## **חלק 1 - מה זה react**

* ספריית JavaScript לבניית ממשקי משתמש (UI).
* גישה מבוססת **רכיבים**.
* רינדור דינמי – משתנה לפי מצב (state) ונתונים (props).

חלק 2: יצירת פרויקט עם Vite

### **שלבים:**

1. **צור פרויקט חדש**:

npm create vite@latest **my-react-app** -- --template react

1. **התקנת חבילות**:

cd my-react-app

npm install

כדי להריץ : npm run dev

כדי לעצור - ctrl c

מחקתי את התוכן של index.css (אבל כמו שאמר סרגיי - הקובץ עצמו נשאר )

מחקתי את התוכן של App.css (אבל כמו שאמר סרגיי - הקובץ עצמו נשאר )

ונשארתי ב App.jsx עם שורות אלו

מה זה קובת עם סיומת jsx - זה קובץ המכיל html ו js משולב

function App() {

return (

<>

<p>hello world</p>

</>

)

}

export default App

צור קומפוננטה הראשית app.jsx

3 משתנים שהם

שם מדינה

שם עיר

רחוב

צור פונקציה המציגה על ידי alert את שלושתם.

הפעל את הפונקציה קומפוננטה הראשית על ידי שילוב כפתור. לחיצה על כפתור תציג את הכתובת

**Hook (וו)** הוא **פונקציה מיוחדת של React** שמאפשרת לך "להתחבר" ליכולות של React – כמו מצב (state), מחזור חיים (lifecycle), הקשר (context) ועוד – בתוך **קומפוננטות פונקציונליות**.

בעבר יכולת להשתמש ב־state רק בקומפוננטות מחלקה (class)  
 מאז ש־Hooks נוספו (בגרסה 16.8), אפשר לעשות הכל בפונקציות רגילות!

## **🔑 דוגמה הכי פשוטה: useState**

import { useState } from "react"

function App() {

const [counter, setCounter] = useState(15)//setCounter is function

//let counter = 0

function increment() {

setCounter(counter + 1)

//counter++

console.log(counter)

}

return (

<>

<h1>useState</h1>

<button onClick={increment}>increment</button>

<p>{counter}</p>

</>

)

}

export default App

### **מה קורה פה?**

* useState(0) מחזיר **מערך עם שני איברים**:  
  1. count – ערך המצב הנוכחי
  2. setCount – פונקציה שמשנה אותו
* React זוכר את הערך ומעדכן את המסך אוטומטית כשמשתמשים ב־setCount.

## **🧪 תרגיל React – מונה עם useState**

### **🎯 מטרות התרגיל:**

* תרגול של שימוש ב־useState
* שינוי ערך משתנה (state) בלחיצת כפתורים
* עבודה עם onClick
* שילוב של CSS בסיסי לעיצוב

### **📝 הוראות:**

1. פתח את קובץ App.js בפרויקט React שלך.
2. צור משתנה מצב (state) בשם count עם ערך התחלתי של 0, באמצעות useState.
3. הוסף שלושה כפתורים:  
   * כפתור שמגדיל את המספר בלחיצה (+)
   * כפתור שמקטין את המספר בלחיצה (-)
   * כפתור שמאפס את המספר (Reset)
4. הצג את הערך של count בתוך כותרת (<h2>).
5. עצב את הכפתורים:  
   * ל־+
   * ל־-
   * כחול ל־Reset

### **🧪 תרגיל 1: החלפת טקסט בלחיצה**

**הוראות:**

* צור משתנה מצב בשם message עם ערך התחלתי "Hello".
* הוסף כפתור שכאשר לוחצים עליו, אם הערך הוא "Hello", הוא משתנה ל־"Goodbye", ואם הוא "Goodbye", הוא משתנה חזרה ל־"Hello".
* הצג את message בכותרת (<h1>).

### **🧪 תרגיל 2: הצגת/הסתרת טקסט**

**הוראות:**

* צור משתנה isVisible מסוג boolean, מאותחל ב־true.
* הוסף כפתור שמחליף את הערך שלו (true ↔ false) בכל לחיצה.
* אם isVisible הוא true, הצג טקסט כלשהו.
* אם false – הסתר אותו. (״״)

### **🧪 תרגיל 3: הצגת שם משתמש**

**הוראות:**

* צור שדה input לקליטת שם.
* שמור את הערך ב־useState.
* הוסף כפתור "הצג שם".
* כאשר לוחצים על הכפתור – הצג את השם שנכתב.
* אם השדה ריק, הצג: "Please enter your name".

**🧪 תרגיל 4: סכום של שני מספרים**

**הוראות:**

* צור שני שדות input type="number".
* קבל מהם שני מספרים.
* כשלוחצים על כפתור "חשב", הצג את הסכום של שני המספרים.
* דאג להמיר את הערכים ממחרוזת (string) למספר (Number() או parseFloat).

### **🧪 תרגיל 5: בדיקת גיל**

**הוראות:**

* צור שדה input type="number" שבו המשתמש יכניס את הגיל שלו.
* כאשר המשתמש לוחץ על כפתור "בדוק", הצג הודעה על פסקה:  
  + אם הגיל קטן מ־18 → "קטין"
  + אם הגיל 18 ומעלה → "בוגר"

### **🧪 תרגיל 6: סיסמה כפולה**

**הוראות:**

* צור שני שדות input type="password": סיסמה ואימות סיסמה.
* כשלוחצים על כפתור "בדוק התאמה", הצג:  
  + "הסיסמאות תואמות" – אם שתיהן זהות
  + "הסיסמאות לא תואמות" – אחרת