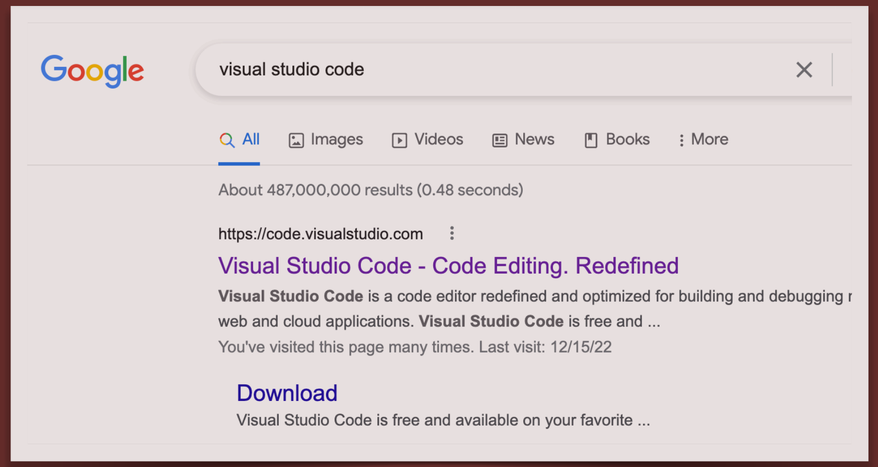
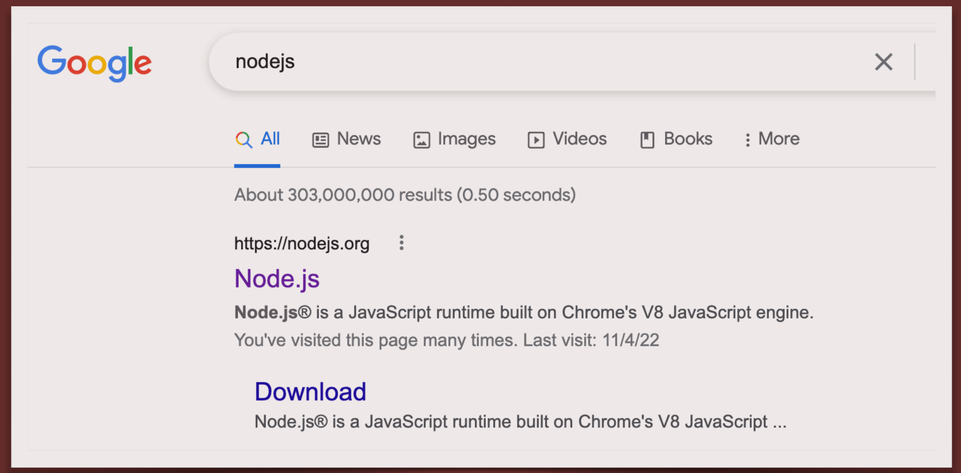
Cossci

סביבת העבודה



עורך קוד VS Code

אנחנו נשתמש ב VS code
העורך הכי פופולארי לבניית אפליקציות
אינטרנטיות



סביבת פיתוח NodeJS

מאפשר להשתמש בפקודות "NPM"
node-package-manager
בכדי להתקין ספריות וכלים

Cossci

למי מתאים הקורס?

למתכנתי אתרים אשר מכירים HTML,CSS,Javascript ומעוניינים ללמוד React

למי שרוצה ללמוד לבנות אתרים איכותיים ודפי נחיתה

לאנשים המכירים את React ומעוניינים להרחיב את הידע שלהם

למי שרוצה ללמוד להשתמש בספריות הכי עדכניות בפיתוח SPA

Cossci

תוכן עניינים

React State
ניהול מצבים והגבה לאירועים בדף

React Components & JSX
שימוש ב Class Components
ושימוש ב Function Components

React Hooks
שימוש בפעולות מיוחדות לפתרון
בעיות שכיחות ביצירת אתר

עיצוב
שימוש בספריות עיצוב Material UI

ניווט באתר
מעבר בין עמודים
שיתוף מידע בצורה נכונה בין עמודים

בקשות רשת
בקשות רשת בReact וקצת על הספריה axios

Firebase
שימוש בכלים של גוגל
לאימות משתמשים, אחסון מידע
והעלאת האתר לאוויר

Cossci

שפת JSX

מה זה JSX

הרחבה של השפה Javascript
שפותחה ע"י פייסבוק כחלק
מפיתוח ריאקט

מה מיוחד בשפה

השפה מאפשרת כתיבה של HTML
בתוך קבצי Javascript שעוברת
קומפילציה ע"י הספריות של React

```
function Navbar() {
    return <nav>
        <a>Home</a>
        <a>About</a>
        <a>Login</a>
    </nav>
}
```

פונקציית Javascript רגילה

המחזירה HTML

Cossci

React Components

כמו שראינו, בJSX אפשר לכתוב פעולות שמחזירות HTML
פעולות כאלה נקראות :
React Functional Components

המהדר של **React** מפרש את הפעולות
ומכניס את האלמנטים לעמוד ה **HTML**

```
function Navbar() {
    return <nav>
        <a>Home</a>
        <a>About</a>
        <a>Login</a>
    </nav>
}
```

סרגל ניווט

```
function CountryList() {

    return <ul>
        <li>Israel</li>
        <li>Germany</li>
        <li>Russia</li>
        <li>China</li>
    </ul>
}
```

רשימת ארצות

```
function MyApp() {
    return <div>
        <CountryList />
    </div>
}
```

שימוש בReact Function Component שיצרנו
בReact Function Component אחר

Cossci

React Components

אנחנו נכתוב React Components והמהדר של ריאקט יכניס
את תוכן הHTML לדף עצמו, אך יש את האלמנט הראשי שתחתיו אנחנו
מכניסים את שאר האלמנטים

האלמנט ה"ראשי" אשר עולה לדף
ולתוכו נכנסים שאר האלמנטים

index.js

```
const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App/>);
```

```
function App() {

    return <div>
        <h1>Hello, welcome to My React App !</h1>
        <About/>
    </div>
}
```

React Component
העוטף את כל האפליקציה

כל התוכן של <App/>
נכנס לתגית ה root

```
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>

</body>
```

דף ה index.html עצמו, אשר אליו ייכנסו האלמנטים בסופו של דבר,
אנחנו לא עורכים בדרך דף זה בצורה ידנית

Cossci

React Components

Class Component

אפשר לכתוב את אותו אלמנט
בצורת Class Component

```
class Navbar extends React.Component {

    render() {
        return <ul>
            <li>Israel</li>
            <li>Russia</li>
            <li>Ireland</li>
            <li>Germany</li>
        </ul>
    }
}
```

Function Component

הצורה הסטנדרטית בה נכתוב React Components
(בצורת פונקציות JS רגילות)

```
function Navbar() {
    return <nav>
        <a>Home</a>
        <a>About</a>
        <a>Login</a>
    </nav>
}
```

מה השיקול בבחירה?

בסופו של דבר
Class Components
מהודרים באותה צורה,
אנחנו לרוב נשתמש בfunctions
אבל יש שימושים רבים בClasses תוך שימוש בקונספטים
של תכנות מונחה עצמים

Cossci

JS בתוך Components

סיטואציה:
עלינו להציג מידע שהוגדר במשתנה
חיצוני בComponent

איך מכניסים את תוכנו לדף?

שימוש בבלוק Javascript

```
function Title() {
    const myTitle = "Welcome to my React App!"
    return <h1>
        {myTitle}
    </h1>
}
```

(2)

```
function CountryList() {
    const countries = ["Egypt","Israel","Germany","Thailand"]
    return <ul>
        <li>{countries[0]}</li>
        <li>{countries[1]}</li>
        <li>{countries[2]}</li>
        <li>{countries[3]}</li>
    </ul>
}
```

(1)

מה שנמצא בגבולות ה {} בתוך ה JSX
מתפרש כ Javascript, לכן ניתן להכניס שם
את המשתנה החיצוני myTitle

```
function CountryList() {
    const countries = ["Egypt","Israel","Germany","Thailand"]
    return <ul>
        {countries.map(country => <li>{country}</li>)}
    </ul>
}
```

(3)

דרך נוספת לאותה תוצאה כמו (1)
תוך שימוש ב map

Cossci

HTML Attributes in react

תכונות של אלמנטים HTML בריאקט
שונים במקצת מדפי html רגילים

כמה דוגמאות:

CSS Class

```
function Navbar() {
    return <nav class="nav_bar">
            <a>Home</a>
            <a>About</a>
            <a>Login</a>
    </nav>
}
```

בדר"כ היינו מגדירים **class** לאלמנט **HTML**
עי שימוש בתכונה **class**,
אך זה לא יעבוד בריאקט

```
function Navbar() {
    return <nav className="nav_bar">
            <a>Home</a>
            <a>About</a>
            <a>Login</a>
    </nav>
}
```

בריאקט משתמשים ב **className**

Event Listeners

```
function MyButton() {

    function clicked() {
        alert("The button was clicked")
    }

    return <button onclick="clicked()">
        Click me - WRONG!
    </button>
}
```

בדר"כ היינו מגדירים **onclick** (פעולת לחיצה) לאלמנט **HTML**
עי שימוש בתכונה **onclick** כמו בדוגמא לעיל
אך זה לא יעבוד בריאקט

```
function MyButton() {

    function clicked() {
        alert("The button was clicked")
    }

    return <button onClick={clicked}>
        Click me - Correct!
    </button>
}
```

בריאקט משתמשים ב**onClick** וב
{} ניתן להכניס את הפעולה בצורה כזו

Cossci

העברת מידע בין קומפוננטות

(מאפיינים) - props

```
function UserPage() {

    return <div>
        <Profile name="ron" age={18} />
    </div>
}
```

```
function Profile(props) {

    return <ul>
        <h2>{props.name}</h2>
        <p>{props.age}</p>
    </ul>
}
```

משתמשים בבלוק javascript
להציג את תכונות ה props

props הינו אובייקט המכיל את התכונות
שמועברים לקומפוננטה בצורת key:value
למשל בדוגמא לעיל

props = { name:"ron", age:18 }

Cossci

העברת מידע בין קומפוננטות

```
function BusinessCard( { name, phone } ) {

    return <div>
        <h1>B-Card</h1>
        <h3>{name}</h3>
        <span>Tel: {phone}</span>
    </div>
}


function Profile( { name, age, phone } ) {

    return <div>
        <h1>My profile</h1>
        <p>Lorem ipsom ..</p>
        <h2>You are {2023 - age} Years old!</h2>
        <BusinessCard name={name} phone={phone}/>
    </div>
}
```

השם טלפון וגיל עוברים מProfile ל BusinessCard

עוד דוגמא להעברת **props**
הפעם משתמשים ב**destructing**
על מנת לגשת לתכונות באופן ישיר

אפשר להעביר הלאה גם לעוד קומפוננטות
כמו בדוגמאות:

Cossci

תגובתיות לאירועים, ועדכון מצבים באמצעות React State

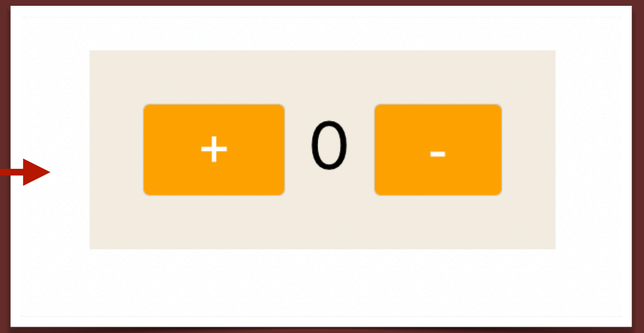
קומפוננטה המצייגת מונה עם כפתורים -,+

```
function Counter() {

  let counter = 0
  function increase() {
    counter++
  }
  function decrease() {
    counter--
  }

  return <div className='counter'>
    <button onClick={increase}>
      +
    </button>
    <div>{counter}</div>
    <button onClick={decrease}>
      -
    </button>
  </div>
}
```

משתנה counter
שכביכול משתנה
ע"י לחיצות על הכפתורים.
אבל אם תנסו להריץ את הקוד,
תראו שהאלמנטים במסך
לא מתעדכנים בלחיצות



נניח שברצוננו לבנות מונה
אשר בו ניתן להוסיף עם + ולהפחית עם -

נתבונן בדרך שגויה לעשות את זה בריאקט:

הבעיה:
עדכון המשתנה בפועל קורה בלחיצות על הכפתורים.
אך תוכן ה HTML בדף לא עובר הידור מחדש
ולכן הערך החדש אינו מופיע.

Cossci

מהי React Router ?

React Router היא ספריית תוסף ל-**React** שמאפשרת לנו לבנות אתרי אינטרנט עם תכונת הניווט מעודכנת באופן דינמי
זה מאפשר לנו להגדיר מסלולי ניווט באתר ולנהל את התנועה בין הדפים בעזרת הקומפוננטות של **React**.

כיצד ניתן להתקין את React Router?

ניתן להתקין את **React Router** עם אחת מהפקודות הבאות:

```
npm install react-router-dom
yarn install react-router-dom
```

כיצד ניתן להתחיל להשתמש ב-React Router?

כדי להתחיל להשתמש ב-**React Router**, נצטרך לעשות שתי דברים:

1. ניכנס לקומפוננטת המופע שמשתמשת ב-**React Router**.
נעשה זאת עם התכונה **import** של **JavaScript** ונימצא את הקומפוננטות שנרצה להשתמש בהן מתוך ספריית **React Router**.

2. נכניס את הקומפוננטות שנרצה להשתמש בהן לתוך המופע שלנו.
נעשה זאת עם התכונה **import** של **JavaScript** ונעניק לה את הקומפוננטות שנרצה להשתמש בהן

Cossci

ניווט - שימוש בספרייה react-router

כדי להתקין את הספריה מ
NPM נריץ:

npm install react-router
npm install react-router-dom

נבצע ייבוא לקומפוננטות החשובות

```
import {BrowserRouter,Routes,Route} from 'react-router'
```

נגדיר את הנתב ואת העמודים באתר

```
function App() {
  return (
    <div className="App">

      <BrowserRouter>

        <Routes>

          <Route path={'/home'} element={<Home />}/>

        </Routes>

      </BrowserRouter>

    </div>
  );
}
```

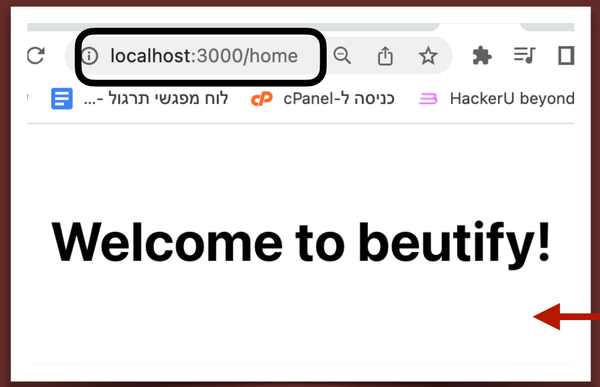
path
שם נתיב העמוד באתר

element
הקומפוננטה לעמוד הזה

בצורה הבאה מספקים את
קומפוננטת ה **BrowserRouter** (נתב)

ובתוכה **Routes** (נתיבים)
אשר מכילה את העמודים השונים

כל **Route** מיוצג בנתיב אחר כלומר דף אחר
/home, /favorites, /login . .




```
function Home({}) {
  return <div>
    <h2>Welcome to beutify!</h2>
  </div>
}
```

Cossci

מהי המצב(state) ב-React?

המצב הוא אחד מהנתונים העיקריים שנמצאים בתוך מופע מסוג **React**.
המצב מאפשר לנו לנהל נתונים בתוך המופע ולעדכן את המופע עם שינויים בנתונים אלה.

כיצד אנחנו משתמשים במצב?

כדי להשתמש במצב, נצטרך לעשות שתי דברים:
1. ניצור את המצב שלנו בתוך המופע שלנו.
נעשה זאת על ידי הגדרת משתנה עם ערך המצב הראשוני שלנו בתוך המופע.

2. נעדכן את המצב באמצעות פונקציית ה-**setState** במחלקות או באמצעות **hooks** מסוג **useState**.
הפונקציה הזאת מאפשרת לנו לעדכן את המצב עם ערך חדש ולדאוג לכך שהמופע יתעדכן בהתאם.

כיצד ניתן לגשת למצב מתוך המופע?

ניתן לגשת למצב מתוך המופע באמצעות המשתנה **this.state** בתוך המופע.
ניתן גם לגשת לפרטים ספציפיים בתוך המצב באמצעות **hooks** מסוג **useState**

Cossci

מה זה useEffect ?

useEffect ב- React מאפשר להפעיל פונקציות כאשר מערכת ההרשמה של התוסף משתנה. זה מאפשר לעשות דברים כמו לטעון נתונים משרת, להתאים איזור עניינים ל- DOM או להגדיר מאזינים לאירועים.

כדי להשתמש ב- useEffect, ראשית עלנו לייבא את הפונקציה מתוך החבילה react:

```
import { useEffect } from 'react';
```

אז, כדי להפעיל פונקציות כאשר מערכת הרשמה של התוסף משתנה,

פשוט נכניס את הפונקציות הרצויות לתוך ה- useEffect כפונקציות מקבלת פרמטר:

```
useEffect(() => { // פונקציות להפעלה כאשר מערכת ההרשמה משתנה });
```

אנחנו יכולים גם לציין מהי מערכת ההרשמה שאליה יש להתרשם, על-פי הצורך.

לדוגמה, אם אתה רוצה להפעיל פונקציות רק כאשר ערך מסויים משתנה, אתה יכול לעשות זאת על-פי הדוגמה הבאה

```
useEffect(() => { // פונקציות להפעלה רק כאשר ערך מסויים משתנה }, [value]); ``
```

Cossci

ספריות עיצוב MUI

Material-UI הינה חבילת עיצוב מרכזית עבור **React** שמאפשרת להכניס את העיצוב של **Material Design** לאפליקציות שלנו.
היא מספקת מגוון גדול של קומפוננטות מוכנות לשימוש, כגון כפתורים, תיבות, טבלאות ועוד.
כך אנו יכולים להקל על הפיתוח ולהתמקד בתוכניות העיצוב שלך.

כדי להתחיל להשתמש ב- **Material-UI**, ראשית עליך לייבא את החבילה לתוך האפליקציה שלך עם הפקודה הבאה:

```
npm install @material-ui/core
```

אחרי שהחבילה התקבלה, נוכל להתחיל להשתמש בפקודות העיצוב של **Material-UI** בקוד ה- **React** שלנו.
לדוגמה, כדי להכניס כפתור מעוצב, נוכל לכתוב את הקוד הבא:

```
import Button from '@material-ui/core/Button';
function MyApp() {
return ( <Button variant="contained" color="primary">
 כפתור מעוצב
</Button> ); }
```

אנחנו יכולים גם להתאים את העיצוב של הפקודות לפי הצורך על-פי השימוש בתכונות הנוספות שמופיעות בתיעוד הפקודה. לדוגמה, כדי לשנות את צבע הכפתור,

```
  <Button variant="contained" color="secondary">
     // כפתור מעוצב.
    </Button>
```

Cossci

איך ליצור פעולות מחזיקות (hooks) משלנו ב- React

פעולות מחזיקות (hooks) הן פונקציות מיוחדות שמאפשרות לנו להעביר תכונות ופונקציות מיוחדים של React לתוך קומפוננטות רגילות.
כדי ליצור פעולת מחזיקה, עלינו לכתוב פונקציות שמקבלות את מערכת ההרשמה של התוסף כפרמטר ומחזירות את מערכת הרהשמה המשתנה.
כמו כן, עלינו לכתוב פונקציות עזר שמטפלות בהפעלת הפעולות ובשינוי מערכת ההרשמה המשתנה.
לדוגמה, כדי ליצור פעולת מחזיקה המטפלת במונה, ניתן לכתוב את הקוד הבא:

```
import { useState } from 'react';
//  אנחנו ממציאים את שם הפעולה המחזיקה.
 function useCounter(initialCount) {
   const [count, setCount] = useState(initialCount);
 function increment() {
   setCount(prevCount => prevCount + 1);
  }
 function decrement() {
   setCount(prevCount => prevCount - 1);
  }
   return [count, increment, decrement];
 }
```

כעת, ניתן להשתמש בפעולה המחזיקה
אחרת בתוך פונקציות React כדי לנהל את ספירת המשתנה:

```
import { useCounter } from ‘./useCounter';
 //      קומפוננטה המייצגת מונה
 function Counter() {
     const [count, increment, decrement] = useCounter(0);
     // counter UI
      return ( <div>
 <button onClick={decrement}>-</button>
    {count}
 <button onClick={increment}>+</button>
 </div> );
}
```

Cossci

איך להשתמש ב'מקרה' ב- React (context)

(context) הוא מקום מיוחד בתוך ריאקט שמאפשר לנו להעביר נתונים מתוך המחזיקה(Container) . העליונה בעץ הקומפוננטות לתוך כל מחזיקה מתחתיה ללא צורך בעדכון המחזיקה העליונה בכל פעם שמתקבלת עדכון.

כדי להשתמש בContext, עלינו ליצור context חדש בתוך המחזיקה העליונה בעץ הקומפוננטות

ולהעביר את הcontext לתוך המחזיקות המתחתיות באמצעות הפקודה React.createContext :

```
import React from 'react';
 const MyContext = React.createContext();
```

```
 function MyProvider({ children }) {
    const [state, setState] = useState({ message: 'Hello World' });
  return ( <MyContext.Provider value={value}>
     {children}
 </MyContext.Provider> );
 }
```

עכשיו, ניתן להשתמש במקרה בתוך המחזיקות המתחתיות באמצעות הפקודה useContext:

```
import { useContext } from 'react.'
 function Child() {
   const context = useContext(MyContext);
   return <p>{context.message}</p>;
 }
```

Cossci

אימות Firebase ב React

ראשית, עלינו ליצור חשבון Firebase ולהפעיל את שירות האימות בפאנל הניהול של Firebase.
כעת, עלינו להתקין את הספריות הנחוצות לתוכנת ה־React שלנו באמצעות הפקודה הבאה:

```
npm install firebase
```

בנוסף, נתקין גם את ספריית ה־React wrapper של Firebase באמצעות הפקודה הבאה:

```
npm install react-firebase-hooks
```

כעת, ניתן לממש את תהליך האימות בתוך התכנית שלנו.
ראשית, עליכם לייבא את הספריות הנחוצות לתוכנה שלכם באמצעות הקוד הבא:

```
import firebase from 'firebase';
import { useAuthState } from 'react-firebase-hooks/auth';
```

עכשיו, נעשה שימוש בפונקציות האימות של Firebase כדי לטעון את המשתמש הנוכחי מהמערכת.

זה ניתן לעשות באמצעות הקוד הבא:

```
const [user, loading, error] = useAuthState(firebase.auth());
```

דוגמאות לשימוש בשקופית הבאה

Cossci

דוגמאות לשימוש באימות באמצעות Firebase

כפתור להתחברות אם המשתמש מנותק:

```
{user ? ( <p>Welcome, {user.email}</p> )
: ( <button onClick={() => firebase.auth()
     .signInWithPopup(new firebase.auth.GoogleAuthProvider())}>
 Sign in with Google
</button> )}
```

ניתן גם לממש פעולות נוספות, כמו להתנתק מהמערכת או ליצור חשבון חדש. לדוגמה, ניתן לכתוב פונקציית התנתקות באמצעות הקוד הבא:

```
const signOut = () => { firebase.auth().signOut(); }
```

וניתן להשתמש בה כדי להתנתק מהמערכת באמצעות כפתור:

```
<button onClick={signOut}>Sign out</button>
```

אם נרצה ליצור חשבון חדש, ניתן לעשות זאת באמצעות הקוד הבא:

```
firebase.auth().createUserWithEmailAndPassword(email, password)
.then(() => { console.log('Successfully created new user'); })
.catch(error => { console.error(error); });
```

אם יש לכם שאלות נוספות או תנאים, אנא אל תהססו לשאול.