#### **🧱 יחידה 1: מבוא לריאקט**

## **מה זה React ולמה להשתמש בה**

## **יצירת פרויקט ראשון עם Vite או Create React App**

## **מבנה קבצים בסיסי בפרויקט ריאקט**

## **כתיבת קומפוננטה בסיסית (Component)**

## **JSX – כתיבה דמוית HTML בתוך JavaScript**

## 

#### **🧮 יחידה 2: Props ו-State**

## **שימוש ב־Props להעברת מידע לקומפוננטות**

## **שימוש ב־useState לניהול מצב (state)**

## **אירועים (onClick, onChange)**

#### **🔁 יחידה 3: תנאים ולולאות**

## **תנאים ב־JSX (if, ternary)**

## **שימוש ב־map לרינדור רשימות**

## **שימוש ב־key**

## **📝 תרגול: אפליקציה שמציגה רשימת מוצרים/משימות/סטודנטים**

## 

#### **🧩 יחידה 4: קומפוננטות מורכבות והרכבה**

## **הפרדת הקוד לקומפוננטות שונות**

## **תקשורת בין קומפוננטות (props, events)**

## **📝 תרגול: רשימת משימות עם יכולת הוספה ומחיקה**

## 

#### **🔄 יחידה 5: useEffect ו- Lifecycle**

## **מה זה useEffect ולמה הוא חשוב**

## **מתי useEffect רץ (תלות במערכים)**

## **אפקטים כמו טעינת נתונים או טיימרים 📝 תרגול: אפליקציה שמציגה את השעה הנוכחית ומתעדכנת כל שנייה**

## 

#### **🌐 יחידה 6: שליפת נתונים מהשרת (API)**

## **שימוש ב־fetch או axios**

## **שליפת נתונים וטעינתם לרכיבים**

## **טיפול בשגיאות וטעינה 📝 תרגול: הצגת משתמשים מ־https://jsonplaceholder.typicode.com/users**

## 

#### **🎨 יחידה 7: עיצוב עם CSS**

## **עיצוב קומפוננטות עם CSS**

## **מודולים של CSS**

## **ספריות עיצוב: Tailwind, Bootstrap, Material UI 📝 תרגול: עיצוב רשימת מוצרים עם כפתורים ותגובות**

## 

#### **📦 יחידה 8: ניהול נתונים מתקדם (useContext)**

## **מה זה Context API ולמה צריך אותו**

## **יצירת Provider ו-Consumer**

## **שיתוף נתונים בין קומפוננטות ללא props**

## **📝 תרגול: ניהול מצב משתמש מחובר (login/logout)**

## 

#### 

#### **📁 יחידה 9: ניתוב (Routing)**

## **שימוש ב־react-router-dom**

## **יצירת דפים שונים (Routes)**

## **ניווט בין דפים (Link, useNavigate)**

## **פרמטרים ב־URL**

## **📝 תרגול: אפליקציית מוצרים עם דף ראשי ודף פרטי מוצר**

## 

#### **🚀 יחידה 10: פרויקט גמר**

## **רעיונות לפרויקטים:**

## **אפליקציית ניהול משימות**

## **חנות אונליין קטנה**

## **אפליקציית שאלות ותשובות**

## **לוח זמנים או מערכת שעות**

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

**חלק 1**

## 

## **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 (component name : HideText )**

**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 (component name = SumNumbers)**

**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. (CheckAge)
* 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

**חלק 2**

### 📚 חלק 1: Props – העברת מידע בין קומפוננטות

#### הסבר:

ב־React כל קומפוננטה יכולה לקבל Props (קיצור של "properties") – ערכים המועברים אליה מהקומפוננטה ההורה.

#### דוגמה:

function Greeting(props) {

return <h2>Hello, {props.name}!</h2>;

}

function App() {

return <Greeting name="Ariel" />;

}

### **✅ קומפוננטת מונה שמקבלת ערך התחלתי דרך props:**

import { useState } from 'react';

// קומפוננטת Counter שמקבלת ערך התחלתי דרך props

function Counter(props) {

const [count, setCount] = useState(props.start); // שימוש בערך המתקבל מהורה

return (

<div>

<h2>Current count: {count}</h2>

<button onClick={() => setCount(count + 1)}>+</button>

<button onClick={() => setCount(count - 1)}>-</button>

</div>

);

}

// קומפוננטת App שמעבירה את ערך ההתחלה

function App() {

return <Counter start={5} />;

}

export default App;

### **📌 הסבר קצר:**

* הקומפוננטה Counter מקבלת prop בשם start.
* ערך התחלתי של state (useState) נקבע לפי props.start.
* לחיצות על הכפתורים מגדילות או מקטינות את המונה.

### **🧪 תרגיל 3: הצגת שם משתמש באמצעות קומפוננטה**

#### **🎯 מטרת התרגיל:**

לתרגל שימוש ב־useState, עבודה עם טופס, ולבנות קומפוננטה שמגיבה ללחיצה על כפתור.

#### **📝 הוראות:**

1. צור קומפוננטה חדשה בשם UserNameDisplay.
2. בתוך הקומפוננטה:  
   * הוסף שדה קלט (input) לקבלת שם המשתמש.
   * שמור את הערך המוקלד בעזרת useState.
   * הוסף כפתור עם הכיתוב **"הצג שם"**.
   * כאשר לוחצים על הכפתור:
     + אם השדה ריק – הצג הודעה: **"Please enter your name"**.
     + אם הוזן שם – הצג: **"Hello, [השם שהוזן]"**.
3. הוסף את הקומפוננטה לקובץ App.jsx והעבר לה prop בשם emptyMessage עם הערך "Please enter your name".

#### **💡 טיפים:**

* השתמש ב־useState לשמירת הטקסט ולתוצאה שתוצג.
* השתמש ב־onChange כדי לעדכן את ה־state.
* השתמש ב־onClick של הכפתור כדי לבדוק ולהציג את ההודעה.

### **🧪 תרגיל 4: סכום של שני מספרים – קומפוננטה עם prop**

#### **🎯 מטרת התרגיל:**

לתרגל שימוש ב־useState, קליטת קלט מספרי מהמשתמש, ופעולה מתמטית פשוטה – **סכימה**, תוך שימוש ב־**prop** להעברת טקסט כפתור מבחוץ.

#### **📝 הוראות:**

1. בנה קומפוננטה חדשה בשם: SumCalculator.
2. הקומפוננטה תקבל prop בשם buttonText – הכיתוב שיוצג על הכפתור.
3. בתוך הקומפוננטה:
   * צור **שני שדות קלט** (input) מסוג number.
   * שמור את שני הערכים בעזרת useState.
   * הוסף כפתור, שבכיתוב שלו ישתמש ב־buttonText המתקבל מה־props.
   * כאשר לוחצים על הכפתור:
     + המר את שני הערכים למספרים (באמצעות Number() או parseFloat()).
     + חשב את הסכום.
     + הצג את התוצאה על המסך (בתוך תגית <p> או <h3>).

**✅ דוגמה לשימוש בקומפוננטה:**

**<SumCalculator buttonText="חשב סכום" />**

### **🧪 תרגיל 5: בדיקת גיל – קומפוננטה עם props להודעות**

#### **🎯 מטרת התרגיל:**

**לתרגל קלט מספרי, תנאים (if), וניהול state, תוך שימוש בפרופס להעברת טקסטים מהורה.**

#### **📝 הוראות:**

1. **צור קומפוננטה חדשה בשם AgeChecker.**
2. **הקומפוננטה תקבל שלושה props:**
   * **buttonText: טקסט שיוצג על הכפתור (למשל "בדוק").**
   * **minorMessage: ההודעה שתוצג אם הגיל קטן מ־18 (למשל "קטין").**
   * **adultMessage: ההודעה שתוצג אם הגיל 18 ומעלה (למשל "בוגר").**
3. **בתוך הקומפוננטה:**
   * **צור שדה קלט (input type="number") לקליטת הגיל.**
   * **שמור את הערך בעזרת useState.**
   * **כאשר לוחצים על הכפתור:**
     + **בדוק את הגיל:**
       - **אם הגיל < 18 → הצג את minorMessage.**
       - **אחרת → הצג את adultMessage.**

#### **✅ דוגמה לשימוש בקומפוננטה:**

**<AgeChecker**

**buttonText="בדוק"**

**minorMessage="קטין"**

**adultMessage="בוגר"**

**/>**

### **🧪 תרגיל 6: סיסמה כפולה – קומפוננטה עם props**

#### **🎯 מטרת התרגיל:**

**לתרגל עבודה עם קלטים מסוג password, השוואת מחרוזות, וניהול מצב באמצעות useState, תוך שימוש ב־props להעברת טקסטים מהורה.**

#### **📝 הוראות:**

1. **צור קומפוננטה חדשה בשם PasswordMatchChecker.**
2. **הקומפוננטה תקבל 3 props:**
   * **buttonText: הטקסט שיופיע על הכפתור (לדוגמה: "בדוק התאמה").**
   * **matchMessage: הודעה שתוצג כאשר הסיסמאות זהות (לדוגמה: "הסיסמאות תואמות").**
   * **notMatchMessage: הודעה שתוצג כאשר הסיסמאות שונות (לדוגמה: "הסיסמאות לא תואמות").**
3. **בתוך הקומפוננטה:**
   * **צור שני שדות קלט (input type="password") – אחד לסיסמה ואחד לאימות סיסמה.**
   * **שמור את הערכים עם useState.**
   * **כאשר לוחצים על הכפתור:**
     + **אם הסיסמאות זהות → הצג את matchMessage.**
     + **אחרת → הצג את notMatchMessage.  
       ✅ דוגמה לשימוש בקומפוננטה:**

**<PasswordMatchChecker**

**buttonText="בדוק התאמה"**

**matchMessage="הסיסמאות תואמות"**

**notMatchMessage="הסיסמאות לא תואמות"**

**/>**

#### **💡 טיפים:**

* **ניתן להשתמש ב־trim() כדי להתעלם מרווחים מיותרים.**