React – reusability

Spa – single page application

זהו קובץ התחלתי שמכיל את כל הדפים בתוך קובץ של HTML.

קבצים חובה להרצה:

* ספריות של NODE MOUDULES
* PUBLIC
* MAIN.JSX
* INDEX.HTML
* PACKAGE.JSON

חוק ראשון: חייב להיות "אבא" או אלמנט שעוטף.

חוק שני: כדי לכם להתסכל על JSX כטיפוס נוסף בשפה, תחשבו שהוסיפו לנו עוד טיפוס. מה זה נותן לנו? מאפשר לנו לשלוח אותו כפרטמרים לפונקציות נוספות. לדוגמא :

תמונה שמכילה טקסט, צילום מסך, גופן

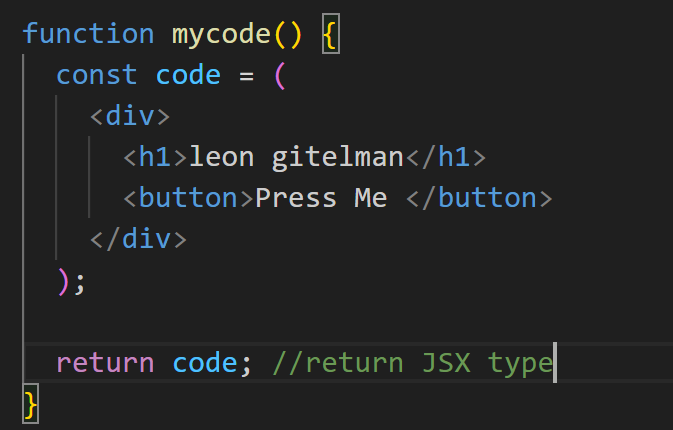
התיאור נוצר באופן אוטומטי

חוק שלישי: כאשר רוצים להכניס ערך של ביטוי של JS אני צריכה לכתוב את זה אחרת. צריכים לכתוב את זה בסוגריים מסולסלים. Interpolation הכנסת ערך ביטוי של JS .

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

כל פעם שאנחנו רוצים לשתול JS צריכים לשתול את זה בסוגריים מסולסלים. נעשה את זה כל הזמן. האינטרפולציה היא ייחודית עבור שפת REACT.

כל משתנה אנחנו נגדיר אותו בCONST .

קומפוננט – זה למעשה אובייקטים כמו כפתורים כותרות. כדי להגדיר אותה נצטרך ליישם 2 חוקים:

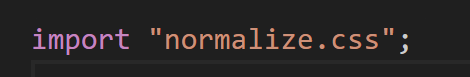
אפשר לבנות על ידי CLASS כמו בJAVA או שאפשר לייצר אותם באמצעות פונקציות. קוראים לה functional component כדי שהיא תהפוך לקומפוננט צריך לכתוב אותה באמצעות אות **גדולה בהתחלה** .

חוקים:

1. שם פונקציה אמורה להתחיל עם אות גדולה
2. חייב להחזיר קובץ JSX
3. לקרוא לפונקציה הראשית בשם App

ברגע ששני החוקים האלה מתקיימים, מתקיים לנו מצב של קומפוננטה. קומפוננטה זה כמו אלכמנט חדש של HTML. מקובל לקרוא לקומפוננטה הראשונה APP.

**פונקציה ראשית - App** יכיל את מבנה האתר

**מה זה :**

התפקיד שלו זה לאפשר תצוגה שווה לכל המשתמשים, כלומר התפקיד שלה זה למנוע הבדלים בין כל הדפדפנים. מכיוון שכל הדפדפנים אינם כתובים באותו אופן ויש הבדלים בין כל דפדפן לדפדפן. אנחנו חייבים להשתמש בפקודה של IMPORT NORAMLIZE.CSS

שיעור 4 - **אנחנו נרצה כל קומפוננט בקובץ נפרד- זה נושא ההרצאה**

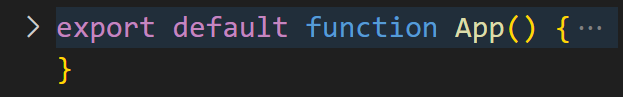
**סיכום עד עכשיו**

קומפוננט פונקציונלית- זוהי תבנית שמגדירה בלוק HTML,CSS,JS וממנה מייצרים אובייקטים. היא צריכה להתחיל עם **אות גדול** והיא חייבת להחזיר קובץ JSX . איך אנחנו מייצרים אובייקטים ממנה.

מקובל שכל קומפוננטה נכתוב בקובץ נפרד. זה אומר שAPP צריך להגדיר אותו בקובץ נפרד. כלומר נוציא אותה החוצה.

אנחנו נקרא מקבצים אחרים באמצעות פונקציות EXPORT AND IMPORT.

סוגי EXPORTS

1. רגיל – כמות לא מוגבלת לכל מיני נתונים פונקציות
2. DEFAULT – רק אחד לקובץ עבור קומפונטטה

**תמונה שמכילה גופן, טקסט, צילום מסך, גרפיקה

התיאור נוצר באופן אוטומטי**

כאשר מייבאים קובץ, אפשרי לוותר על סוגריים { } מכיוון שזה קובץ DEFAULT . המערכת מזהה שיש רק אחד לכן יהיה אפשר להשתמש בו.

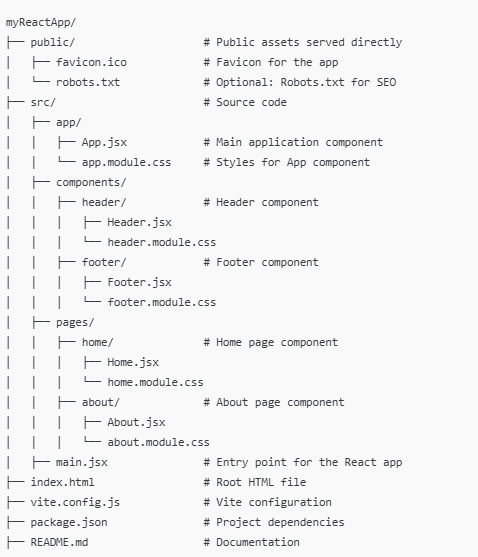
עיצובים:

PADDING עושים באמצעות יחידות של EM וREM

כדי שלא יהיה בלגן CSS אנחנו לא נכתוב בוררים של BUTTONש

**שיעור 5**

MAIN לא עושים כקופננוטה, מכיון שזה יעזור לנו בעתיד בעיצוב, לכן משאירים אותו כמו שהוא. אבל זה לא אומר שבעתיד נצטרך לעבוד כך.

מבנה של REACT

**אנחנו נתחיל עם הפרדה של HEADER :**

אנחנו הולכים לעשות HEADER . כל התיקיות של קומפוננטים צריכים להכניס לתיקייה ראשית של קומפוננט. לא חובה, אבל ככה במבחן צריך לעשות.

כאשר אנחנו רוצים לייבא קומפוננטה מבחוץ, אין צורך לעשות IMPORT אפשר ישירות לקרוא לה והתוכנה תוסיף באופן אוטומטי את הIMPORT. כאשר אנחנו רוצים לחזור לקובץ הראשי אנחנו נצטרך להשתמש ב ".. " .

**איך עושים IMPORT יותר מהיר:**

מתחילים לכתוב את השם שלה, וברגע שכותבת את השם מספיק ארוך, מתחילים לראות אופציות שהמערכת נותנת לנו.

CSS

כאפשר יש לנו FIRST-CHILD זה סוג של בוררים בשפע. כאשר יש לנו SPAN מאותו סוג אז זה מאפשר לנו לבחור איזה מהם לשנות.

APP

איך לעצב את הקובץ של APP .

  display: flex;- תצוגה בשורה

  flex-direction: column;- תצוגה בעמודות

איך להוריד את הFOOTER למטה

תמונה שמכילה טקסט, צילום מסך, תצוגה, תוכנה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, תוכנה, תצוגה

התיאור נוצר באופן אוטומטישיטה יותר טובה:

סיכום : אם יש לנו שלושה אלמנטים HEADER FOOTER MAIN . כאשר אנחנו רוצים לדחוף משהו לתחתית האתר אנחנו צריכים להגדיר את כל הקונטייר לFLEX ואז לעשות אותו בעמודות COLUMN ואז צריך להגדיר לו גובה מינמלי בעזרת MIN-HEIGHT. ואז הולכים לבלוק המרכזי ועושים לו FLEX 1 זה למעשה מותח את כל הכול לכל העמוד.

מה זה אינטרפולציה? זה קטע קוד שהוא ביטוי שמחזיר ערך בשפת JS. כותבים אותה בתוך סוגריים מסולסלים. יכולים להיות מספר ערכים בביטוי:

* ערך מספרי מסוג 'מספר'.
* ערך של מחרוזת.
* משתנים בוליאנים אפשר לכתוב אבל לא נראה את זה. לכן, רצוי לא לעשות את זה.
* מערך --> רשימות

שגיאות באינטרפולציה:

* אסור לכתוב אובייקטים במבחן!
* אסור לכתוב תנאים, אך ורק ביטויים!

מערכים בREACT

הוא יודע להציג מערכים, כאשר יש בתוך המערך ערכים מסוג HTML הטיפוס של המערך הוא טיפוס מסוג JSX.

UNIQUE KEY

בכל תא צריך להגדיר KEY כדי שכל תא יהיה ייחודי, כאשר הרשימה היא סטטית ולא משתנה, כלומר אף שם לא התווסף ואף שם לא יימחק אפשר לעבוד עם אינדקס כי כל אינדקס הוא ייחודי. למרות זאת, בהמשך אנחנו נעבוד עם רשימות דינמיות לכן, האינדקס לא יהיה רלוונטי.

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי

כל המשתנים ושמות פונקציה בcamelCase

כאשר אנו עובדים עם פונקצית MAP תמיד נשאוף לרשום RETURN .

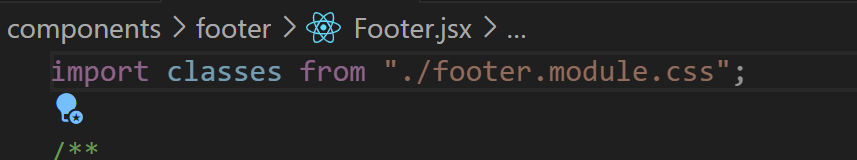
CSS

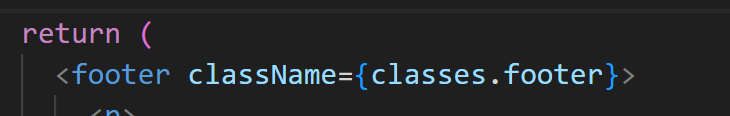
הindex.CSS ראשי הוא הכי חלש, מכיוון שכל CSS אחרי הוא דורס את ההגדרות שלפניו. כל קובץ שעושים לו IMPORT כולם מכירים אותו. לכן לסדר של IMPORT יש משמעות. זאת אומרת שכולם מכירים את כולם, ויש עדיפות לסדר. זה יוצר בלגן. אנחנו לא רוצים לזכור את הסדר. לכן אנחנו צריכים לעבוד עם CLASS כדי שלא יהיה בלגן. זה פתרון אחלה, בתנאי ששמות הCLASS שונים. עדיין צריכים להיות נורא זהירים, מכיון שאנחנו רוצים למנוע התנגשות. בעיה נוספת: לא תמיד אפשר לתת . CLASS NAMESלכן יש לנו פתרון עם מודולים.

תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

css – modules

צריך לשנות את השם של קובץ CSS

 לאחר ששינינו את הקבצי CSS אנחנו ניגש לקובץ JSX כדי לשנות את ה CLASS על ידי שימוש באינטרפולציה.

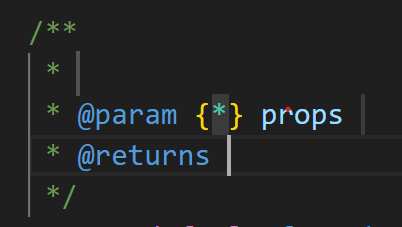
הערות לפני המבחן:

כל קומפוננט מעצבת את עצמה לכן כאשר נעצב, נעצב רק את הCSS .

גם את הpadding עושים בQUOTE אם רוצים לעשות MARGIN עושים את זה באבא ב.APP מכיוון margin נחשב לסידור כללי של האירוע, לכן אנחנו נשים את זה באבא.

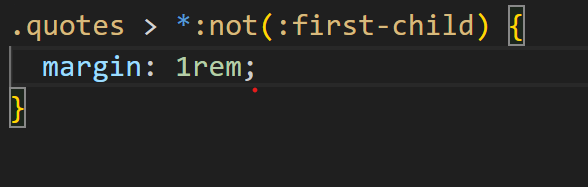
מי אחראי על הצגת האובייקטים component הprops . כאשר אנחנו נרצה לייבא נתונים מקובץ חיצוני נצטרך לכתוב import {quotes} from '../ file name'

**לא לשכוח לכתוב תיאור פונקציה**



אפשר לפנות לאובייקטים בעזרת props.obj.cite או props.obj.quote. **אבל זה לא נהוג**.

טריק CSS אם אנחנו רוצים לפנות לכל הבנים תחת COMPONENTS אנחנו נצטרך לכתוב:



כאשר לא רוצים את הראשון, אבל רוצים את כל השאר. נרשום ב.CSS

עבודה עם אירועים Events בReact

תמונה שמכילה טקסט, צילום מסך, גופן

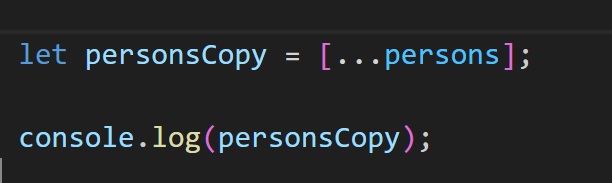
התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, גופן, צילום מסך

התיאור נוצר באופן אוטומטי

ההבדל הוא כאשר אנחנו כותבים את שם הפונקציה עם() היא תפעל אוטמטית ברגע שהכפתור נולד. כאשר אין סוגריים לפונקצייה הפונקציה תפעל אך ורק כאשר לוחצים על האלמנט ומפעילים את הEVENT

איך מוחקים מערך בREACT אנחנו יכולים למחוק אותו על ידי שאנחנו דורסים.

דרך טובה להעתיק מערך:

מוסיפים 3 נקודות ואז את השם המערך. ברגע שמשנים את המערך תוך כדי ריצה, התצוגה לא תשתנה באתר סטטי. לכן אנחנו צריכים לעבוד עם משתנים חדשים מיוחדים שנקראים: state variables . לכל קומפוננטה יש כמה מצבי חיים.

משתני מצב - התצוגה מתעדכנת כאשר:

* קומפוננטה נוצרה
* יש שינוי באחד מהערכים של PROPS
* משתנה אחד או יותר של stateמשתנה

אם אנחנו רוצים שהתצוגה שלנו תשתנה בתלות של משתנים, אז חייבים להגדיר את המשתנים כמשתני מצב.

איך מגדירים משתני מצב: תמיד צריך להגדיר אותם בתחילת הקומפוננטה.

תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

מחזיר מערך, הערך הראשון הוא שם של המשתנה, ומשתנה שני הוא הפנייה לפונקציה בשביל לשנות את TEXT. בתוך הפרטמר של הפונקציה אנחנו נכתוב ערך התחלתי. חשוב לייבא את הסיפרייה של useState

תמונה שמכילה טקסט, גופן, צילום מסך, קו

התיאור נוצר באופן אוטומטי

ברגע שעושים set למשתנה מצב, כל הפונקציה הראשית רצה מחדש.

איך מוסיפים אלמנטים עם משתני מצב:

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, גופן, צילום מסך, קו

התיאור נוצר באופן אוטומטי

**איך עושים setters**

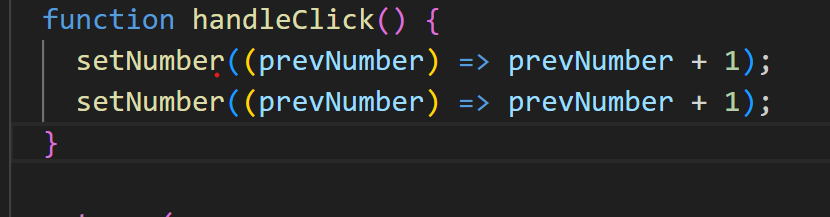
במידה ואנחנו מגדירים משתנה בתוך בsection הוא חייב להיות משתנה מסוג state .

יש אפשרות גם לקרוא לפונקציה. ברגע שעושים Set הערך החדש לא תלוי במה שהוצג קודם.

פונקציות של set הם פונקציות async . כלומר אם יש פונקציות sync לפניהם הם יבוצעו לפניהם.

**מה עושים כאשר הערך החדש תלוי בערך הישן:**

כותבים פונקציה אנונימית prenumber הוא readOnly



זה הדרך לכתוב פונקציה אנונימית

תמונה שמכילה טקסט, צילום מסך, גופן, קו

התיאור נוצר באופן אוטומטיבמידה ורוצים לקדם כל פעם באחד אבל בטווח של 1 עד 6

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

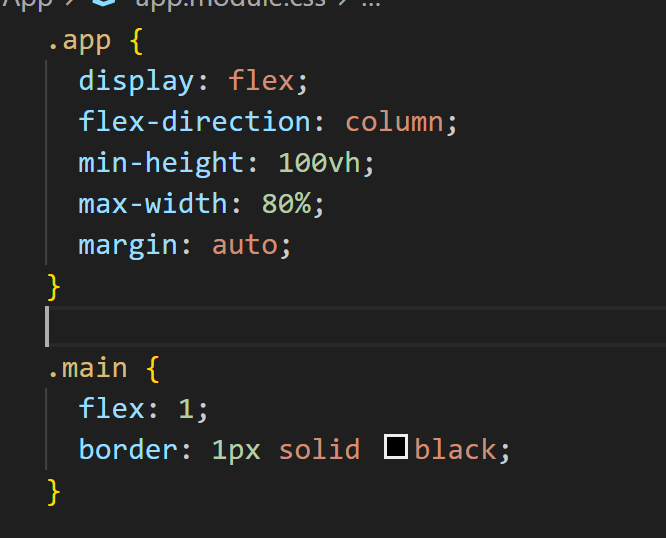
עבודה עם תיבות INPUT איך לקרוא מתיבה

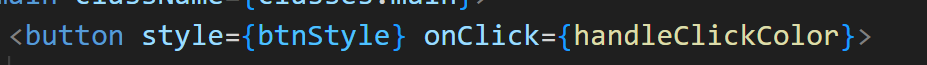
תמונה שמכילה טקסט, גופן, צילום מסך, קו

התיאור נוצר באופן אוטומטיכל פונקציה שעושה אירוע מקבלת אירוע מסוג E

תמונה שמכילה טקסט, צילום מסך, גופן, קו

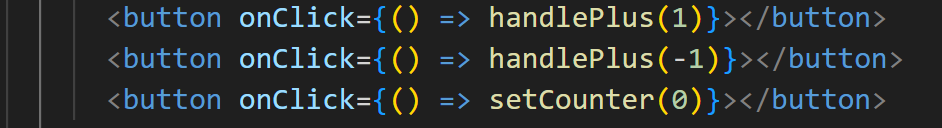
התיאור נוצר באופן אוטומטי

עיצוב ראשי של APP:

תמונה שמכילה גופן, צילום מסך, גרפיקה

התיאור נוצר באופן אוטומטילהגדיר STYLE כמשתנה:

קומפוננטה זה משהו לוגי שכולל גם עיצוב.

דרך לכתוב את פונקציה אחת תרגיל הCOUNTER

מייצרים פונקציה אחת ומעבירים לתוכה את הפרמטרים שאותם נרצה לשנות. חשוב לכתוב סוגריים לפני כדי ליצור הפנייה כדי לשלוח פרמטר. כך למעשה אנחנו לא משכפלים קוד.

למה אנחנו מייצרים הפניה?

**כי אם לא נייצר הפנייה הפונקציה תתחיל לעבוד אוטומטית, לכן אנחנו נייצר הפנייה =>(). כלומר מייצר פונקציה בתוך פונקציה. כלומר במקרה שלנו הפונקציה תעלה ותוריד ערכים שקיבלנו בפרטמר אוטמטית אילולא לא יצרנו הפניה.**

בuseState אנחנו לא נחזיק טיפוסים מסוג JSX אלא נחזיק משתנים של JS.

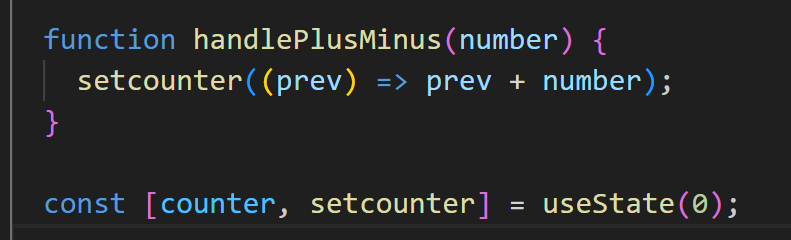
**כל האותיות והשמות של התיקיות חייבים להיות קטנות**

הקומנוטטה רצה בשלושה מצבים כאשר היא נולדה, פעם שנייה כאשר השתנה STATE והיא רצה עוד פעם כאשר יש שינוי בPROPS .

איך עושים כפתורים רספונסיבי :

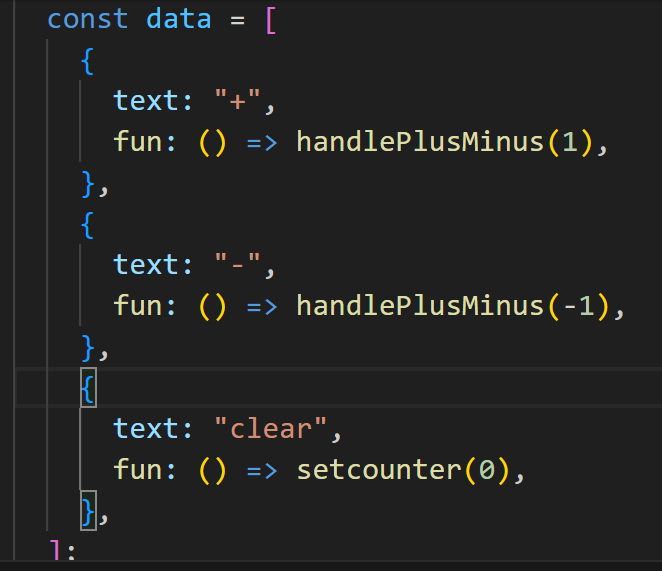


**תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטיאיך מעבירים פונקציה בתוך קופוננטה בתרגיל של COUNTER**

שינוי של props משפיע על התצוגה וראינו state lifting שמעבירים מאבא לבן פונקציה והבן מבצע את הפונקציה.

מערך של אובייקטים בREACT:



תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

**איך להגדיר סיפרייה של אייקונים:**

**npm I react-icons**

**תרגיל של Accordion**



תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

התיאור נוצר באופן אוטומטי

איך עובדים עם state ומערכים בreact:

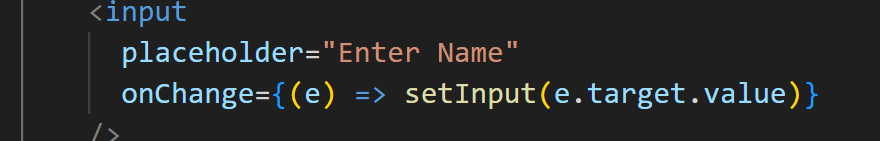
איך ניגשים לתיבה ?

הדרך הכי טובה זה לעקוב אחרי הקלדה של USER ולוחצים על + אז המשתנה כבר נשמר

e.target אלמנט של HTML שיש עליו שינוי.

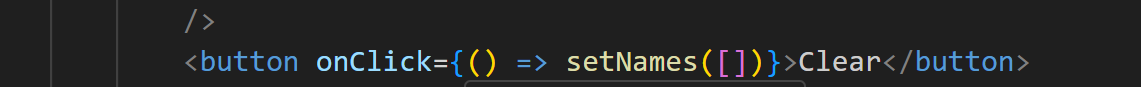
כאשר מתבצע מופע או שינוי REACT יכול להעביר הפנייה לאובייקט. בתוך האובייקט יש אינדקציה על מי עבר האירוע ((TARGET כדי להגיע לערך value

**תרגיל של הוספה/ הסרה רשימת שמות באמצעות state**

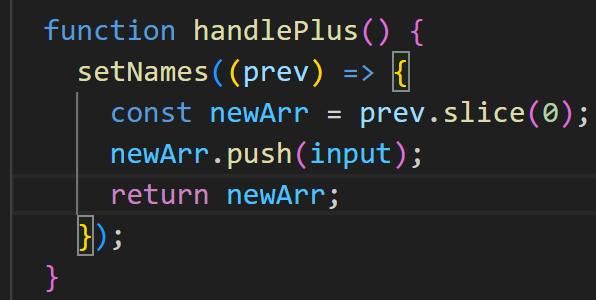


תמונה שמכילה טקסט, גופן, צילום מסך, כתב יד

התיאור נוצר באופן אוטומטי



להוסיף לרשימה על ידי שימוש בSTATE של input

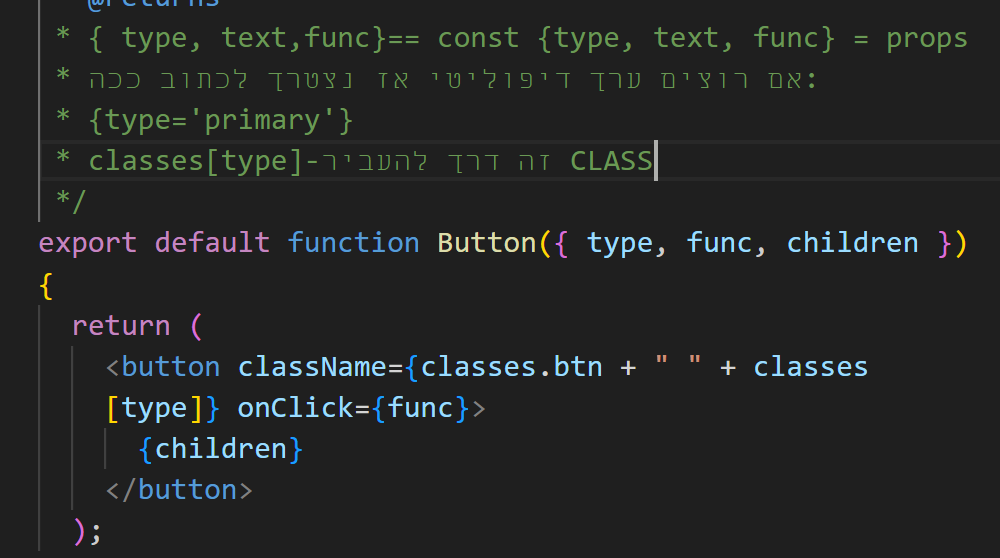


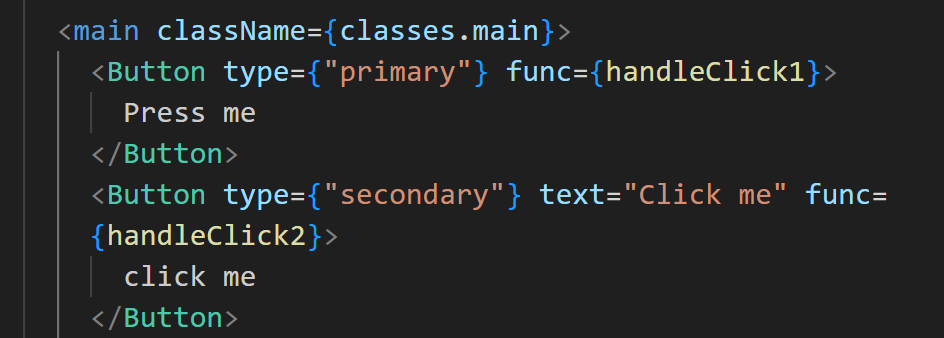
איך למחוק אובייקט מתוך מערך באמצעות שימוש בפונקציה filter()

**איך לעצב כפתורים שונים בreact בעזרת הוספה של class:**

בקטע קוד הבא ניתן לראות כיצד מעבירים לכל כפתור מחלקה שונה של css באמצעות שימוש בProps. כיצד לחבר בין 2 מחלקות שונות ומעבר: אנחנו משרשרים מחרוזות כמו כן אפשר להשתמש ב` ` .

בתוך הפרמטרים של button הכנסו את הProps שמועבר מAPP . כלומר זה עוד דרך לקרוא לprops.: הדרך המקורית שלמדנו היא: const {type,func,children} = props

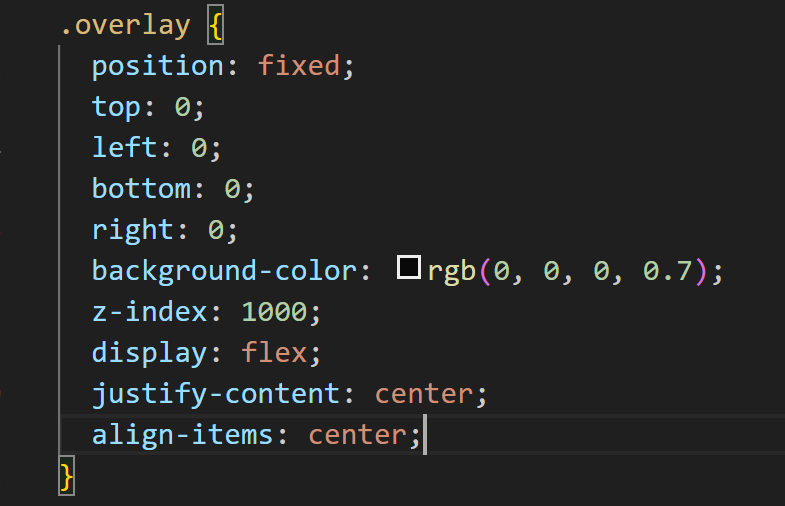
****

**שימוש בChildren :**

**חלון POPUP**

**תמונה שמכילה טקסט, צילום מסך, גופן, מספר

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.Css:**

****

Fixed- שכבה שעולה על הכול

z-index : מוודא שהיא תיהיה השכבה העליונה ביותר.

איך ממרכזים את התוכן **תמיד דרך האבא :**

* display: flex;
  + - * justify-content: center;
      * align-items: center;

**תמונה שמכילה טקסט, צילום מסך, גופן

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.**

**תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.תמונה שמכילה טקסט, צילום מסך, תוכנה, תצוגה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.**

חזרה למבחן:

1. לייצר 2 קומפוננטות שבלחיצת כפתור מוצגת הקומפוננטה הרלוונטית:



1. תרגיל 2 : יש תיבות של טקסט

+

=

import classes from "./app.module.css";

import Footer from "../components/Footer/Footer";

import Header from "../components/Header/Header";

import Button from "../components/button/Button";

import Modal from "../components/modal/Modal";

import { useState } from "react";

function Home() {

  return (

    <div>

      <h2>Home</h2>

      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>

    </div>

  );

}

function About() {

  return (

    <div>

      <h2>About</h2>

      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>

    </div>

  );

}

export default function App() {

  const [comp, setcomp] = useState("Home");

  const [result, setResult] = useState("");

  const [value1, setValue] = useState();

  const [value2, setValue2] = useState();

  const [act, setact] = useState();

  function handleValue1(e) {

    setValue(e.target.value);

  }

  function handleValue2(e) {

    setValue2(e.target.value);

  }

  function handleAct(e) {

    setact(e.target.value);

  }

  function calcReuslt() {

    if (act === "+") setResult(value1 \* 1 + value2 \* 1);

    if (act === "-") setResult(value1 \* 1 - value2 \* 1);

  }

  return (

    <div className={classes.app}>

      <Header />

      <main className={classes.main}>

        <button onClick={() => setcomp("Home")}>Home</button>

        <button onClick={() => setcomp("About")}>About</button>

        {(comp === "Home" && <Home />) || (comp === "About" && <About />)}

        <p>Ex2:</p>

        <div>

          <input type="text" onChange={(e) => handleValue1(e)} />

          <input type="text" onChange={(e) => handleAct(e)} />

          <input type="text" onChange={(e) => handleValue2(e)} /> {result}

        </div>

        <button onClick={() => calcReuslt()}>calc</button>

      </main>

      <Footer />

    </div>

  );

}

לא חייב להחזיק תיבות וטפסים בstate :