוננטה כי אנו לא יודעים מתי ריאקט תרנדר אותה שוב ותגרום שוב לפעולת ה-Side Effect.

כן ניתן לבצע Side-Effects בתוך אירועים כמו אירוע של לחצן.

קיים Hook בשם useEffect לביצוע Side-Effects ישירות בקומפוננטה.

**Props**

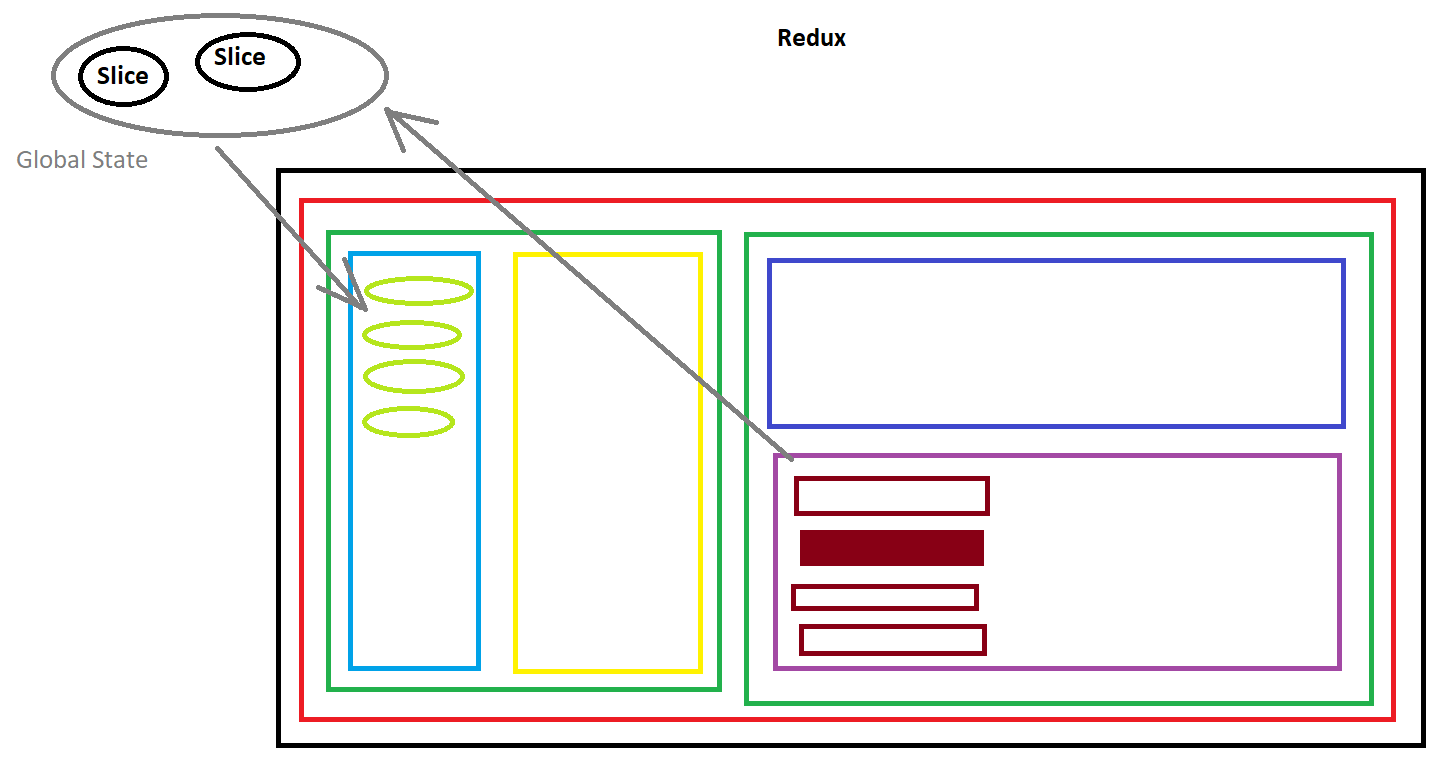
אובייקט היכול להישלח מ-Parent Component ל-Child Component:

****

**Redux**

ספרייה לניהול Global State.

התקנה: npm i @reduxjs/toolkit



**טרמינולוגיה:**

Global State: המידע הקיים ברמת האפליקציה. נקרא גם AppState.

Slice: חלק אחד מכלל המידע הגלובלי.

Action: זו פעולה אחת המתבצעת על ה-Global State. לדוגמה הוספת מוצר, מחיקת פריט מסל הקניות, עדכון מבצע...

Payload: משתנה בתוך אובייקט Action המתאר את המידע של הפעולה. לדוגמה, אם הפעולה היא הוספת מוצר, אז ה-payload יהיה המוצר להוספה.

Reducer: זו פונקציה המבצעת בפועל את הפעולה הדרושה. לדוגמה, פונקציה בשם addOne המוסיפה מוצר חדש ל-Global State.

פונקציית Reducer מקבלת שני פרמטרים:

1. ה-State הנוכחי של ה-Slice.
2. אובייקט ה-Action הנדרש לביצוע.

פונקציה כזו תמיד תחזיר Slice State חדש אחרי השינוי שהיא ביצעה. פונקציית Reducer לא תשנה את המידע הנשלח אליה אלא היא תמיד תבנה מידע חדש, תשנה את המידע החדש ותחזיר אותו. ואז המידע החדש שנחזיר יחליף ב-Global State את המידע הישן.

Store: האוביקט הראשי שמנהל את כל ה-Global State.

Dispatch: שליחת Action לביצוע.

**react-redux**

ספרייה שמספקת יכולות נוספות עבור React בהקשר של Redux.

התקנה: npm i react-redux @types/react-redux

**Auth**

Authentication: אימות - בדיקת זהות המשתמש.

Authorization: הרשאות - מה המשתמש יכול לעשות.

**JWT**

JSON Web Token

זו מחרוזת ששרת מחזיר ללקוח. המחרוזת מכילה אובייקט אחד כלשהו, לרוב זה יהיה אובייקט ה-User, תאריך יצירה ותאריך תפוגה. המחרוזת משמשת לצורך כניסה חוזרת לשרת וביצוע פעולות. על היוזר לשלוח את המחרוזת הזו עבור ביצוע פעולות בשרת הדורשות Login.

את ה-Token עלינו לשלוח ע"י Header בשם Authorization המכיל את הפורמט הבא: "Bearer the-token"

אפשר לבצע זאת בכל מקום המבצע Request אך נכון יותר לבצע זאת פעם אחת במנגנון Interceptor.

פענוח ה-Token: npm i jwt-decode

Role: תפקיד שקיים לכל יוזר. לרוב התפקיד הכי פשוט נקרא "User". יכולים להיות סוגים שונים של Roles. לדוגמה Manager או Admin.

**Redux Middleware**

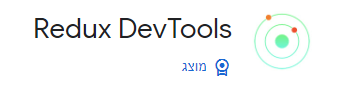
פונקציה שאנו כותבים המתבצעת בכל Dispatch.

בכל שינוי ה-Global State אנו יכולים לבצע פעולה כלשהי.

קיים Middleware המבצע Log עבור Redux:

npm i redux-logger @types/redux-logger

Debug Extensions





**MUI**

Material UI – ספריית קומפוננטות מעוצבות.

התקנה:

npm install @mui/material @emotion/react @emotion/styled @mui/icons-material