React

שיעור שישי

הפעלת reducer עם אוביקט

- ה payload הנשלח מהaction ומגיע ל reducer יכול להכיל גם אובייקט עם מספר ערכים, ואפילו מערך.

```
const users = [ {index: 1, age: 30, name: "mike"}, {index: 2, age: 40, name: "lally"}]
```

- ב reducer ניתן לשנות לפי ערך מסוים.

```
case "CHANGEUSERNAME": {
  return { ...state, name: action.payload[1].name };
}
```

הוספת אובייקט ל state

- ניתן להוסיף אובייקט ל state בצורה הבאה.
- בהגדרת ה state נגדיר אותו כמערך.

```
case "CHANGEUSERNAME": {
  return { ...state, name: action.payload[1].name };
}
```

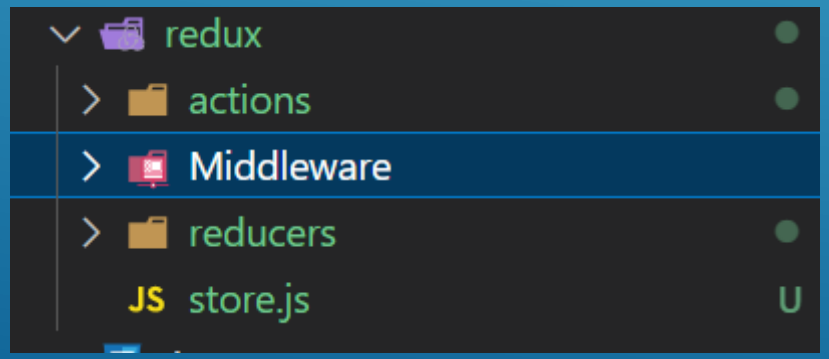
- פונקציית הוספה תשמש הפונקצייה push לדחוף את האובייקט למערך האובייקטים.

```
case "ADDUSER": {
  state.arr.push(action.payload)
}
```

- מחיקת אובייקט נעשית כרגיל – באמצעות הפונקציה filter.

Middleware - ReduxReact

- Middleware ב react-redux היא פונקציה המתרחשת בין הפעלת action לביצוע הפעולה ב reducer כשלב ביניים.
- ב middleware ניתן לשנות את הפעולה הנשלחת ל reducer לפני שהיא מגיעה לשם או להוסיף פעולה נוספת שתתבצע בעת הפעלת ה action.
- Middleware היא פונקציה המחזירה פונקציה.
- ניצור תיקיית Middleware בתיקיית ה redux.



Middleware

- Middleware ניתן לומר שזה פונקציה המקבלת next,action ומחזירה פונקציה המקבלת את הפונקציה הבאה נוספת... ולבסוף מחזירה פונקציה אחרת שמקבלת את הaction.
- חתימת middleware נראית כך:

```
export const customMiddleware = () => next => action => {
```

- next היא הפעולה שנקראת בסיום ה middleware (הפונקציה שה Middleware מחסזירה) תפקידה להפעיל middleware נוסף או להפעיל את ה action ב reducer כפי שהגדרנו.
- action– הוא הפעלה שהופעלה.
- דוגמא ל middleware בסיסי (הכותב לוג בעת הפעלת ה action)
- נדאג לייצא פונקציות אלו על מנת שנוכל להשתמש בהם.

```
export const customMiddleware = () => next => action => {
    console.log("middleware triggered", action)
    return next(action);
}
```

React

Middleware - הפעלה

- הפעלת ה middleware נעשית ב store – שם נגדיר את ה middlewares שנרצה שיופעלו.
- נייבא את applyMiddleware מספריית redux :

```
import { applyMiddleware } from "redux";
```

- נייבא את פונקציות ה middleware שיצרנו.

```
import { customMiddleWare } from "./Middleware/UserMiddleware";
```

- במיקום בו מצהירים על ה reducers (ב createStore) נוסיף את ה Middleware בתוך אובייקט ה apllyMiddleware בצורה הבאה:

```
const store = createStore(
  allReducers, applyMiddleware(customMiddleWare)
);
```

- בכתיבה זו, ה middleware ירוץ בכל הפעלת action.

שליטה על ביצוע ה Middleware

- בכתיבה רגילה של Middleware, לאחר הוספה שלו ב store הוא ירוץ בכל הפעלת action.
- ניתן לשלוט על ביצוע ה Middleware שיופעל רק ב action מסוים. את זה נעשה בתוך ה Middleware.
- ה Middleware מקבל action – את הפעול שעכשיו הופעלה, ניתן לפי סוג ה action שהגיע ל Middleware להפעיל את ה Middleware או לא בצורה הבאה:

```
export const customMiddleware = () => next => action => {
    if (action.type === "CHANGEUSERNAME") {
        console.log("middleware triggered", action)
    }
    return next(action);
}
```

- בצורה זו ה Middleware יופעל רק אם הגיע אליו action type זה.
- כך ניצור Middleware מתאים לכל סוג פעולה.

Middleware עם קריאות API

- הרבה פעמים נרצה ליצור קראיות API בהפעלת Middleware, וזו ה-מטרה של react-redux שבמקום שנצטרך צד שלישי שידאג לבצע שליפות וכד' ניתן לכתוב בתוך ריאקט עצמו ב redux.
- על מנת לבצע קריאות API אנחנו צריכים שפונקציית ה Middleware תהיה א-סינכרונית. שלבים:
- URL מתאים לקריאת API. דוגמא ל URL לשליפת יוזרים: https://api.github.com/users/hadley/orgs
- נוסיף לפני ה action שמקבלת הפונקציה את המילה async שאומרת שפעלוה זו היא א-סינכרונית.
- ניצור פעולת fetch – משיכת המידע (כפי שלמדנו בקריאות API) פעולה זו צריכה להיות עם await.
- נקבל את המידע לצורה של json (קל להתעסקות) ונבצע את הפעולות שצריך.
- כמובן לא נשכח לייצא פונקציה זו ולהכיל אותה ב store.

```
export const getUserName = () => next => async action => {
    if (action.type === "CHANGEUSERNAME") {
        const http = await fetch("https://api.github.com/users/hadley/orgs");
        const userList = await http.json();
    }
    return await next(action)
}
```

שימוש ב reducer עם המידע שהתקבל

- על מנת להשתמש במידע שהתקבל (זאת המטרה העיקרית ב Middleware) – פשוט נשנה את ה payload הנשלח ל reducer מה action.
- (גם אם נשלח מידע אחר ב payload בתוך ה action או לא נשלח שום ערך – ה Middleware הוא הקובע את ה payload כי קורה בין הפעלת ה action לביצוע ב reducer.
- חוץ מזה הכל נשאר כמו שהיה. (ב action וב reducer)
- דוגמא:

```
export const getUserName = () => next => async action => {
    if (action.type === "CHANGEUSERNAME") {
        const http = await fetch("https://api.github.com/users/hadley/orgs");
        const userList = await http.json();
        action.payload = userList[0].login
    }
    return await next(action)
}
```

React dev tools

- יש תוסף לכרום ששמו react dev tools. תפקידו לשרת אותנו בכתיבת react-redux.
- באמצעותו ניתן לראות את כל הפעולות המתרחשות ב redux בזמן אמת, מקל לעלות על בעיות.
- התקנת התוסף בלינק הזה https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd

שיעורי בית

- יצירת state של Products עם מספר פריטים (מערך אובייקטים)
- הצגת כל הפריטים בקומפוננט (שליפת האובייקט מה store והצגת הנתונים מתוך map)
- תהיה אפשרות עריכה לכל אחד מהפריטים. – ליד כל אחד מהפריטים יהיה input עם אפשרות עריכה של פריט זה. במקרה זה ב onChange ישלח ב dispatch אובייקט המכיל אינדקס ואת הפרטים שצריך.
- פונקציית עידכון ה state יכיל middleware המבצע קריאת API לשליפת המוצרים דוגמא ל URL: https://shopname.myshopify.com/products.json
  רק לאחר שהמוצרים נשלפו העידכון יוכל להתבצע. ניתן גם שהעידכון יעשה באמצעות הנתונים שנשלפו מה API.
- הוספת יוזר – יהיה ניתן לבצע הרשמה ליוזר, במקרה וקיים יוזר – בוצעה הרשמה, כותרת העמוד תהיה שלום + שם היוזר. אחרת לא תוצג כותרת לעמוד.

בהצלחה!