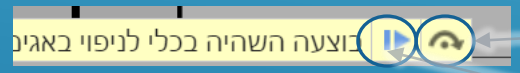
React
שיעור שני

debug

- דיבוג הוא כלי חזק לפתירת בעיות. לעיתים נרצה לדבג ולראות שורה אחר שורה את המתבצע בקוד.
- שלבים:
  * כתיבת המילה debugger במיקום בפונקציה ממנו נרצה לדבג.
  * הרצת הפרויקט ופתיחת dev tools – (f12), כאשר הקוד המתבצע מכיל debugger ההרצה תעצור ונוכל לדבג את שורות הקוד.
- מעבר לשורת קוד הבאה
- קפיצה עד לdebugger הבא אם יש.



שמירת מערכים ב state

- States יכולים להכיל מערך או מערך של אובייקטים בצורה הבאה:
בתוך ה useState נשים סוגריים מרובעים – מערך.

```
const [arr, setArr] = useState([2,3,5,6,7]);
```

- במערך של אובייקטים – כל ערך במערך יהיה אובייקט המכיל בתוכו את הערכים שנצטרך.
הערכים כתובים כאובייקט – בצורה של key ו value לדוג:

```
const [products, setProducts] = useState([
    {
        name: "product1",
        amount: 60
    },
    {
        name: "producr2",
        amount: 90
    },
]);
```

React

גישה לאיברי המערך

- הגישה לאיברי מערך ב state נעשית בצורה הבאה:
במערך רגיל – של המשתנה ואינדקס אליו נרצה לגשת.
לדוגמא הצגת האיבר השלישי במערך arr :

```
<h1>{arr[2]}</h1>
```

- במערך אובייקטים – שם ה state, האינדקס, ואת ה key אליו נרצה לגשת.
לדוגמא הצגת הערך ב amount הקיים באיבר הראשון במערך:

```
<h1>{products[0].amount}</h1>
```

לולאות - map

- שימוש בלולאת for לדוגמא בשביל מעבר על מערך והצגת פריטי המערך, שונה מכתיבתה ב java script ושאר השפות ונעשית באמצעות map :
- דוגמא להצגת כל איברי המערך arr בשורות נפרדות: - בתגית h1 נפרדת לכל אחד:
- - itemה מכיל את הערך הנוכחי עליו נמצא המערך.

```
{arr.map(item =>(
    <h1>{item}</h1>
))}
```

- ניתן גם לשמור את ה index עליו הלולאה עומדת - אם נוסיף index כפרמטר נוסף ב map בצורה הבאה:

```
{arr.map((item, index) => (
    <h1>{item}{index}</h1>
))}
```

- (משמש הרבה לדוגמא כדי לדעת על שינויים הנעשים במערך - באיזה אינדקס צריך לעדכן וכד'.)

לולאות על מערך אובייקטים - map

- באובייקטים כל key במערך האובייקטים הוא רשימה בפני עצמה. ולכן מעבר על רשימת אובייקטים הוא על ידי מעבר על כל אחד מה keys במערך. לדוגמא בצורה זו:

```
{products.map(({ name, amount }) => {
    return <p>{name} - {amount}</p>
})}
```

- ניתן גם לעשות זאת באמצעות הצגה בתוך טבלה או רשימה כאשר השורות יהיה הkeys של האובייקטים והעמודות יהיו הערכים. לדוגמא:

```
<table>
    {products.map((item, index) => (
        <tr>
            <td>
                {item.name}
            </td>
            <td>
                {item.amount}
            </td>
        </tr>
    ))}
</table>
```

bootstrap

- הוספת bootstrap בפרויקט react פשוט ביותר. ומאפשר לעצב כל אלמנט שנרצה בצורה מקובלת, רספונסיבית נקייה וללא מאמץ.
- יש המון עיצובים לכל אלמנט, ניתן לראות בקישור הזה:https://getbootstrap.com/docs/4.4/components/alerts/
- התקנת bootstrap בפרויקט react בצורה הקלה ביותר היא להוסיף את ה script הזה בעמוד index.html בפרויקט.:

```
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"

    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
```

- מעכשיו נוכל לעצב בעזרת bootstrap או לחילופין ב css כרגיל.

שינוי מערך

- שינוי מערך לא נעשה בצורה רגילה, לדוגמא אם נרצה לערוך או להוסיף ערך למערך נצטרך לבצע את השלבים הבאים:

```
<input onChange={(e) => changeProductAmount(e.target.value, index)} />change
amount
```

- העתקת המערך למערך חדש.
- שינוי הערך במערך החדש.
- החלפת המערך המקורי במערך החדש

```
const changeProductAmount = (newAmount, index) => {
    let updatedProducts = [...products];
    if (newAmount) {
        updatedProducts[index].amount = newAmount;
    }
    setProducts(updatedProducts);
}
```

- הסבר: היינו מצפים לעשות את הפעולה הבאה לעידכון המערך: `products[index].amount = newAmount;`
אבל היא לא אפשרית ולא תשנה את המערך. שינוי ערכים בreact הוא רק על ידי הפונקציה set והיא מקבלת את כל האובייקט אותו צריכה להכניס למשתנה. ולכן בשינוי מערכים, צריך לשלוח מערך שלם ערוך אותו נכניס במקום המערך הקודם.
- אותה הדרך גם בשינוי מערך רגיל:

```
const changeNumber = (number, index) => {
    let newNubbers = [...arr];
    newNubbers[index] = number;
    setArr(newNubbers);
}
```

הסבר האופרטור '...'

- האופרטור ... מאפשר העתקת מערך או אובייקט למערך או אובייקט אחר.

```
let updatedProducts = [...products];
```

- ניתן גם להעתיק כמה מערכים לתוך אובייקט באמצעות האופרטור , בין המערכים:

```
let obj = [...products , ...arr];
```

עדכון מערך רק לאחר שמירת הערכים

- ניתן גם לעדכן מערך בפעם אחת לאחר ביצוע כל השינויים ולא לאחר ביצוע כל שינוי.
- אם הבנו שדרך שינוי מערך צריכה להיעשות במערך שלם – להחליפו במערך הקיים אז צריך ליצור אחד כזה.

```
const [itemNane, setItemName] = useState('');
const [itemAmount, setItemAmount] = useState('');
```

- במקרה זה ניצור משתנה לכל ערך מהמערך אותו נרצה לשנות.
- נשנה ערכים אלו בביצוע שינויים על ידי הפונקציה set של המשתנה כרגיל.

```
<input onChange={(e) => setItemName(e.target.value)}>
```

- בהפעלת שינוי המערך – נשלח את ערכי המשתנים

```
<button  onClick={updateItem(itemNane, itemAmount, index)}>update
```

- ושאר השלבים הם כמו בעידכון רגיל.

```
const updateItem = (name, amount, index) => (event) => {
    let newArr = [...products];
    if (name) {
        newArr[index].name = name;
    }
    if (amount) {
        newArr[index].amount = amount;
    }
    setProducts(newArr);
};
```

React

הוספת איבר במערך

- בשביל להוסיף איבר למערך בשלבים הבאים:
- נשלח איבר לפונקציה

```
<button onClick={() => addItem({ "name": itemNane, "amount": itemAmount })}>הוספת פריט<
```

- העתקת המערך למערך חדש הכולל גם את האיבר החדש.
- עידכון המערך הרגיל במערך החדש.

```
function addItem(newItem) {
    let newState = [...products, newItem];
    setProducts(newState);
}
```

React

מחיקת איבר מהמערך - filter

- מחיקת איבר ממערך נעשית באמצעות הפונקציה filter.
- הפונקציה filter מסננת את המערך לפי האינדקס שנתנו לה. ומחזירה מערך חדש המכיל רק את הערכים השווים ל true מהתנאי שהצבנו לה.
- דוגמא למחיקת איבר מהמערך:

```
const deleteItem = (i) => {
    setProducts(currentItems => currentItems.filter((item, index) => index !== i));
}
```

שיעורי בית:

- יש ליצור מעין חנות הכוללת את האפשרויות הבאות:
  * הצגת כל המוצרים.
  * עידכון מוצר
  * הוספת מוצר
  * מחיקת מוצר.
- המוצרים יאוכסנו במערך של אובייקטים וכל הפעולות יהיו כפי שמפורט במצגת.
- עצבו את הטבלה והכפתורים באמצעות bootstrap.

בהצלחה!!

React