

A JavaScript Library for building user interfaces

Created By David Yakin 2023

תוכן עניינים

7	Introduction	>
10	Virtual DOM	>
13	Create-react-app	>
22	React Server	>
25	React Developer Tools	>
28	<u>Getting Started</u>	>
36	<u>Bable.js</u>	>
40	<u>Component</u>	>
45	<u>Template</u>	>
49	<u>Compilation Error</u>	>
53	<u>Logic</u>	>
55	<u>String interpolation</u>	>
57	<u>Styles</u>	>
59	<u>Inline style</u>	>
61	<u>Styles from module</u>	>
63	<u>External libraries</u>	>
65	<u>Props</u>	>
66	<u>Passing string</u>	>
70	<u>Passing Object</u>	>
74	<u>Sending two keys</u>	>

80	Loops	➤
84	Conditional Rendering	➤
86	Events	➤
87	JAVASCRIPT EVENTS	➤
89	Function invocation with parameters	➤
91	Catching Event	➤
94	Raising Events	➤
98	PropTypes	➤
102	PropTypes Errors	➤
105	Main Types	➤
107	Array & Object of Types	➤
109	oneOfType vs oneOf	➤
112	Exact &isRequired	➤
116	Shape Any & defaultProps	➤
119	node & children	➤
128	Shared Components	➤
129	PageHeader.jsx	➤
133	Static Folder	➤
137	React Hooks	➤
140	useState	➤

152	<u>Layout</u> >
160	<u>Error Page</u> >
163	<u>React Router Dom</u> >
169	<u>Routes</u> >
171	<u>BrowserRouter</u> >
174	<u>Link & NavLink</u> >
181	<u>useNavigate</u> >
183	<u>Navigate</u> >
187	<u>useParams</u> >
193	<u>Nested Routes</u> >
198	<u>Life Cycle Hooks</u> >
201	<u>Initial rendering</u> >
204	<u>useEffect</u> >
213	<u>Custom hooks</u> >
219	<u>Memoization</u> >
221	<u>useCallback</u> >
228	<u>useMemo</u> >
233	<u>axios</u> >
238	<u>card ApiService</u> >
244	<u>CardsFeedback.jsx</u> >
253	<u>useCards</u> >
261	<u>axios interceptors</u> >

267	Context	›
270	example	›
281	Snackbar	›
287	Forms	›
291	Input.jsx	›
294	FormButton.jsx	›
297	Form.jsx	›
301	useForm.js	›
309	Form implementation	›
314	Login	›
316	jwt-decode	›
318	localStorageService.js	›
320	UserProvider.jsx	›
323	usersApiService.js	›
325	useUsers.js	›
329	Joi-schema	›
331	Initial Form Data	›
333	Axios interceptor	›
335	LoginPage.jsx	›
339	Get Current User	›

347	Logout	>
348	handleLogout	>
350	MemuLink.jsx	>
352	Menu.jsx	>
357	MenuProvider.jsx	>
364	Signup	>
366	initialSignupForm.js	>
368	Normalize User	>
370	Joi-schema	>
371	usersApiService.js	>
374	useUsers.js	>
376	SignupPage.jsx	>
381	MyCardsPage.jsx	>
385	Delete Card	>
398	Edit Card	>
400	mapToModel	>
406	EditCardPage.jsx	>
411	Like Card	>
419	FavCardsPage.jsx	>
423	Search bar	>
424	useSearchParams	>
428	SearchBar implementation	>

React Introduction

https://www.youtube.com/watch?v=XxVg_s8xAms



Definition

React is a free and open-source front-end JavaScript library for building user interfaces based on UI components.

It is maintained by Meta (formerly Facebook) and a community of individual developers and companies.

React is only concerned with state management and rendering that state to the DOM

creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

👍 Benefits

Virtual DOM

User Experience

State Handle

Components

No Explicit Data Binging

Life cycle hooks

Open source

Virtual DOM

החדשון והקונספט המרכזי אותו מביא
ריאקט הוא זה - **Virtual DOM**

React עוקב אחר השינויים בדום וירטואלי
ותעדכן את הדום האמיתי רק במקומות
שבהם התרחשו השינויים.

שיטה זאת יוצרת חיסכון אדיר במשאבים
ומהירות תגובה גבוהה לכל שינוי.

<https://reactjs.org/docs/faq-internals.html>



Virtual DOM implementations



State



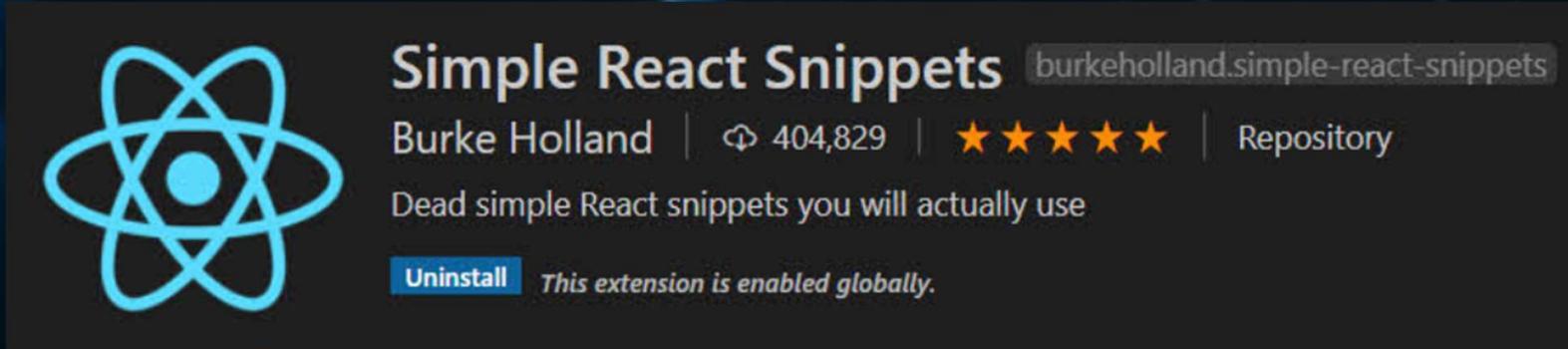
Life cycle
hooks



Components

Simple React Snippets

תוסף שיעזר לנו בעבודה עם React בסביבת העבודה של vscode



Create-react-app

כלי שיעזר לנו לפתח פרויקט חדש ב - React





Installation

```
npm i -g create-react-app
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

To address all issues (including breaking changes), run:
npm audit fix --force

Run `npm audit` for details.

Created git commit.

Success! Created client at C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client
Inside that directory, you can run several commands:

npm start
Starts the development server.

npm run build
Bundles the app into static files for production.

npm test
Starts the test runner.

npm run eject
Removes this tool and copies build dependencies, configuration files
and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

cd client
npm start

Happy hacking!

C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app>[]

New React Project

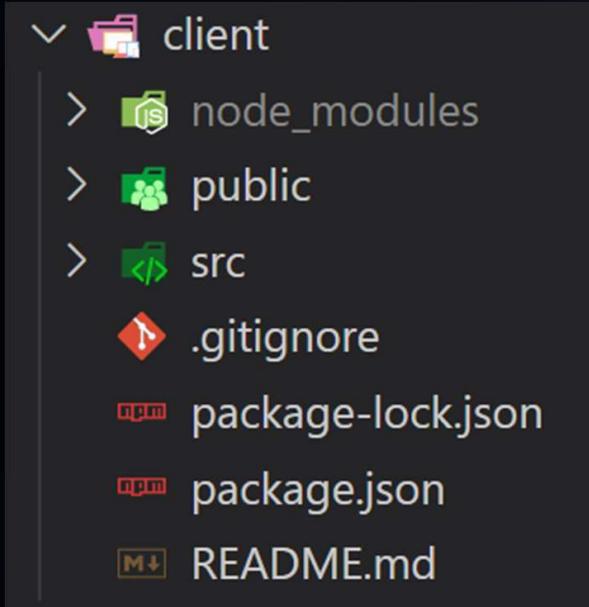
לאחר שהורדנו והתקנו את התוסף create-react-app נוכל להשתמש בו כדי לפתח פרויקט חדש ב - React

- ניכנס לתוך המ תיקיה בה אנו מעוניינים ליצור פרויקט חדש של React

• נקייש בטרמינל את הפקודה create-react-app client

- 수행ולת ה - CLI תסתיים ונראה את הכיתוב "Happy hacking"

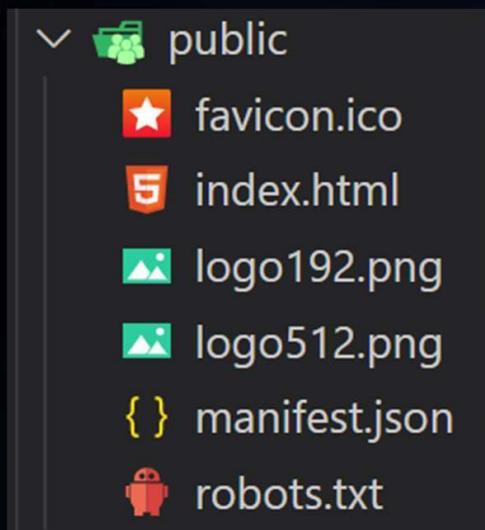
תשתיית הקבצים שה – CLI יצר



- **Node_modules** – תיקייה השומרת בתוכה את הספריות שהורدنנו לפרויקט
- **public** – תיקייה השומרת את הקבצים הסטטיסיים
- **src** – תיקייה בה רוב הקוד של הפרויקט יכתב
- **.gitignore** – קובץ קונפיגורציות של git ובתוכו ההוראות מאיילו תייקיות וקבצים להעתלם ולא להעלות ל- github
- **package.json** – קובץ קונפיגורציות הכלול בתוכו פקודות, רשימת תלויות בספריות ועוד
- **package-lock.json** – קובץ השומר את גרסאות הספריות
- **README.md** – קובץ בו ניתן לכתוב פרטים על הפרויקט

Public

- **favicon.ico** – קובץ התמונה שריינט משמשת בו באופן דיפולטיבי בפרויקט חדש
- **index.html** – קובץ ה – HTML המרכזי של הפרויקט
- **logo192.png** – הלוגו של ריאקט קטן
- **logo512.png** – הלוגו של ריאקט בגודל יותר
- **manifest.json** – קובץ קונפיגורציות לאפליקציה של מובייל או דסктופ
- **robots.txt** – קובץ העוזר למנוע החיפוש google בסריקת האפליקציה



! [לינק לסרטון המסביר על קובץ robots.txt](https://www.youtube.com/watch?v=fzm-zYHjlgY)

index.html

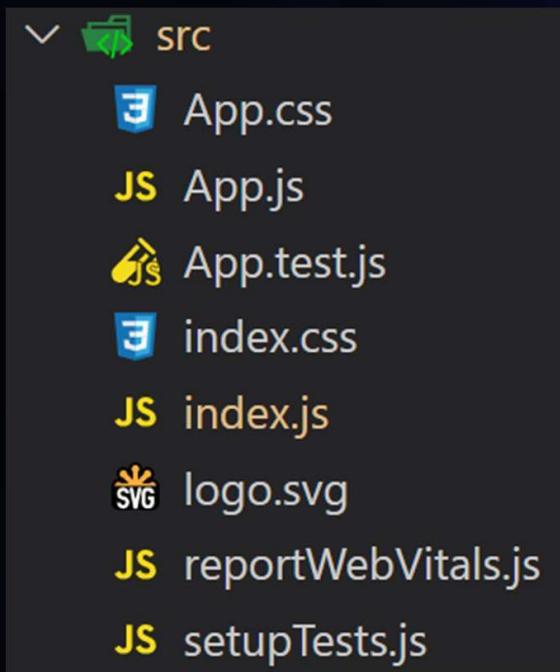
- קישור למיקום תמונה ה – favicon
- קישור למיקום התמונה ל – apple
- קישור מיקום קובץ json – manifest.json
- כותרת האפליקציה
- האלמנט אליו יזרקו כל שאר הkomponentot

```
5 index.html M X
client > public > 5 index.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3     <head>
4       <meta charset="utf-8" />
5       <link rel="icon" href="%PUBLIC_URL%/favicon.ico" /> ←
6       <meta name="viewport" content="width=device-width, initial-scale=1" />
7       <meta name="theme-color" content="#000000" />
8       <meta
9         name="description"
10        content="Web site created using create-react-app"
11      />
12      <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13      <link rel="manifest" href="%PUBLIC_URL%/manifest.json" /> ←
14      <title>React App</title> ←
15    </head>
16    <body>
17      <noscript>You need to enable JavaScript to run this app.</noscript>
18      <div id="root"></div> ←
19    </body>
20  </html>
```

SRC

- **App.css** – קובץ העיצוב של הקומponent `app.js`
- **App.js** – קובץ הלוגיקה של הקומponent `app`
- **App.test.js** – קובץ הבדיקות של הקומponent `app`
- **index.css** – קובץ העיצוב של הקומponent `index`
- **index.js** – קובץ הלוגיקה של קומponent `index`
- **logo.svg** – קובץ הלוגו של ריאקט
- **reportWebVitals.js** – בדיקת אופטימיזציה של האפליקציה
- **setupTests.js** – קובץ הבדיקות המרכזי של האפליקציה

! לינק לסרטון שמסביר על webVitals
<https://www.youtube.com/watch?v=00RoZfIYE34>



JS App.js X

client > src > JS App.js > ...

```
1 import logo from './logo.svg'; ←
2 import './App.css'; ←
3
4 function App() { ←
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           | Edit <code>src/App.js</code> and save to reload.
11         </p>
12         <a
13           |   className="App-link"
14           |   href="https://reactjs.org"
15           |   target="_blank"
16           |   rel="noopener noreferrer"
17           | >
18             |   Learn React
19           | </a>
20         </header>
21       </div>
22     );
23   }
24
25 export default App;
```

App.js

- "בוא קובץ ה – סו לkomponent
- "בוא קובץ העציב לkomponent
- ייצרת komponent App
- התצוגה לגולש (מה שהפונקציה מחזירה)

index.js

קובץ הלוגיקה המרכזי של האפליקציה

- **"יבוא מודולים":**

- מופע מהספרייה React לקומפוננט
- קובץ העיצוב של הקומפוננט App
- קובץ בדיקות האופטימיזציה לקומפוננט

- **יצירת קבוע בשם root שערך שווה**

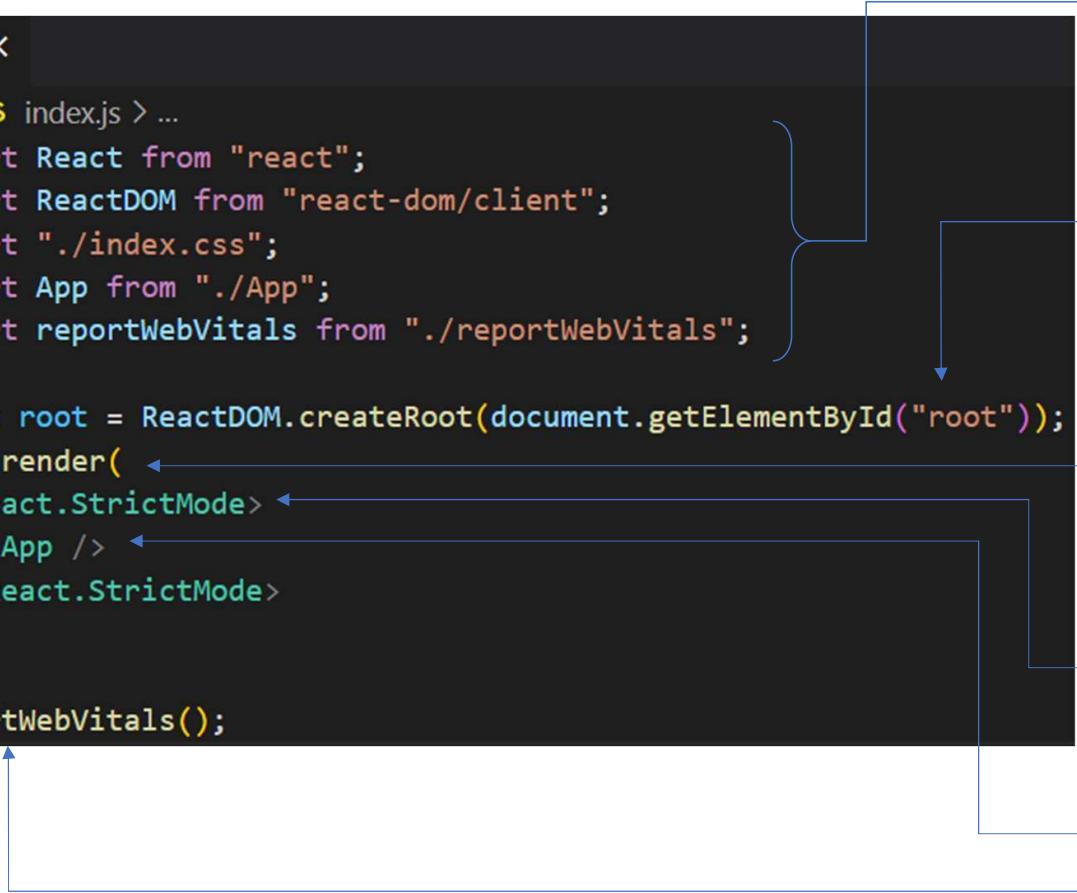
ערך להפעלת מטודת ReactDOM.createRoot שתתפס את האלמנט זהה נמצא בתוך הקובץ index.html (בתיקיה index.html) (public)

- הפעלת מטודת root.render אשר תזריק לתוך האלמנט שתפסנו root את הקומפוננטות (ידובר בהרחבת המשך)

עטיפת האפליקציה בקומפוננט של React שייכפה strictMode על הקוד שייכתב בתוכו (ידובר על כך בהרחבת המשך המציגת)

- הצבת קומפוננט App כרך שתוצג לגולש reportWebVitals()
- הפעלת מטודת reportWebVitals()

```
JS index.js M X
client > src > JS index.js > ...
1 import React from "react";
2 import ReactDOM from "react-dom/client";
3 import "./index.css";
4 import App from "./App";
5 import reportWebVitals from "./reportWebVitals";
6
7 const root = ReactDOM.createRoot(document.getElementById("root"));
8 root.render(
9   <React.StrictMode>
10    <App />
11   </React.StrictMode>
12 );
13
14 reportWebVitals();
```





React Server

בעזרת הפקודה בטרמינל `npm start` נפעיל את השרת הזמן של ריאקט וונכל לראות את התצוגה הראשונית של האפליקציה



npm start

נפעיל את השרת הזמני של React
שגם יאפשר לשימושים בקוד ויתן לנו
תצוגת אמת של הקוד שלנו

- נכנס לתיקייה הRELATIONAL בנה נמצא פרויקט ה – React

• נקליד את הפקודה npm start

- ה – CLI יודיע לנו ובמידה ולא תהיה שגיאה בתהיליך compile נראה את הכתיב הבא

- ה – CLI יעדכן אותנו כי אנו מודים לפורט הדיפולטיבי של React שהוא 3000

- ה – CLI יעדכן אותנו כי אנחנו בסביבת עבודה של development

! כמו כן ה – CLI יפתח את הדף בכתובת
ובפורט הרשמי

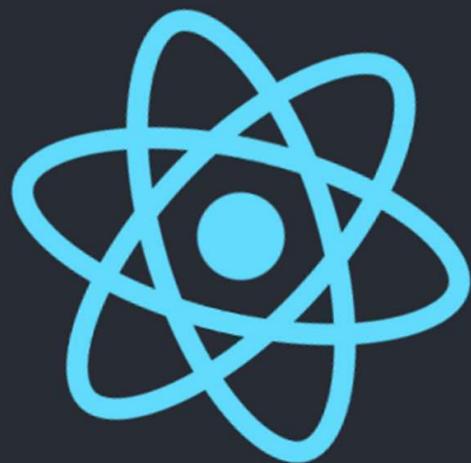
```
Compiled successfully!
```

```
You can now view client in the browser.
```

```
Local:          http://localhost:3000
On Your Network:  http://10.0.0.8:3000
```

```
Note that the development build is not optimized.
To create a production build, use npm run build.
```

```
webpack compiled successfully
```



Edit `src/App.js` and save to reload.

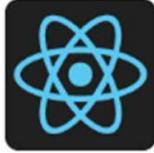
[Learn React](#)

התוצאה בדף

אם לא התקבלה שגיאה והכל תקין אנו
אפשרים לראות בדף את התצוגה
הבא

React Developer Tools

כלי שיעזר לנו בשלבי הפיתוח ב - React



React Developer Tools

Featured

★★★★★ 1,402 | [Developer Tools](#) | 3,000,000+ users

<https://reactjs.org/blog/2015/09/02/react-developer-tools-for-react-0.13.html>



Components

לשונית זאת מראה לנו נתונים נוספים על קומponent נבחרת

- בחלק זה של הדף נראה איפה עומדת הקומponent בהיררכיה של האפליקציה

- החלק השני מדבר על הקומponent עצמה ובו ניתן לראות נתונים כמו:

- props – נתונים שהועברו לקומponent
- state – משתנים ש React תגיב לשינויים בערכיהם
- rendered by – הฟונקציה שאחראית על טעינה וטעינה מחדש ומוחדש של הקומponent
- source – קובץ המקור

! בغالל שאנו לא מנהלים state בקומponent זהה אני לא רואה את הנתונים הללו

The screenshot shows the React DevTools Components tab. At the top, there's a search bar and some status indicators. Below it, the component tree starts with an 'App' component, which is itself rendered by the 'createRoot()' function from 'react-dom@18.2.0'. The 'props' section shows a single prop named 'new_entry' with a value of an empty string. The 'source' section indicates the code is located at 'src/index.js:10'. On the left side of the slide, there's a large blue React logo and some UI text: 'Edit src/App.js and save to reload.' and 'Learn React'.

Profiler

The screenshot shows the React DevTools Profiler interface across three stages:

- Initial State:** Shows the React logo and a message: "Edit `src/App.js` and save to reload." Below it is a "Learn React" link.
- Recording:** Shows the same UI with a red record button instead of a blue one. It displays the message "Profiling is in progress..." and "Click the record button ⚡ to stop recording."
- Detailed Breakdown:** Shows the recorded data. The left pane lists components: BrowserRouter, Router, Navigation.Provider, Location.Provider, and App. The right pane shows a tree of components under "App": Header, Navbar, Navbar (ForwardRef), NavbarContext.Provider, Context.Provider, Container (ForwardRef), NavbarCollapse (ForwardRef), Anonymous (ForwardRef), Anonymous (ForwardRef), Transition, Conte..., and Nav (F...). A tooltip over the "Main" node shows its rendering times: "Rendered at: 3.2s for 14.2ms, 3.9s for 10.6ms, 4.6s for 13.2ms, 5s for 32.1ms, 7.2s for 2.5ms, 8.8s for 11ms".

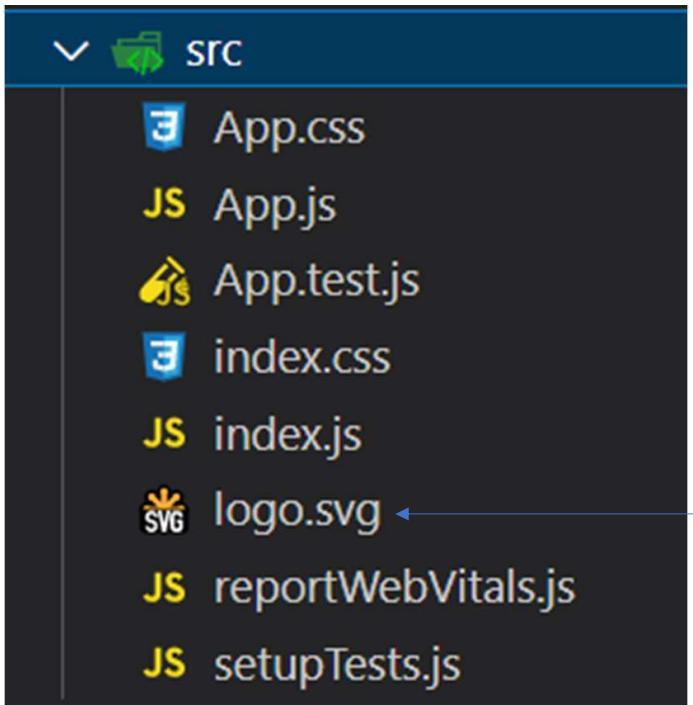
בלשונית זאת נוכל לעשות בדיקות אופטימיזציה

- לחיצה על כפתור `record` תתחיל להאזין לארועים שוקרים ב- DOM
- לחיצה נוספת תעצור את ההקלטה ובמידה ויהיו נתונים להציג (כמו משך הזמן שלקח לכל אירוע לפחות) החלקים הללו יוצגו לנו.
- לחיצה על אחד מהאירועים תיתן לנו **פרטים עליון**

תחילה עבודה

ניקוי ראשוני של האפליקציה מתמונות וערכים דיפולטיביים של
create-react-app





src

- מחיקת קובץ הלוגו של ריאקט

App.js

- מחייקת יבוא קובץ הלוגו של ריאקט
- מחייקת תוכן האלמנט div עם המחלקה העיצובית App
- התוצאה

```
1 import logo from './logo.svg'; ←
2 import './App.css';
3
4 function App() {
5   return (
6     <div className="App">
7       <header className="App-header">
8         <img src={logo} className="App-logo" alt="logo" />
9         <p>
10           | Edit <code>src/App.js</code> and save to reload.
11         </p>
12         <a
13           className="App-link"
14           href="https://reactjs.org"
15           target="_blank"
16           rel="noopener noreferrer"
17         >
18           | Learn React
19         </a>
20       </header>
21     </div>
22   );
23 }
24
25 export default App;
```

```
1 import "./App.css";
2
3 function App() {
4   return <div className="App"></div>; ←
5 }
6
7 export default App;
```

index.css

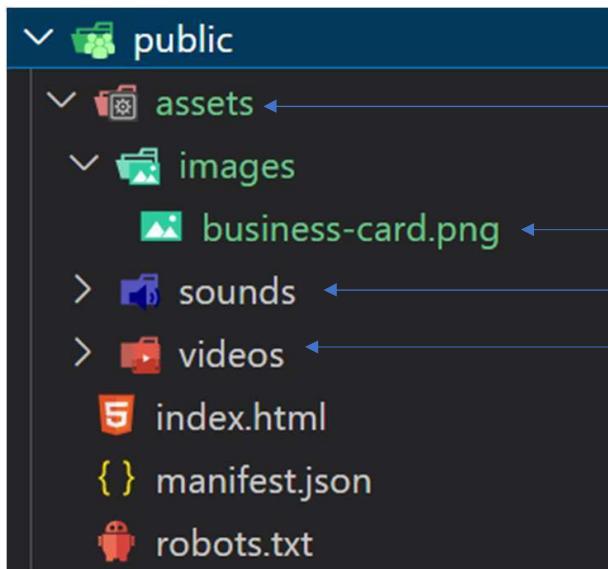
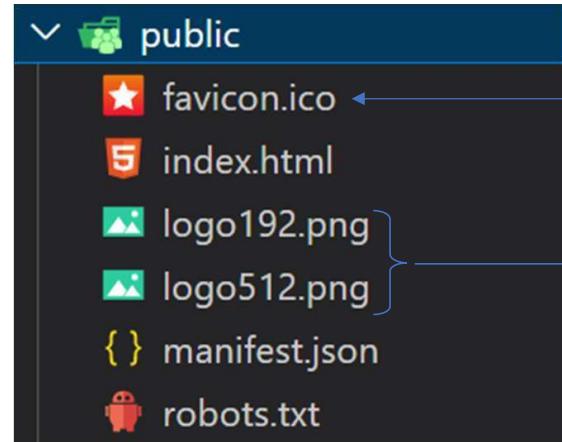
```
index.css X
src > index.css > body
1 body {
2   margin: 0;
3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
4   |   'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
5   |   sans-serif;
6   -webkit-font-smoothing: antialiased;
7   -moz-osx-font-smoothing: grayscale;
8 }
9
10 code {
11   font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
12   |   monospace;
13 }
```

```
index.css M X
src > index.css > ...
1 * {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5 }
6
7 center {
8   display: flex;
9   justify-content: center;
10  align-items: center;
11 }
12
13 cursor {
14   cursor: pointer;
15 }
```

- מחיקת תוכן הקובץ
- יצירת של מחלקות עיצוביות משלנו
- מחיקת התוכן של הקובץ App.css

App.css M X

```
src > App.css
1 |
```



public

- מחיקת קובץ favicon.ico של ריאקט react
- מחיקת הלוגואים של react
- הוספה תיקייה בשם assets ובהוכה שלוש תיקיות:
 - Images •
 - נוריד מאתר pixabay איקון מתאים לאפליקציה שלנו
 - sounds •
 - videos •

index.html

- החלפת ה icon לתמונת ה – icon שיבאנו לפרויקט
- החלפת ה – icon למקורה ומשתמשים ב – apple
- שינוי הכתוב באלמנט ה – title לשם האפליקציה

```
index.html M X  
public > index.html > html > head > link  
1   <!DOCTYPE html>  
2   <html lang="en">  
3     <head>  
4       <meta charset="utf-8" />  
5       <link rel="icon" href="%PUBLIC_URL%/assets/images/business-card.png" />  
6       <meta name="viewport" content="width=device-width, initial-scale=1" />  
7       <meta name="theme-color" content="#000000" />  
8       <meta  
9         name="description"  
10        content="Web site created using create-react-app"  
11      />  
12      <link  
13        rel="apple-touch-icon"  
14        href="%PUBLIC_URL%/assets/images/business-card.png" />  
15      </link>  
16      <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />  
17      <title>Business Cards App</title> ←  
18    </head>  
19    <body>  
20      <div id="root"></div>  
21    </body>  
22  </html>
```

```
{ } manifest.json M X  
public > { } manifest.json > ...  
1  {  
2    "short_name": "Business cards app", ←  
3    "name": "Business card application for business and clients",  
4    "icons": [  
5      {  
6        "src": "./assets/images/business-card.png", ←  
7        "sizes": "64x64 32x32 24x24 16x16",  
8        "type": "image/x-icon"  
9      },  
10     {  
11       "src": "./assets/images/business-card.png", ←  
12       "type": "image/png",  
13       "sizes": "192x192"  
14     },  
15     {  
16       "src": "./assets/images/business-card.png", ←  
17       "type": "image/png",  
18       "sizes": "512x512"  
19     }  
20   ],  
21   "start_url": ".",  
22   "display": "standalone",  
23   "theme_color": "#000000",  
24   "background_color": "#ffffff"  
25 }
```

manifest.json

- שינוי השם המופיע של האפליקציה
- קביעת השם המלא של האפליקציה
- שינוי מיקום התמונות בשביל ה - icons

משימת app



Business-cards-app

- הורד את ה – CLI של create-react-app בapoן גלובלי
- פתח פרויקט חדש בעזרת create-react-app בשם client
- הכן את הפרויקט לעבודה על ידי ניקוי הקבצים והתיקיות ללא רלוונטיות לפרויקט
- הוסף קבצים ותיקיות שיידרשו לפרויקט כמו שמופיע בשקפים הקודמים

Bable.js

A JavaScript compiler

<https://babeljs.io>





Definition

Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments



👍 Benefits

Source code transformations

Shows compilation errors clearly

Compatibility with all types of browsers

Allows writing declarative code

Polyfill features that are missing in your target environment through a third-party polyfill

Babel sandbox

דוגמה לתחילת המרת הקוד באמצעות babel.js ניתן לראות באתר שלהם בתפריט הניוט של <https://babeljs.io/> Try it out

- ארגז המשחקים זהה בנויה משלושה חלקים:

- מסך ימני – מציג את הקוד לאחר תחילת הקומpileציה
- מסך אמצעי – משמש לנכיתה קוד javascript דקלרטיבי ועדכני
- תפריט צידי – בו אפשרויות שונות לתצוגת הקוד לאחר קומPILEציה

The screenshot shows the Babel sandbox interface. On the left, there's a sidebar with 'SETTINGS' (Evaluate, Line Wrap checked, Prettify, File Size, Time Travel), 'Source Type' (Module selected), and 'TARGETS' (defaults, not ie 11, not ie_mob 11). The main area has two code panes. The left pane contains the original code:1 "use strict";
2 hallo
3 </> The right pane contains the transformed code:

```
1 /*#__PURE__*/React.createElement(React.Fragment,
  null, "hallo");
```

https://www.youtube.com/watch?v=UeVq_U5obnE&t=149s

לינק להרצאה המסביר איך babel.js עובדת מאחור הקלעים !

Component

יחידת קוד עצמאית ואחת מארגוני היסודות של ספריית React





Definition

Components let you split the UI into independent, reusable pieces, and think about each piece in isolation

Components structure



TEMPLATE
HTML

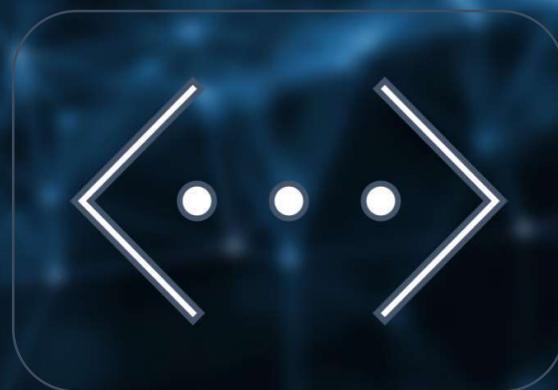


LOGIC
JAVASCRIPT



STYLES
CSS

Components Types



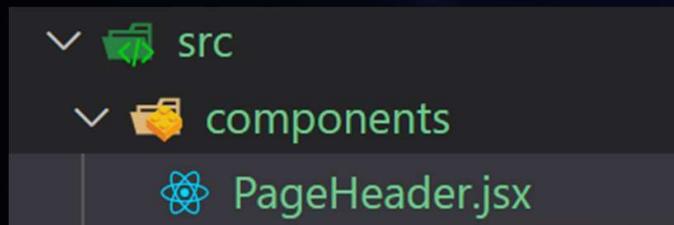
Function



Class

הכנת תשתיות

- בתוך תיקייה `src` ניצור תיקייה חדשה בשם **components**
- בתחום ניצור קובץ בשם **PageHeader.jsx**



! קומפוננט תמיד מתחיל באות גדולה
! סימנת `jsx` / `tsx` אילו הסימנות של `table`



Template

ניצור קומפוננט מסווג פונקציה שתחזיר
אלמנט של HTML אותו ניתן נציג לגולש



PageHeader

PageHeader.jsx •

- ניצור קבוע בשם `PageHeader` שערך יהיה שווה לפונקציה אונימית

- הפונקציה תחזיר אלמנט של `HTML` מסוג `H1` עם הכיתוב בתוכו

- לבסוף ניצאת הפונקציה מהמודול באמצעות `export default`

App.js •

- ניבא את הקומפוננט שיצרנו
- נציב אותה בתוך החלק של ה `HTML` אותו הפונקציה של הקומפוננט `App` מחרירה.

! במצגת זאת נתמקד בקומפוננטות מסוג **functional components** של `React` !
חויה לעטוף את כל האלמנטים שהפונקציה מחרירה באלמנט או של `HTML` או של `React`

PageHeader.jsx

```
src > components > PageHeader.jsx > ...
1  const PageHeader = () => { ←
2    |  return <h2>pageHeader works!</h2>; ←
3  };
4
5  export default PageHeader; ←
```

App.js

```
src > App.js > ...
1  import "./App.css";
2  import PageHeader from "./components/PageHeader";
3
4  function App() {
5    return (
6      <div className="App">
7        <PageHeader /> ←
8      </div>
9    );
10 }
11
12 export default App;
```

התוצאה בדף

- ניתן לראות שהטקסט שהחזינו מהקומponent שיצרנו Pageheader מוצג לגולש עם העיצוב של אלמנט ה – H2 שנתנו לו

pageHeader works!



Compilation Error

במידה ותהיה שגיאה בקוד Babel תתריע
לי על כר במספר מקומות



איתור שגיאות

The screenshot shows a code editor interface. At the top, there is a file tree with the following structure:

- src
- components
- PageHeader.jsx
- App.css
- App.js

Below the file tree, the code editor displays the content of the `PageHeader.jsx` file:

```
src > components > PageHeader.jsx > PageHeader
1 const PageHeader = () => {
2   return (
3     <h2>pageHeader works!</h2>
4     <p>hallo world</p>
5   );
6 }
7
8 export default PageHeader;
```

Red arrows point from the following elements in the sidebar to specific parts of the code:

- A red arrow points from the "components" folder icon to the `components` folder in the file tree.
- A red arrow points from the "PageHeader.jsx" file icon to the `PageHeader.jsx` file in the file tree.
- A red arrow points from the "PageHeader.jsx" file icon to the first line of the code (`const PageHeader = () => {`).
- A red arrow points from the "PageHeader.jsx" file icon to the `<h2>pageHeader works!</h2>` line in the code.

- עכ התיקייה והקובץ ייצבע באדום
- לצד הקובץ בו נעשתה השגיאה יופיע מס' השגיאות בדף
- הלשונית של המודול תיצבע אדום
- מתחאת לקטעי הקוד שדורשים תיקון יופיע קוו אדום משונן

בטרמינל של vscode

- בלשונית TERMINAL

- תופיע השגיאה
- פירוט השגיאה
- באיזה נתיב היא נמצאת

- בלשונית PROBLEMS

- יופיע באופן מוקוצר מיקום השגיאה
- מהות השגיאה
- סוג השגיאה

The screenshot shows the VS Code interface with the following components visible:

- Terminal Tab:** Shows the command "Local: http://localhost:3000". Below it, an error message "Failed to compile." is shown, followed by a stack trace of a SyntaxError from a file named "PageHeader.jsx". The stack trace indicates that JSX elements must be wrapped in an enclosing tag.
- PROBLEMS Tab:** Shows two errors for the file "PageHeader.jsx":
 - An ESLint error: "JSX expressions must have one parent element. ts(2657) [Ln 3, Col 3]".
 - A ESLint error: "Parsing error: Adjacent JSX elements must be wrapped in an en... eslint [Ln 4, Col 2]".
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, JUPYTER, and Jupyter notebook icons. It also features a filter bar and a status bar with a count of 2.

בדפס

- בקונסול תופיע השגיאה
 - במסך התצוגה הראשי יופיעו פרטי השגיאה

במסך התצוגה הראשי ניתן ללחוץ על הסימן X ולוחזר לתצוגת האפליקציה אף מומלץ לתקן את השגיאה בקדם במקום

Compiled with problems: X

ERROR in ./src/components/PageHeader.jsx

Module build failed (from ./node_modules/babel-loader/lib/index.js):
SyntaxError: C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\src\components\PageHeader.jsx: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>? (4:2)

```
2 |   return (
3 |     <h2>pageHeader works!</h2>
> 4 |     <p>hallo world</p>
|     ^
5 |   );
6 |
7 | };
```

at instantiate (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules@babel\parser\lib\index.js:67:32)
at constructor (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules@babel\parser\lib\index.js:364:12)
at FlowParserMixin.raise (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules@babel\parser\lib\index.js:7210:18)
at FlowParserMixin.jsxParseElementAt (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules@babel\parser\lib\index.js:7220:17)
at FlowParserMixin.parseElement (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules@babel\parser\lib\index.js:7233:19)
at FlowParserMixin.parseExprAtom (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules@babel\parser\lib\index.js:11171:23)

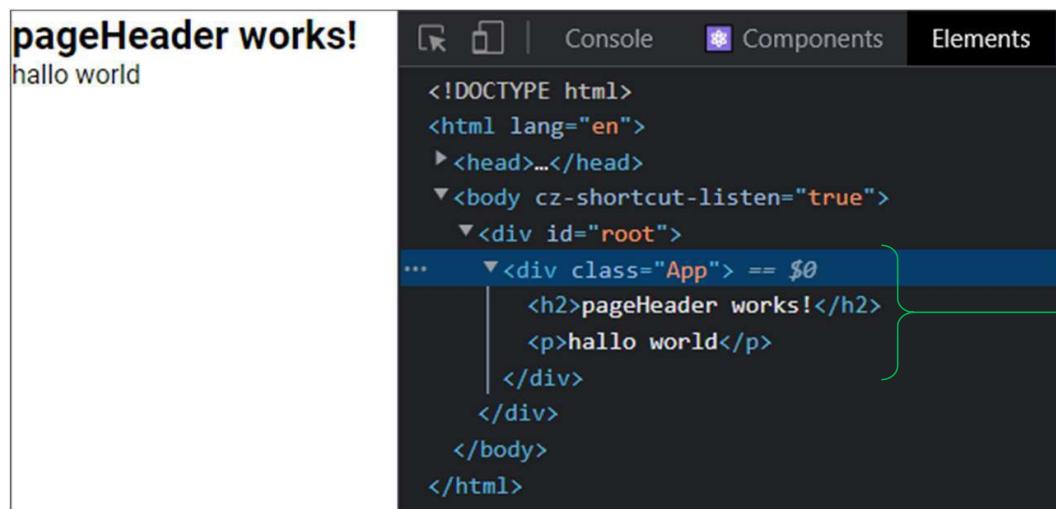
2 Issues: 2

✖ Uncought Error: Module build failed (from ./node_modules/babel-loader/lib/index.js):
SyntaxError: C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\src\components\PageHeader.jsx: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>? (4:2) (at App.js:12:12)

App.js:12

תיקון השגיאה

```
PageHeader.jsx
src > components > PageHeader.jsx > PageHeader
1  const PageHeader = () => {
2    return (
3      <> <
4        |<h2>pageHeader works!</h2>
5        |<p>hallo world</p>
6      </>
7    );
8  };
9
10 export default PageHeader;
```



במקרה זהה מקור השגיאה היה
שניסיתי להחזיר לעללה אלמנט
HTML אחד מהкомпонент

- אם אני לא מעוניין לעוטף את שני
האלמנטים באלמנט עיצובי של
HTML כמו div ריאקט pseudo element
משלה שנקרא element

React.Fragment שבעזרתו אוכל
לעוטף את האלמנטים אולם האלמנט
זה לא יראה ב – DOM

- כפי ב – dev tools של דפדפן chrome
בלשונית Elements לא
נוסף לנו אלמנט עיצובי של HTML

! הדרכ המקוצרת של כתיבת האלמנט
React.Fragment היא <></>



Logic

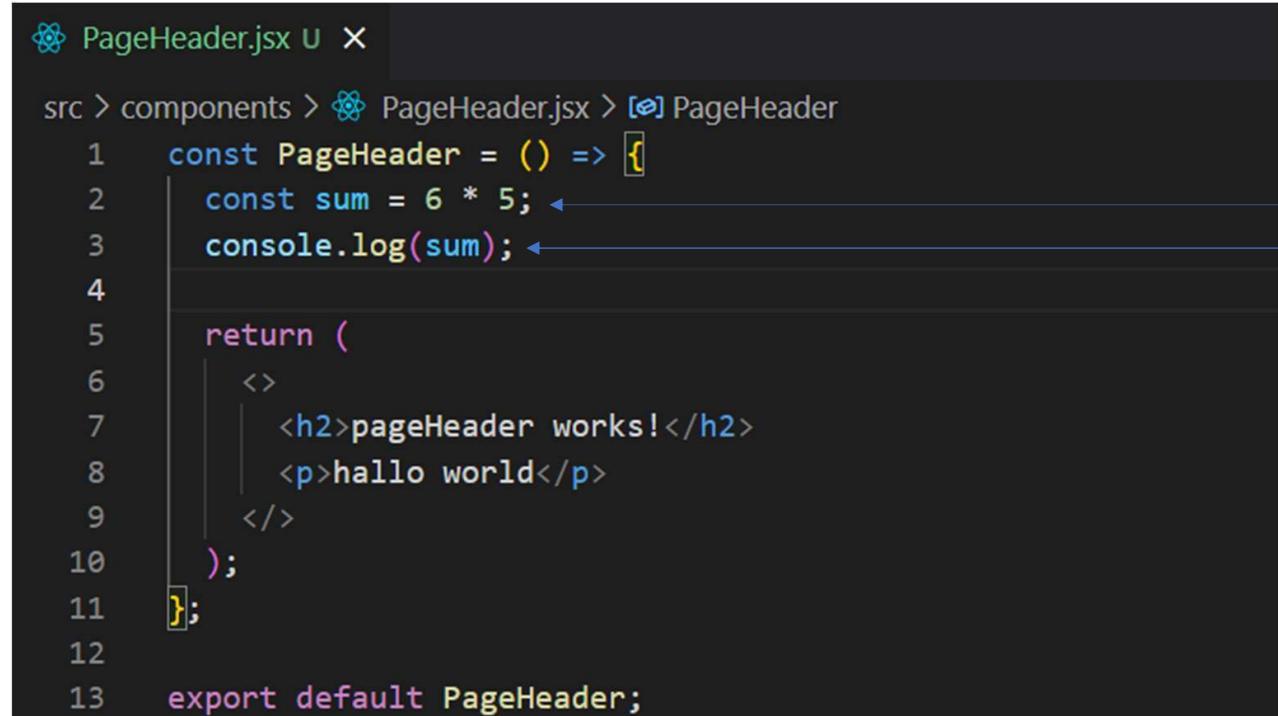
כמו בכל פונקציה גם בקומponent ניתן ליצור לוגיקה מלבד החזרת אלמנט HTML וczאת שתשפיע על האלמנט המוחזר



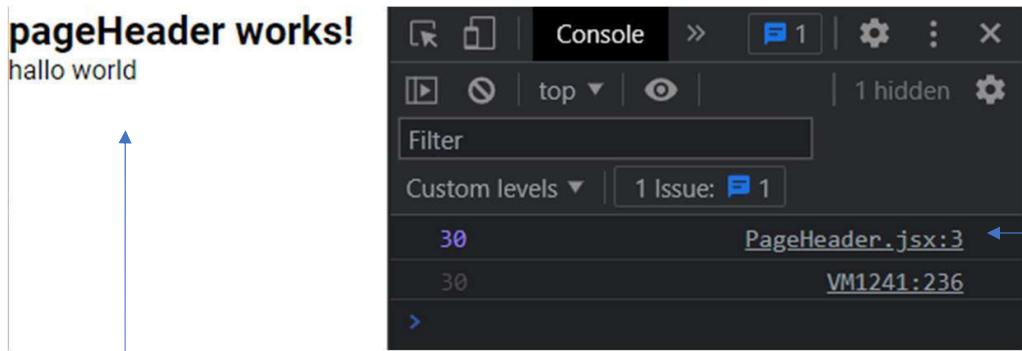
הוספה לוגיקה

כפי שניתן לראות הקומponent מתנהגת כפונקציה לכל דבר ועניין. בדוגמה שלහן:

- אני יוצר קבוע בשם sum ומשווה אותו להכפלת הספרה 6 בספרה 5
- אני מדפס את התוצאה בקונסול
- התוצאה בדף
- הדפסת ערכו של המשתנה sum שיצרתי בקונסול
- לצד תצוגת ה – HTML לגולש



```
src > components > PageHeader.jsx > PageHeader
1  const PageHeader = () => [
2    const sum = 6 * 5; ←
3    console.log(sum); ←
4
5    return (
6      <>
7        <h2>pageHeader works!</h2>
8        <p>hallo world</p>
9      </>
10 );
11 };
12
13 export default PageHeader;
```





String interpolation

יצירת איזור JAVASCRIPT באיזור המוגדר
HTML בקומפוננט



String interpolation example

פתחת אזור JAVASCRIPT באזורי המועד
ל – HTML מתבצעת על ידי פתיחה
וסגירה של סוגרים מסולסלים

דוגמה שלහן:

- בתוך ה – scope של הקומפוננט יוצרתי קבוע בשם text והשוויתי את הערך שלו למחוזת תווים
- באזורי המועד לשפת HTMLفتحתי אזור של JAVASCRIPT בעזרת פתיחה סוגרים מסולסלים ובתוכם הצבתי את שם הקבוע שיצרתי
- בעזרת string interpolation נוסףفتحתי אזור JAVASCRIPT גם בתוך אלמנט נוסף של HTML והפעם ביצעת חישוב כפי שפה JAVASCRIPT יודעת לעשות
- התוצאה בדף:
 - ניפוי שניתן לראות הטעסט הוצב במקום שהגדרתי לו
 - החישוב בוצע במקום שהגדרתי לו

PageHeader.jsx

```
src > components > PageHeader.jsx > ...
1 const PageHeader = () => {
2   const text = "Hallo world"; ←
3
4   return (
5     <>
6       <h2>pageHeader works!</h2>
7       <p>{text}</p> ←
8       <p>{5 * 6}</p> ←
9     </>
10  );
11};
12
13 export default PageHeader;
```

pageHeader works!

Haloo world
30



Styles

הוסף עיצוב לkomponent



Styles Types



INLINE



IMPORT STYLES FROM
MODULE



EXTERNAL LIBRARIES



Inline style

הדרך להזrik inline style לאלמנט HTML בקומפוננט של ריאקט היא על ידי הוספת המאפיין style ולהשווות את הערך שלו לאזרור javascript שלתוכו נעביר אובייקט עם קונפיגורציות העיצוב שאנו מעוניינים לשנות.



Inline style

בדוגמה של להלן:

```
❖ PageHeader.jsx U ●  
src > components > ❖ PageHeader.jsx > ...  
1  const PageHeader = () => {  
2  
3      const headLineStyle = { ←  
4          color: "red",  
5          fontFamily: "Roboto",  
6      };  
7  
8      return ( ←  
9          <>  
10         <h2 style={headLineStyle}>pageHeader works!</h2>  
11         <p style={{ color: "green", marginTop: "5px" }}>inline style</p>  
12     </>  
13 );  
14 };  
15  
16 export default PageHeader;
```

pageHeader works!
inline style

- ניצור קבוע בשם `headLineStyle` ונשווה את הערך שלו לאובייקט `JAVASCRIPT` שבו המאפיינים העיצובים שהוא מעוניינים לשנות וערכים כמו בכל אובייקט `JAVASCRIPT` יופיעו לאחר הנקודות.
- ניצור בתגית `HTML` הפותחת מאפיין `style` ונשווה את הערך שלו לאזורי `JAVASCRIPT` אליו נעביר את שם הקבוע שיצרנו.
- בדוגמה השנייה נעביר ישירות אובייקט `configuration` לתוך המאפיין `style` בתגית הפותחת של האלמנט `HTML`.
- התוצאה בדף

! יש לשים לב כי אם המפתח של האלמנט העיצובי בעל שני מיללים אין לחבר אותם במקף כמו שהיינו עושים בדרך כלל ב – `css` אלא להשתמש ב – `camel case syntax`



Styles from module

הדרך נוספת לשנות עיצוב של אלמנטים ב –
HTML שמחזירה הקומפוננט היא על ידי ייצירת
קובץ עיצוב "יעודי" עם מחלקות עיצוביות, הבאות
למודול של הקומפוננט ושימוש במחלקות
העיצוביות



Styles from module

דוגמה שלהן:

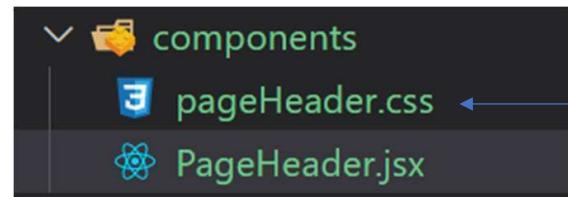
- ניצור קובץ חדש בשם pageHeader.css

- ניצור מחלקת עיצובית של css בקובץ שיצרנו

- ניבא את המודול לתוך הקובץ של הקומפוננטה

- השתמש במחלקה העיצובית שיצרנו

! יש לשים לב ל syntax של המאפיין className בפרויקט שהוחלף ל - className



pageHeader.css U X

```
src > components > pageHeader.css > ...
1   .blue {
2     color: skyblue;
3     font-weight: bold;
4 }
```

PageHeader.jsx U X

```
src > components > PageHeader.jsx > ...
1 import "./pageHeader.css";
2
3 const PageHeader = () => {
4   return <h2 className="blue">pageHeader works!</h2>;
5 }
6
7 export default PageHeader;
```



External libraries

הדרך השלישית לעיצוב אלמנטים של HTML
בקומפוננטות של ריאקט היא באמצעות היבאת
ספריות עיצוב חיצונית ושימוש במחלקות
העיצוב שלهن



Material UI

The Material Design library adapted to work with React

<https://mui.com/>

! יש לעبور על מצגת UI-material לפני שימושיכים במצגת זאת

Props

הדרך להזrik נתוניים מkomponent ab לkomponent bn





Passing string

העברת מחרוזת תווים מקומפוננט אב
לקומפוננט בן בקומפוננט מסווג פונקציה



Child Component

- ניצור קומפוננט מסווג פונקציה שתתקבל props בפרמטר אובייקט של props
- נחלץ את מפתח string מאובייקט props
- נפתח איזור של JAVASCRIPT בתוך החלך המיועד ל – HTML בקומפוננט ונציב בתוכו את הערך של המפתח שהילצנו מתוך אובייקט הprops

```
client > src > sandbox > props > ChildComp.jsx > ...
1 import { Typography, Box } from "@mui/material";
2 import React from "react";
3
4 const ChildComp = props => {
5   const { string } = props;
6
7   return (
8     <>
9       <Box
10         sx={{
11           backgroundColor: "primary.dark",
12           width: 100,
13           height: 100,
14           "&:hover": {
15             backgroundColor: "primary.main",
16             opacity: [0.9, 0.8, 0.7],
17           },
18         }}>
19         <Typography variant="body1"> child Component</Typography>
20         <Typography>{string}</Typography>
21       </Box>
22     </>
23   );
24 };
25
26 export default ChildComp;
```

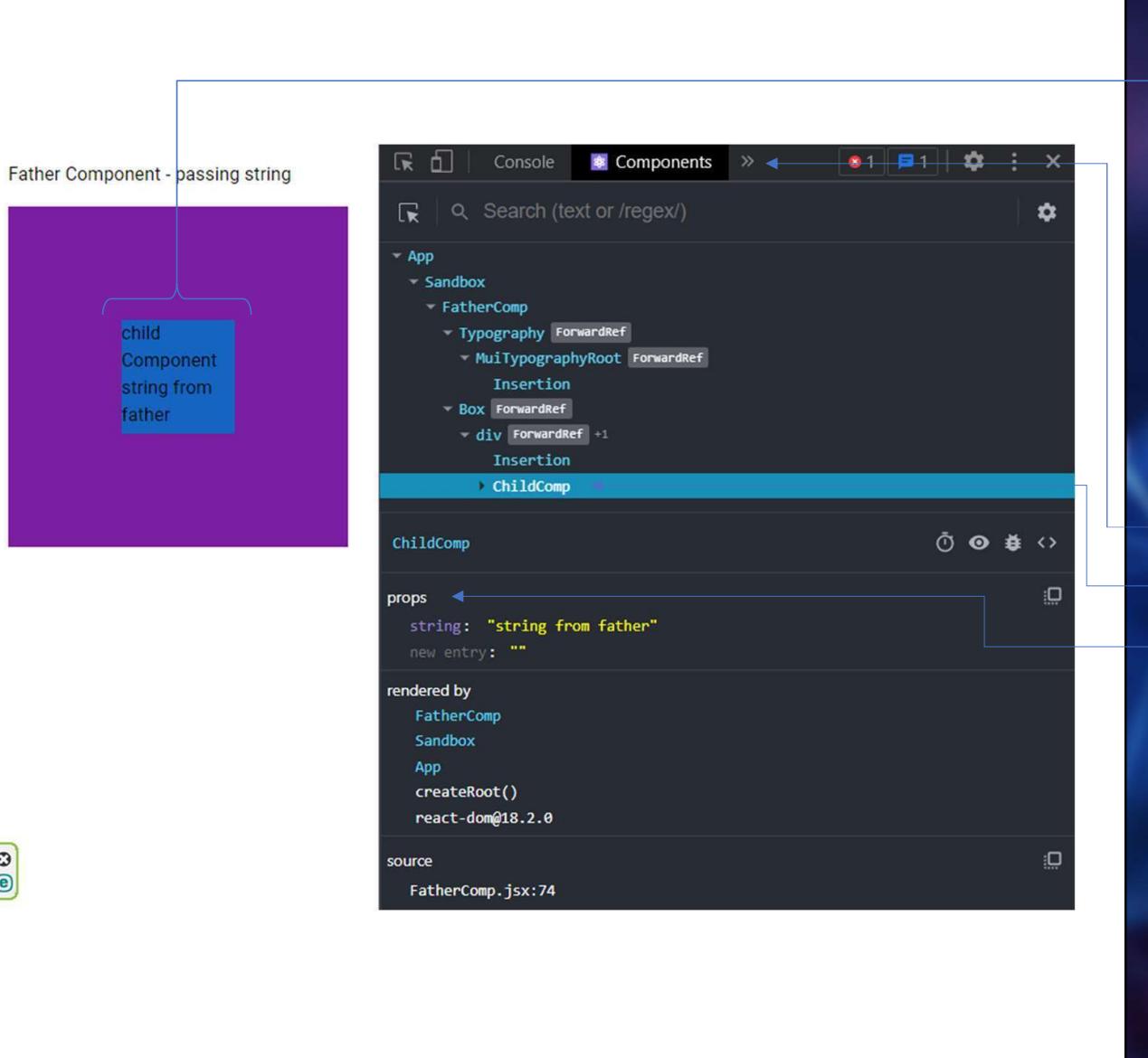
```
client > src > sandbox > props > FatherComp.jsx > ...
1  import { Box, Typography } from "@mui/material";
2  import React from "react";
3  import ChildComp from "./ChildComp";
4
5  const FatherComp = () => {
6    const string = "string from father";
7
8    return (
9      <>
10        <Typography variant="body1" m={2}>
11          {" "}
12          Father Component - passing string
13        </Typography>
14        <Box
15          sx={{
16            m: 2,
17            display: "flex",
18            justifyContent: "center",
19            alignItems: "center",
20            width: 300,
21            height: 300,
22            backgroundColor: "secondary.dark",
23          }}>
24          <ChildComp string={string} />
25        </Box>
26      </>
27    );
28  };
29
30  export default FatherComp;
```

Father component

- ניצור קומפוננט בשם **FatherComp**
- ניצור קבוע בשם **string** שערך יהיה מחוץ תווים
- נציב את קומפוננט הבן **ChildComp** בתוך החלק המיועד ל-HTML בקומפוננט האב וنعשה השמה למפתח **string** בתוך אובייקט ה-**props** ונקבע את ערכו **לקבוע** שיצרנו.

התווצהה בדף

- ניתן את הטקסט שהעבכנו מkomponent האב מוצג בתוך komponent הבן
- בנוסף בגלל שהורדנו את התוסף react tools dev能够 אנו יכולים לgesת לשונית components
- ללחוץ על הקומponent שמעניינת אותנו
- ולקבל בין היתר את המפתחות והערכיהם שמועברים באובייקט props





Passing Object

העברת אובייקט מkomponent ab לkomponent ב
בקומפוננט מסווג פונקציה וחילוץ המפתחות
והערכים ממנו



Child Component

- ב글 שkomponent מסוג פונקציה מתנהגת כמו כל פונקציה ב – JavaScript אני יכול לחלץ מאובייקט props את המפתחות הרלוונטיים ישר בתוך הפרמטר של הפונקציה
- נחלץ את מפתחות first, last מתוך props המפתח name שבאובייקט ה - props
- נפתח איזור של JavaScript בתחום החלק המיועד ל – HTML בkomponent ונציב בתוכו את הערכים של המפתחות שהיצנו מתוך אובייקט הprops

```
47 const ChildComp = ({ name }) => { ←
48   const { first, last } = name; ←
49   return (
50     <>
51       <Box
52         sx={{
53           backgroundColor: "primary.dark",
54           width: 100,
55           height: 100,
56           "&:hover": {
57             backgroundColor: "primary.main",
58             opacity: [0.9, 0.8, 0.7],
59           },
60         }}>
61         <Typography>{first}</Typography> ←
62         <Typography>{last}</Typography> ←
63       </Box>
64     </>
65   );
66 };
```

Father Component

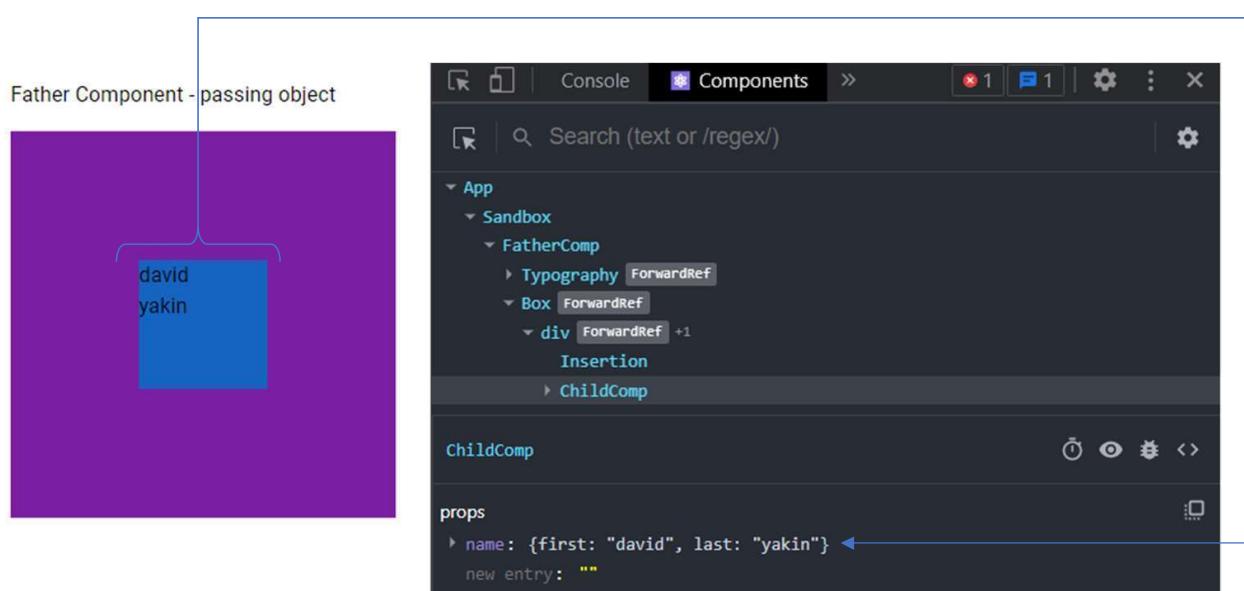
- ניצור קבוע בשם name שערך יהיה אובייקט עם מפתחות וערכים
- נעשה השמה למפתח name בתווך אובייקט ה – props ונקבע את ערכו קבוע name שיצרנו.

```
31 const FatherComp = () => {
32   const name = { first: "david", last: "yakin" };  

33
34   return (
35     <>
36       <Typography variant="body1" m={2}>
37         {" "}
38         Father Component - passing object
39       </Typography>
40       <Box
41         sx={{
42           m: 2,
43           display: "flex",
44           justifyContent: "center",
45           alignItems: "center",
46           width: 300,
47           height: 300,
48           backgroundColor: "secondary.dark",
49         }}>
50         <ChildComp name={name} />
51       </Box>
52     </>
53   );
54 }
```

התוצאה בדף

- ניתן את הכתוב שהערכנו מקומפוננטה האב לקומפוננטה הבן בתוך אובייקט מוצג בקומפוננטה הבן
- וכי אובייקט ה – `props` מכיל עצם מפתח בשם `name` שהערך שלו זה האובייקט שיצרנו





Sending two keys

הדרך להעביר יותר מפתח אחד לאובייקט
הפרופו



Child Component

- ב글ל שkomponenT מסוג פונקציה מתנהגת כמו כל פונקציה ב – JAVASCRIPT וכל לחץ מאובייקט props מספר מפתחות first, last
- נפתח איזור של JAVASCRIPT בתוך החלק המיועד ל – HTML בkomponenT ונציב בתוכו את הערךIM של המפתחות שחילצנו מתוך אובייקט הprops

```
69 const ChildComp = ({ first, last }) => { ←
70   return (
71     <>
72       <Box
73         sx={{
74           backgroundColor: "primary.dark",
75           width: 100,
76           height: 100,
77           "&:hover": {
78             backgroundColor: "primary.main",
79             opacity: [0.9, 0.8, 0.7],
80           },
81         }}>
82         <Typography>{first}</Typography>
83         <Typography>{last}</Typography>
84       </Box>
85     </>
86   );
87 };
88
```

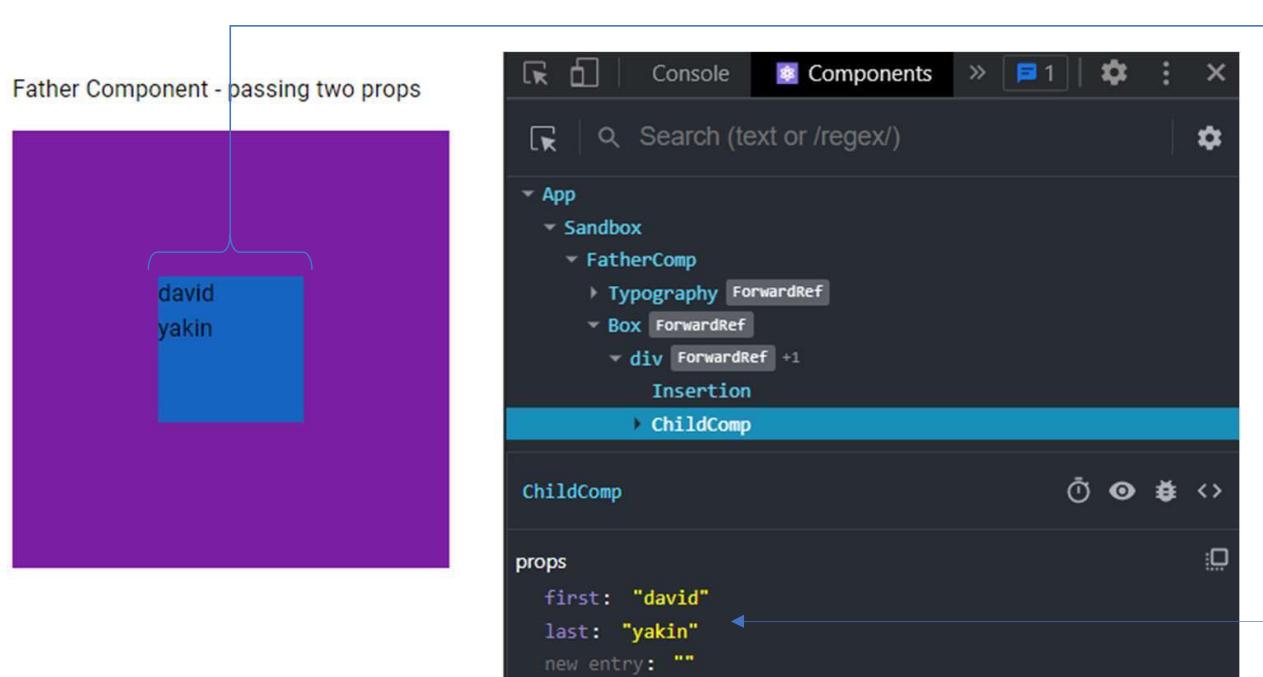
```
57 const FatherComp = () => {
58   const name = { first: "david", last: "yakin" };
59
60   return (
61     <>
62       <Typography variant="body1" m={2}>
63         {" "}
64         Father Component - passing two props
65       </Typography>
66       <Box
67         sx={{
68           m: 2,
69           display: "flex",
70           justifyContent: "center",
71           alignItems: "center",
72           width: 300,
73           height: 300,
74           backgroundColor: "secondary.dark",
75         }}>
76         <ChildComp first={name.first} last={name.last} /> ←
77       </Box>
78     </>
79   );
80 };
```

Father Component

- ניצור קבוע בשם name שערך יהיה אובייקט עם מפתחות וערכים
- הפעם נעביר כל מפתח מהאובייקט שিירנו לתוך מפתח משלה באובייקט הפכנו

התוצאה בדף

- ניתן את הכתוב שהערכנו מקומפוננט האב לקומפוננט הבן בתוך אובייקט מוצג בקומפוננט הבן
- וכי אובייקט ה – props מכיל עצ מפתח בשם name שהערך שלו זה האובייקט שיצרנו



Props

```
const card = {
  _id: "63765801e20ed868a69a62c4",
  title: "first",
  subtitle: "subtitle",
  description: "testing 123",
  phone: "050-0000000",
  email: "test@gmail.com",
  web: "https://www.test.co.il",
  image: {
    url: "assets/images/
business-card-top-image.jpg",
    alt: "Business card image",
  },
  address: {
    state: "",
    country: "country",
    city: "tel-aviv",
    street: "Shinkin",
    houseNumber: 3,
    zip: 1234,
  },
  bizNumber: 1_000_000,
  user_id: "63765801e20ed868a69a62c2",
};
```

Business-cards-app

- בஹמשך למשימת Card במצגת UI-HTML צור בתיקייה בנתיב `src/cards/components/card` שלושה קבצים נוספים:
 - CardHead – קומפוננט זה יכול תמונה שאת הערכים שלו (url, alt) מקבל מאובייקט הprops
 - CardBody – קומפוננט זה יכול כותרת ראשית ומשנית לכרטיס, חוץ ושלוש שורות טקסט כפי שמופיע בדוגמה שבסך הבא. את הערכים לשדות הטקסט עליו לקבל מפתח בשם `card` מתוך אובייקט ה- `props`
 - CardActionBar – איזור זה בכרטיס יכול אייקון של לב
- בקובץ `Card`
 - צור קבוע בשם `card` שי יכול את המפתחות והערכים המופיעים בדוגמה משמאל
- הציב את שלושת הקומפוננטות שיצרת בתוך הקומפוננט `.Card`.
- העבר לקומפוננטות הבנים את המידע הדרוש להם באמצעות אובייקט הprops על מנת שיוכלו להציגו לגולש

משימת Props

חלק ב'



forth

subtitle

Phone: 050-0000000

Address: Shinkin 3 tel-aviv

Card Number: 4000000



Loops

הדרך לבצע לולאות ב React



Map Loop

React בחרה להשתמש בMETHOD map
בשביל לבצע לולאות על מערכם באזור
המיועד ר-HTML

בדוגמה שללן:

- צרנו קבוע בשם `arrayOfString` והשווינו את ערכו לערך של מחזורות תווים בחלק המיועד ל-HTML
- אנחנו מבצעים לולאה על הקבוע `arrayOfString` באמצעות METHOD map שמקבלת עד שלושה פרמטרים:
 - Item – האיבר במערך
 - Index – מספר האינדקס של האיבר במערך
 - array – המערך שעליו נערך הא Iteration
- בתוך הלולאה אני מדפיס את המערך ומציג גולש כתוב שמקורו במערך
- כל אלמנט שאנו מכפילים בא Iteration צריך לקבל את המאפיין `key` שצריך להיות ייחודי.

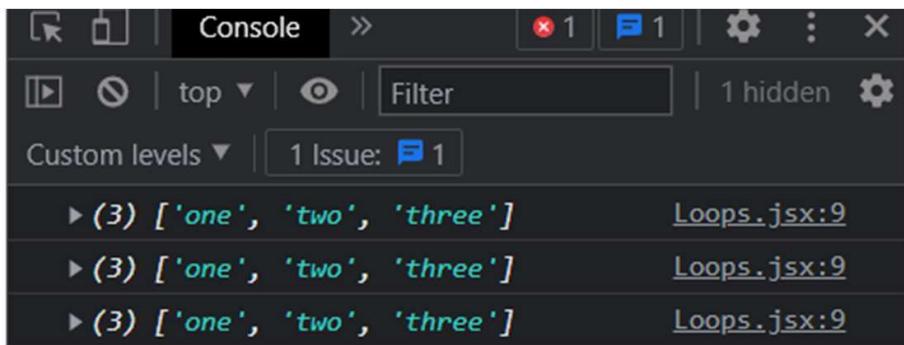
```
Loops.jsx X
client > src > sandbox > Loops.jsx > ...
1 import React from "react";
2 import { Box } from "@mui/material";
3
4 const Loops = () => {
5   const arrayOfString = ["one", "two", "three"];
6   return (
7     <Box m={2}>
8       {arrayOfString.map((item, index, array) => {
9         console.log(array);
10        return <div key={index}>item: {item}</div>;
11      })}
12     </Box>
13   );
14 };
15
16 export default Loops;
```

התוצאות בדף

React בחרה להשתמש בمتודת map
בשביל לבצע לולאות על מערכים באזור
המיועד ר HTML

- וכי אובייקט ה – props מכיל עצם מפתח
בשם name שהערך שלו זה האובייקט
שיצרנו

item: one
item: two
item: three



```
Console  »  ✖ 1  ⚠ 1  ⚙  ⌂  ✕
▶  top  Filter  1 hidden  ⚙
Custom levels  ▾  1 Issue: ⚠ 1
▶ (3) [ 'one', 'two', 'three' ]  Loops.jsx:9
▶ (3) [ 'one', 'two', 'three' ]  Loops.jsx:9
▶ (3) [ 'one', 'two', 'three' ]  Loops.jsx:9
```

משימת Map



Business-cards-app

- צור את הנתיב הבא:
`src/cards/components/Cards.jsx`
 - `Cards.jsx`
 - צור מערך עם שלושה אובייקטים שמייצגים כרטיסים
(המפתחות האובייקטים הללו צריכים להיות תואימים
למפתחות של אובייקט הכרטיים מהתרגיל הקודם)
 - השתמש במתודת `map` כך שעל כל איבר במערך תציג
לגולש כרטיס בעזרת הקומponent `Card.jsx`.
 - בדוק בדף כי אכן הכרטיסים מוצגים לגולש
- ! על הקומponent `Card` לקבל באובייקט ה-`props` כרטיס `card`
במקום הקבוע `card` שיצרנו בתרגיל הקודם.

Conditional Rendering

תצוגות מידע שונות כאשר יש מידע להציגה וכאשר אין

! יש לעبور על החלק של Layout במצגת של UIW לפני שימושים במצגת היזאת



```
client > src > cards > components > Cards.jsx > ...
1 import { Container, Stack, Typography } from "@mui/material";
2 import React from "react";
3 import CardComponent from "./card/Card";
4
5 const Cards = () => {
6   // const cards = [ ...
71
72   const cards = [];
73   if (!cards.length)
74     return (
75       <Typography m={2}>
76         | Oops... it seems there are no business cards to display
77       </Typography>
78     );
79   return (
80     <Container>
81       <Stack
82         gap={2}
83         direction="row"
84         my={2}
85         flexWrap="wrap"
86         justifyContent="center">
87           {cards.map((card, i) => (
88             <CardComponent key={i} card={card} />
89           ))}
90         </Stack>
91       </Container>
92     );
93   };
94
95 export default Cards;
```

Conditional Rendering

לעתים המידע שברצוננו להציג לגולש חסר ונרצה לעדכן בכר את הגולש.

- נשים לرجע את הקבוע cards שיצרנו בתוך הערה
- ኒצור קבוע שני עם אותו השם אך הפעם נשווה את ערכו למערך ריק (מצב זה מדמה כאשר יש שורת חיפוש והמשתמש חיפש משהו שלא נמצא או שאין את המידע שהוא מחפש במאגר המידע)

נתנה שאם אין אורך לערך cards הקומponent יעצור ויחזיר תצוגה לגולש שתכלולאזור טקסט עם מחזורת תווים

Events

הדרך להאזין לאירועים ולהפעיל מטודות בעקבותיהם





JAVASCRIPT EVENTS

נותנת תמיכה לכל סוגי האירועים ב -
React JAVASCRIPT ובכל אחד מהם ניתן להפעיל
פונקציה או מטודה אחרת

<https://developer.mozilla.org/en-US/docs/Web/Events>



onClick event

בוגמה שלהן:

- יצרנו קומפונט בשם **OnClick**

- יצרנו קבוע בשם **handleClick** שמשווה ערך לפונקציה אטומית כשתופעל תזדמיס בקונסול מחרוזת תווים

- יצרנו כפטור שיאזין לאיירע **onClick** ויפעל את הפונקציה **handleClick** ברגע שהאיירע יקרה.

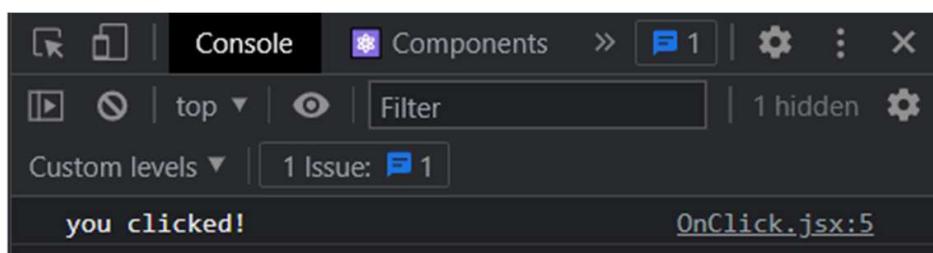
- התוצאה בדף:

- כשנלחץ על הכפטור תודפס בקונסול מחרוזת התווים מתוך הפונקציה **handleClick**

OnClick.jsx X

```
client > src > sandbox > events > OnClick.jsx > ...
1 import React from "react";
2 import Button from "@mui/material/Button";
3
4 const OnClick = () => {
5   const handleClick = () => console.log("you clicked!");
6
7   return (
8     <Button onClick={handleClick} variant="outlined" sx={{ m: 2 }}>
9       Click
10      </Button>
11    );
12  };
13
14 export default OnClick;
```

CLICK





Function invocation with parameters

הפעלת פונקציה עם פרמטרים



Function with parameters

בדוגמה שללן:

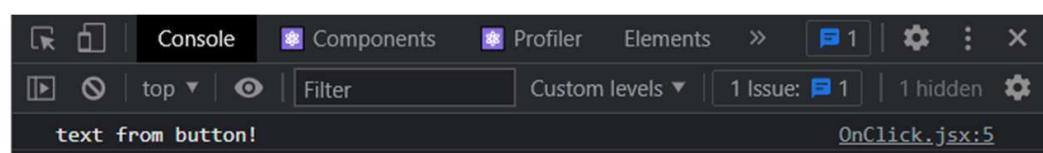
- הקבוע handleClick שווה לפונקציה אנונימית שמקבלת פרמטר text מסוג MERCHANTABILITY תווים ומדפיסה אותו בקונסול
- אני מעביר בערך של האירוע onClick פונקציה call back אונימית שכשהאירוע יקרה היא תופעל ובתוֹר הֆונקציה אני מפעיל את הֆונקציה handleClick עם הערך שאני מעוניין שיודפס בקונסול
- התוצאה בדףן:

כשנלחץ על הכפתור תודפס בקונסול MERCHANTABILITY התווים מתוך הֆונקציה handleClick

אם עבור לארוע ישירות את הפעלת המטודה handleClick עם הפרמטר ללא הֆונקציה האונימית היא תופעל עם יצירת הקומponent ולא מתוֹר שהאירוע יקרה

```
client > src > sandbox > events > OnClick.jsx > ...
1 import React from "react";
2 import Button from "@mui/material/Button";
3
4 const OnClick = () => {
5   const handleClick = text => console.log(text);
6
7   return (
8     <Button
9       onClick={() => handleClick("text from button!"})
10      variant="outlined"
11      sx={{ m: 2 }}
12      Click
13     </Button>
14   );
15 };
16
17 export default OnClick;
```

CLICK





Catching the event and passing it to the function

תפיסה האירוע והעברתו ל함קציית ה – call
back



Catching event

דוגמה שלהן:

- הקבוע handleClick שווה לפונקציה אוניברסלית שמקבלת פרמטר e מסוג אירוע ומדפיסה אותו בקונסול את האלמנט שתפס את האירוע

- אני מעביר בערך של האירוע onClick פונקציה call back אוניברסלית שמקבלת את האירוע ומבצעת את הפונקציה handleClick עימם

התוצאה בדף:

- כשנלחץ על הכפתור יודפס בקונסול האובייקט שהاذין לאירוע והפעיל את handleClick הפונקציה

The screenshot shows a code editor with a file named `onClick.jsx`. The code defines a functional component `OnClick` that logs the target of a click event to the console. It uses the Material-UI `Button` component with an `outlined` variant and a `size="medium"` style. A `CLICK` button is shown in the browser's developer tools' Elements tab, with the element's class and type attributes highlighted.

```
client > src > sandbox > events > ⚛ onClick.jsx > ...
1 import React from "react";
2 import Button from "@mui/material/Button";
3
4 const OnClick = () => {
5   const handleClick = e => console.log(e.target);
6
7   return (
8     <Button onClick={e => handleClick(e)} variant="outlined" sx={{ m: 2 }}>
9       Click
10      </Button>
11    );
12  };
13
14 export default OnClick;
```

CLICK

The developer tools' Elements tab shows the DOM structure of the `CLICK` button. The element node is highlighted, showing its class, size, and type attributes.

```
<button class="MuiButtonBase-root MuiButton-root MuiButton-outlined MuiButton-outlinedPrimary MuiButton-sizeMedium MuiButton-outlinedSizeMedium MuiButton-root MuiButton-outlined MuiButton-outlinedPrimary MuiButton-sizeMedium MuiButton-outlinedSizeMedium css-19skcmy-MuiButtonBase-root-MuiButton-root" tabindex="0" type="button">...</button> (flex)
```

הדרך השניה להעבר

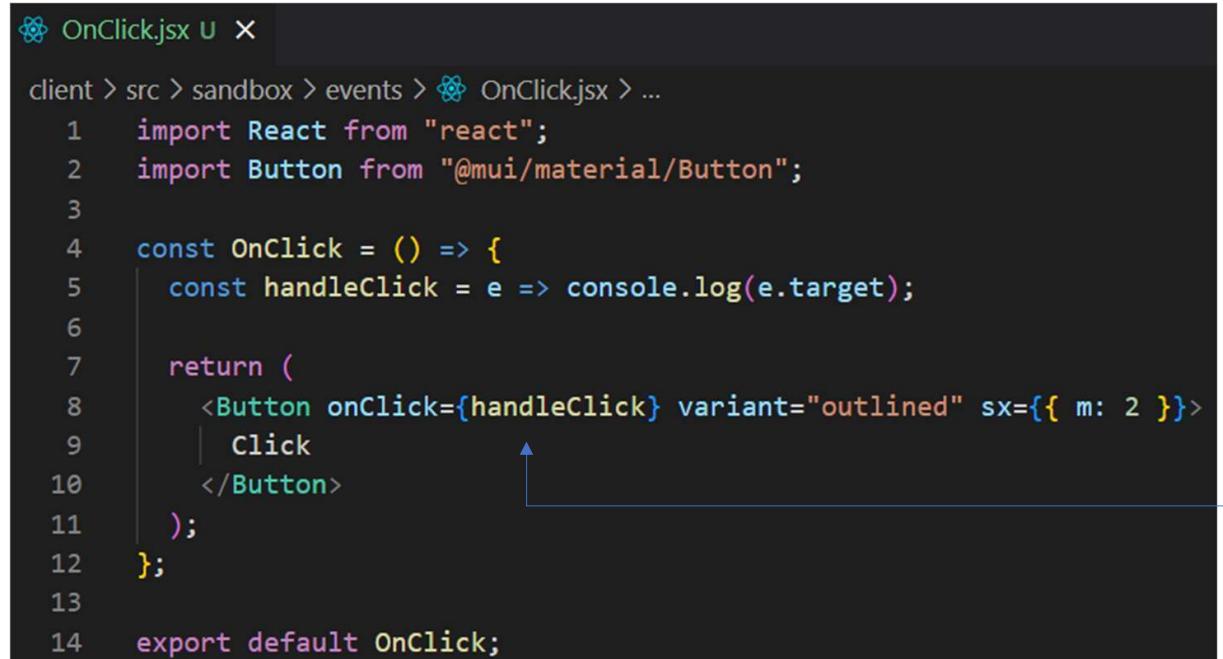
איירונן לפונקציה

בדוגמה שללן:

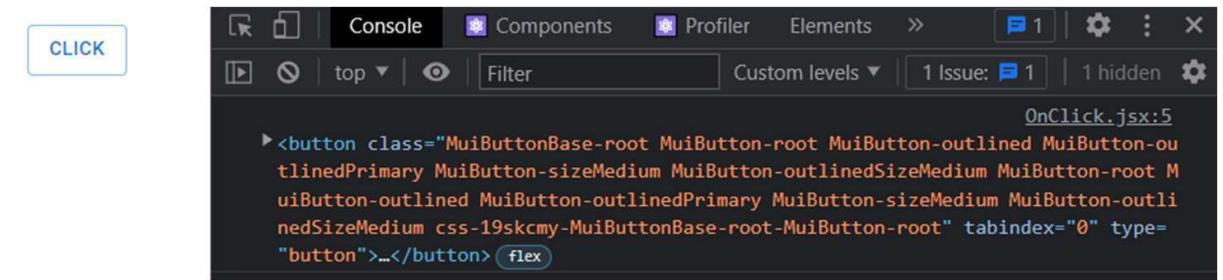
- הפעם אני כבירול לא מעביר לפונקציה handleClick פרמטר אלא מעביר אותה כפונקציית javascript callback. אולם javascript callback זאת תעביר לפונקציה את האירוע ואוכל להדפיסו בקונסול

• התוצאה בדף:

- כナルח על הכפתור יודפס בקונסול האובייקט שהאזין לאיירונן והפעיל את הפונקציה handleClick



```
client > src > sandbox > events > OnClick.js > ...
1 import React from "react";
2 import Button from "@mui/material/Button";
3
4 const OnClick = () => {
5   const handleClick = e => console.log(e.target);
6
7   return (
8     <Button onClick={handleClick} variant="outlined" sx={{ m: 2 }}>
9       Click
10      </Button>
11    );
12  };
13
14 export default OnClick;
```





Raising Events

ניתן להעביר פונקציה באובייקט הпроופס כך
שיקרא אירוע בקומפוננט הבן הוא יפעיל מטודה
בקומפוננט האב



הפעלת מטודה בקומפוננט

האב קומפוננט הבן

יהו מקרים בהם נרצה שקומפוננט הבן יוכל להפעיל מטודה בקומפוננט הבא.

בקומפוננט האב

- אני יוצר קבוע בשם handleClick שמשווה ערך לפונקציה אונימית שמדפיסה בקונסול מחרוזת תווים
- אני עושה השמה למפתח בשם handleClick באובייקט הförופס ומשווה את הערך שלו למטרודת handleClick שיצרתי לעיל.

בקומפוננט הבן

- אני מחלץ את המפתח מForObject הförופס
- אני מאדין לאיירוע onClick שתפעיל את מטרודת handleClick

```

83 const FatherComp = () => {
84   const handleClick = () => console.log("you clicked!");
85 
86   return (
87     <Box
88       sx={{
89         m: 2,
90         display: "flex",
91         justifyContent: "center",
92         alignItems: "center",
93         width: 300,
94         height: 300,
95         backgroundColor: "secondary.dark",
96       }}>
97       <ChildComp handleClick={handleClick} />
98     </Box>
99   );
100 }

```

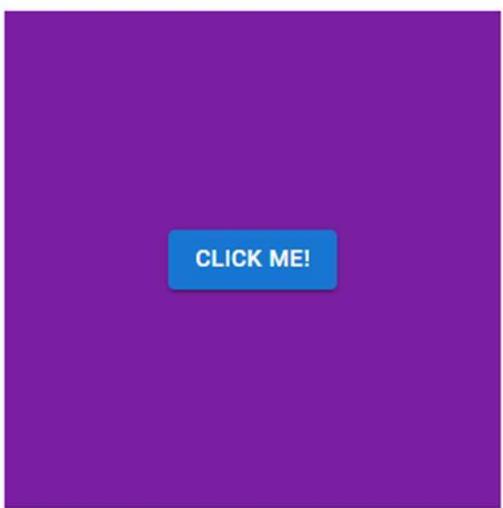
```

90 const ChildComp = ({ handleClick }) => {
91   return (
92     <Button onClick={handleClick} variant="contained">
93       click me!
94     </Button>
95   );
96 }

```

התווצה בדף

כשאני לוחץ על הceptor בкомпонент הבן
מופעלת המטודה handleClick שמדפסה
בקונסול את מחרוזת התווים שקבעתי
ראש



Events



Business-cards-app

- צור בקומפוננט `xsj.Cards` קבוע בשם `handleCardDelete` שיבוא שווה ערך לפונקציה אונומית שתתקבל בפרמטר `_id` מסוג מספר ותדפיס בקונסול את מחרוזת התווים "no." ונוסף את המספר שמופיע ב מפתח `_id`
- העבר באובייקט הפרופס את מטודות `onLike` `onDelete` מkomponent האב `xsj.Cards` דרך קומפוננט הבן `xsj.Card`. ועד לkomponent שמתעסקת עם Action Buttons
- בלחיצה על אייקון הלב הדפס את מחרוזת התווים "you liked card no: " ונוסף את הערך שמופיע ב מפתח `_id`
- בלחיצה על אייקון פח האשפה הדפס את מחרוזת התווים "you liked card no: " ונוסף את הערך שמופיע ב מפתח `_id`
- בלחיצה על אייקון עריכת הCARTEIS הדפס את מחרוזת התווים "edit card no: " ונוסף את הערך שמופיע ב מפתח `_id`

propTypes

Strong Typing in React

<https://reactjs.org/docs/typechecking-with-proptypes.html>





Q Definition

Runtime type checking for React props
and similar objects.



Installation

`npm i prop-types`



Types

הערות	PropTypes	ערך לבדיקה	או'
	.string	"string"	.1
	.number	5	.2
	.bool	true/false	.3
	.object	{ }	.4
מגדיר את סוג ערכי המפתחות באובייקט	.objectOf()		.5
	.array	[]	.6
מגדיר את סוג האיברים במערך	.arrayOf()		.7
	.func	()=>{ }	.6
אחד מהסוגים המפורטים בתוך המערך של הפונקציה	.oneOfType([]),		
אחד מהתווים שמצוועים במערך בלבד	.oneOf(['News', 'Photos'])		.13
instance of a class	.instanceOf(new Class)		
ערך דיפולטיבי	. defaultProp		
יצור סימן חדש	.symbol	Symbol()	.7
	.bigint		.8
	.node		.9
	.element		.10
	.elementType		.11
	.shape({ })		.16
מודוא שלא הועברו מפתחות שלא נמצאות בסוג הפרויקט	.exact({ })		.17
	.isRequired		.18
	any		.19



PropTypes Errors

תצוגת שגיאות של PropTypes



PropTypes Error

על מנת ליצור שגיאה של PropTypes

בקומפוננט האב

- נציב את קומפוננט הבן בתוך קומפוננט האב אך לא נעביר לה את המפתחות שהיא זקוקה להם באובייקט הפרווף

בקומפוננט הבן

- יצירת מופע של מחלקת PropTypes מתוך הספרייה שייבאנו "prop-types"

- יצירת קומפוננט בשם PropTypeComponent שצריכה לקבל string באובייקט הפרווף מפתח בשם string

- אני עושה השמה למפתח PropTypes בquito לkomponent PropTypeComponent שיצרנו ומשווה את הערך שלו לאובייקט שיבדק בעזרת המופע של מחלקת PropTypes את סוג הערכים של המפתח

! ב글 שפונקציה מאחוריו הקלעים היא אובייקט אני יכול לעשות השמה למפתחות שלא

```
client > src > sandbox > propTypes > FatherPropTypes.jsx > FatherPropTypes.js
1 import React from "react";
2 import PropTypeComponent from "./PropTypeComponent";
3
4 const FatherPropTypes = () => {
5   return <PropTypeComponent />; ←
6 }
7
8 export default FatherPropTypes;
```

```
client > src > sandbox > propTypes > PropTypeComponent.jsx > ...
1 import React from "react";
2 import PropTypes from "prop-types"; ←
3
4 const PropTypeComponent = ({ string }) => { ←
5   return <div>PropTypeComponent</div>;
6 }
7
8 PropTypeComponent.propTypes = { ←
9   string: PropTypes.string.isRequired,
10 };
11
12 export default PropTypeComponent;
```

התוצאות בדף

בגלל שלא העברתי מקומפוננט האב
לקומפוננט הבן את המפתח שעשייתי עלי^ו
בדיקה באמצעות `propTypes` נזרקת לי על

כך שגיאה

The screenshot shows a browser's developer tools console. The title bar says "PropTypeComponent". The console tab is active. There is one warning message: "Warning: react-jsx-dev-runtime.development.js:87 Failed prop type: The prop `string` is marked as required in `PropTypeComponent`, but its value is `undefined`." The message includes a stack trace:

```
* Warning: react-jsx-dev-runtime.development.js:87
Failed prop type: The prop `string` is marked as required in `PropTypeComponent`, but its value is `undefined`.
    at PropTypeComponent (http://localhost:3000/static/js/bundle.js:392:5)
    at FatherPropTypes
    at Sandbox
    at div
    at App
```



Main Types

סוגי הערכים המרכזיים



Main Types

בקומפוננט האב

- ציב את קומפוננט הבן בתוך קומפוננט האב וubah לה את המפתחות שהיא זוקקה להם באובייקט הprops

בקומפוננט הבן

- הקומפוננט מקבל בפרמטר את האובייקט props
- נחלץ את הערכים מתוך האובייקט props
- נעשה השמה למפתח propTypes בתוך komponentPropTypes שיצרנו ומשווה את הערך שלו לאובייקט שיבדק את המפתחות שאני מעוניין שייהו באובייקט props ואת הערכים שלהם

```
9  const FatherPropTypes = () => {
10    const obj = { key: "value" };
11    return (
12      <PropTypeComponent
13        string="string"
14        number={2}
15        boolean={true}
16        object={obj}
17        array={}
18        cb={console.log}
19      />
20    );
21  };


```

```
13  const PropTypeComponent = props => {
14    const { string, number, boolean, object, array, cb } = props;
15    return <div>PropTypeComponent</div>;
16  };

17

18  PropTypeComponent.propTypes = {
19    string: PropTypes.string,
20    number: PropTypes.number,
21    boolean: PropTypes.bool,
22    object: PropTypes.object,
23    array: PropTypes.array,
24    cb: PropTypes.func,
25  };


```



ArrayOf & ObjectOf Types

הדרך לפרט מה יכולו אובייקטים ומערכות
באמצעות PropTypes



arrayOf & objectOf

בקומפוננט האב

- נציב את קומפוננט הבן בתוך קומפוננט האב ונעביר לה את המפתחות שהוא זקוקה להם באובייקט הprops

בקומפוננט הבן

- הקומפוננט מקבלת בפרמטר את האובייקט props

- נחלץ את הערך ממתוך האובייקט props
- נעשה השמה למפתח propTypes בתוך קומפוננט PropTypeComponent לkomponent שיצרנו ונסווה את הערך שלו לאובייקט שיבדק את המפתחות שאנו מעוניין שהיו באובייקט props ואת הרכים שלהם

```
25 const FatherPropTypes = () => {
26   const obj = { key: "value" };
27   const array = [1, 2, 3];
28   const arrayOfObjects = [{ key: false }];
29
30   return (
31     <PropTypeComponent
32       object={obj}
33       array={array}
34       arrayOfObject={arrayOfObjects}
35     />
36   );
37 }
```

```
28 const PropTypeComponent = props => {
29   const { object, array, arrayOfObject } = props;
30   console.table(props);
31   return <div>PropTypeComponent</div>;
32 };
33
34 PropTypeComponent.propTypes = [
35   object: PropTypes.objectOf(PropTypes.string),
36   array: PropTypes.arrayOf(PropTypes.number),
37   arrayOfObject: PropTypes.arrayOf(PropTypes.objectOf(PropTypes.bool)),
38 ];
```



oneOfType vs oneOf

יש ביכולתנו לקבוע מספר סוגים של ערכים או לקבוע את הערכים באופן ליטרלי



oneOfType & oneOf

בקומפוננט האב

- נציב את קומפוננט הבן בתוך קומפוננט האב ונעביר לה את המפתחות שהוא זקוקה להם באובייקט הprops

בקומפוננט הבן

- הקומפוננט מקבלת בפרמטר את האובייקט props
- נחלץ את הערך מהתוך האובייקט props
- נעשה השמה למפתח propTypes בתוך קומפוננט PropTypeComponent שיצרנו ומשווה את הערך שלו לאובייקט שיבדק את המפתחות שאינו מעוניין שהיו באובייקט props ואת הרכים שלהם

```
40 const FatherPropTypes = () => {
41   return (
42     <>
43       <PropTypeComponent one="string" />
44       <PropTypeComponent one={2} />
45       <PropTypeComponent two="a" />
46       <PropTypeComponent two="c" />
47     </>
48   );
49 }
```

```
41 const PropTypeComponent = props => {
42   console.table(props);
43   return <div>PropTypeComponent</div>;
44 };
45
46 PropTypeComponent.propTypes = {
47   one: PropTypes.oneOfType([PropTypes.string, PropTypes.number]),
48   two: PropTypes.oneOf(["a", "b"]),
49 };
```

התוצאות בדף

אם מקבלים שגיאה בגל שניסינו להעביר
במפתח two באובייקט הפרויקט מחרוזת
תווים שאינה אחת מחרוזות התווים
שהגדכנו

PropTypeComponent
PropTypeComponent
PropTypeComponent
PropTypeComponent

The screenshot shows the React DevTools Components tab. At the top, there are tabs for Console, Components, and a search/filter bar. Below the tabs, it says "Custom levels" and "1 Issue". A red warning icon indicates one issue. The main area shows a component tree with "PropTypeComponent" at the root. A warning message is displayed:

undefined PropTypeComponent.jsx:42

✖ Warning: Failed [react-jsx-dev-runtime.development.js:87](#)
prop type: Invalid prop `two` of value `c` supplied to
`PropTypeComponent`, expected one of ["a", "b"]. ←
at PropTypeComponent
at FatherPropTypes
at Sandbox
at div
at App



Exact & isRequired

יש ביכולתנו לבדוק אם באובייקט הпроוף יש
בדוק את הערכים שאנו מבקשים או לחליפין
לחיבב העברת מפתח ספציאלי



Exact

בקומפוננט האב

- נubby לה את המפתחות שהוא נדרש
לهم באובייקט הprops אולם נשים
פתח אחד יותר מיד' בתוך האובייקט
שאנו מעבירים

בקומפוננט הבן

- אנו קובעים כי יש לקבל מפתח בשם obj
באובייקט הprops והוא חייב להיות עם
المפתחות והערכים הבאים

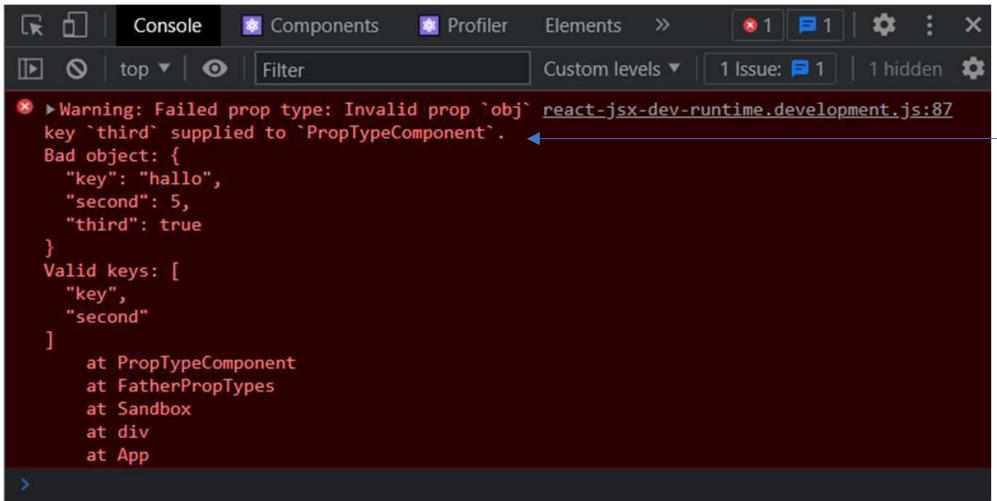
```
52 const FatherPropTypes = () => {
53   const obj = { key: "hallo", second: 5, third: true };
54   return <PropTypeComponent obj={obj} />;
55 };

52 const PropTypeComponent = props => {
53   return <div>PropTypeComponent</div>;
54 };
55

56 PropTypeComponent.propTypes = {
57   obj: PropTypes.exact({←
58     key: PropTypes.string,
59     second: PropTypes.number,
60   }),
61 };
```

התוצאות בדף

בגלל שהעבכנו מפתח שלישי שלishi
זורקתו לנו הודעת שגיאה על כך



The screenshot shows the React DevTools Components tab for a component named "PropTypeComponent". A warning message is displayed: "Warning: Failed prop type: Invalid prop `obj` supplied to `PropTypeComponent`." The message points to the source code at react-jsx-dev-runtime.development.js:87. The code snippet shows a prop object with keys "key", "second", and "third", and a list of valid keys: "key" and "second". The stack trace indicates the prop was passed from App, div, Sandbox, FatherPropTypes, and PropTypeComponent.

```
PropTypeComponent
Warning: Failed prop type: Invalid prop `obj` supplied to `PropTypeComponent`.
    at PropTypeComponent
    at FatherPropTypes
    at Sandbox
    at div
    at App
>
```

```
Bad object: {
  "key": "hallo",
  "second": 5,
  "third": true
}
Valid keys: [
  "key",
  "second"
]
```

isRequired

קומפוננט האב

- נציב את קומפוננט הבן בתוך קומפוננט האב אך לא נעביר לה את המפתח שהיא זקוקה לו באובייקט הprops

קומפוננט הבן

- אני מגדיר את המפתח two כחובה על ידי שימוש המפתח.isRequired לאחר הגדרת סוג הערך המבוקש למפתח

התוצאה בדף

- הודעת השגיאה של PropTypes בגלל שלא העברנו את המפתח שהגדרנו כחובה

```
58 const FatherPropTypes = () => {  
59   return <PropTypeComponent />; ←  
60 };
```

```
63 const PropTypeComponent = props => {  
64   return <div>PropTypeComponent</div>;  
65 };  
66  
67 PropTypeComponent.propTypes = {  
68   two: PropTypes.string.isRequired, ←  
69 };
```

```
✖ Warning: react-jsx-dev-runtime.development.js:87  
Failed prop type: The prop `two` is marked as required  
in `PropTypeComponent`, but its value is `undefined`.
```



Shape Any & defaultProps

יצירת interface של אובייקט



Shape

בעזרת `PropTypes.shape` אני יכול להגדיר `interface` של אובייקט בקומפוננט הاب

- ניצור קבוע בשם `image` שיהיה שווה ערך לאובייקט עם מפתחות וערכים של תמונה
- נעביר את הקבוע שיצרנו למפתח באובייקט הפרויקט בשם `image`

בקומפוננט הבן

- נחלץ את המפתחות שאנו צריכים להם מאובייקט `PropTypes`
- ניצור קבוע בשם `imageType` שיהיה שווה ערך למפתח `shape` מtower אובייקט `PropTypes` שיקבל אובייקט קונFIGורציות עם המפתחות והערכים שאנו מעוניינים שלו ב `interface` של התמונה
- נגידר שהערך למפתח `image` יהיה הקבוע `imageType` שיצרנו

```
63  const image = { ←  
64    url: "https://cdn.pixabay.com/photo/2022/11/13/18/09/  
65    canyon-7589820_960_720.jpg",  
66    alt: "Rock",  
67  };  
68  const FatherPropTypes = () => {  
69    return <PropTypeComponent image={image} />;  
70  };  
  
73  import { shape, string } from "prop-types"; ←  
74  
75  const imageType = shape({ ←  
76    url: string,  
77    alt: string,  
78  });  
79  
80  const PropTypeComponent = props => {  
81    return <div>PropTypeComponent</div>;  
82  };  
83  
84  PropTypeComponent.propTypes = {  
85    image: imageType.isRequired,  
86  };
```

Any & defaultProps

בקומפוננט האב

- נציג שני קומפוננטות בנים בתוך קומפוננט האב
 - לראשונה נעביר מפתח בשם name עם ערך לאובייקט הפרווף
 - בשניה רק נציג את הקומפוננט מבלי להעביר לה מפתחות לאובייקט הפרווף

בקומפוננט הבן

- הקומפוננט צריכה לקבל את המפתח name באובייקט הפרווף ולהציגו אותו
- נגיד שערך המפתח name יכול להיות כל סוג של ערך בعزيزת any אבל שהוא ובה להעביר בו ערך כלשהו
- השתמש ב defaultProps כדי לקבוע ערכים דיפולטיבים למפתחות באובייקט הפרווף כך שאם לא יעבירו לנו ערך במפתח name נקבע שהערך הדיפולטיבי שלו יהיה "david"

התוצאה בדף

- בגלל שהערכנו את המפתח name לקומפוננט בן הראשון אנו מקבלים את הערך שהערכנו
- בgalל שלא הערכנו את מפתח name בקומפוננט בן השני מוצג הערך הדיפולטיבי שקבענו

! לא מומלץ להשתמש ב – any אלא להגדיר את סוג הערך המבוקש

```
74 const FatherPropTypes = () => {  
75   return (  
76     <>  
77     <PropTypeComponent name="shola" /> ←  
78     <br />  
79     <PropTypeComponent /> ←  
80   </>  
81 );  
82 };  
  
89 const PropTypeComponent = ({ name }) => { ←  
90   return name;  
91 };  
92  
93 PropTypeComponent.propTypes = {  
94   name: PropTypes.any.isRequired, ←  
95 };  
96  
97 PropTypeComponent.defaultProps = { ←  
98   name: "david",  
99 };
```

shola
david



node & children

בדיקה העברת אלמנט שניית להציג לגולש
וקומפוננטות ילדים



node & children

קומפוננט האב

- עבור לפתח באובייקט הפרויקט node מחרוזת תווים

- יש אפשרות להציב את הקומפוננט עם תגית פותחת ותגית סגרת ובתוכה להעיבר מחרוזת תווים או אפילו קומפוננטות שלמות

קומפוננט הבן

- הקומפוננט תחלץ מאובייקט הפרויקט את מפתח node ומזה שהעבכנו בין התגית הפותחת לתגית הסגרת של קומפוננט יהיה בתוך מפתח שהוא גם מילה שומרה בשם children

- אני מחייב מהקומפוננט את שני המפתחות שהילצתי כך שיוצגו לגולש

בעזרת PropTypes נודא ש:

- בפתח node הועבר לנו ערך שניינן להציגו לגולש
- שבפתח children מועברת לנו מחרוזת תווים

התוצאה בדף

```
84 const FatherPropTypes = () => {  
85   return <PropTypeComponent node="David">Yakin</PropTypeComponent>;  
86 };  
  
101 const PropTypeComponent = ({ node, children }) => {  
102   return `${node} ${children}`;  
103 };  
104  
105 PropTypeComponent.propTypes = {  
106   node: PropTypes.node.isRequired,  
107   children: PropTypes.string,  
108 };
```

David Yakin



PropTypes.element

בקומפוננט האב

- בין התיגיות הפותחת לtaggit הסוגרת של הקומפוננט אני יכול להעביר קומפוננט שלם ואף יותר אחד. בדוגמה שלහן אני מעביר את קומפוננט הבן בתוך קומפוננט הבן ולכל אחד מהם אני נותן כתוב ב מפתח node

בקומפוננט הבן

- אני מחלץ את המפתחות node children
- מדפיס בקונסול את children
- מוחזיר מהקומפוננט לגולש children
- קבוע כי הערך של המפתח children צריך להיות אלמנט של react על ידי שימוש במפתח element של PropTypes

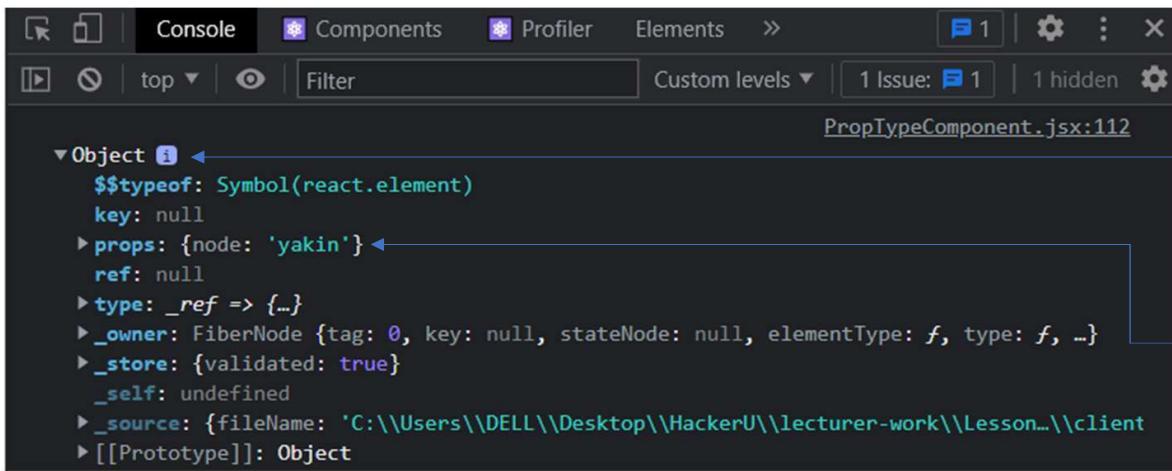
```
89 const FatherPropTypes = () => {
90   return (
91     <PropTypeComponent node="David">
92       <PropTypeComponent node="yakin" />
93     </PropTypeComponent>
94   );
95 };
96
97
98
99
100
101 const PropTypeComponent = ({ node, children }) => {
102   console.dir(children);
103   return (
104     <>
105       {node} {children}
106     </>
107   );
108 };
109
110 PropTypeComponent.propTypes = {
111   node: PropTypes.node.isRequired,
112   children: PropTypes.element,
113 };

114
115
116
117
118
119
120
121
122
123 }
```

התוצאה בדף

- ניתן לראות שהאלמנט שהערתתי לקומפוננט ב – children הוא מסוג אובייקט
- שיש לו מפתח בשם props שהערך שלו הוא אובייקט עם המפתח node והערך שהערתתי לקומפוננט הבן שבתוֹר קומפוננט הבן

David yakin



The screenshot shows the Chrome DevTools Elements tab with the title "PropTypeComponent.jsx:112". The DOM tree displays a single object node with the following properties:

- \$\$typeof: Symbol(react.element)
- key: null
- props: {node: 'yakin'}
- ref: null
- type: _ref => {}
- _owner: FiberNode {tag: 0, key: null, stateNode: null, elementType: f, type: f, ...}
- _store: {validated: true}
- _self: undefined
- _source: {fileName: 'C:\\\\Users\\\\DELL\\\\Desktop\\\\HackerU\\\\lecturer-work\\\\Lesson...\\\\client'}
- [[Prototype]]: Object

arrayOf(element)

בקומפוננט האב

- הפעם נعبر לkomponent הבן בפתחה –
children מספר אלמנטים

בקומפוננט הבן

- נחלץ את מפתח children
- נדפס אותו

- נחזיר אותו מהkomponent
- באמצעות PropTypes נבדוק שאכן hooverו לkomponent יותר מאלמנט אחד באמצעות השילוב של הפונקציה PropTypes.element שתקבל arrayOf

```
98  const FatherPropTypes = () => {
99    return (
100      <PropTypeComponent>
101        <div>first child</div>
102        <p>second child</p>
103      </PropTypeComponent>
104    );
105  };

126  const PropTypeComponent = ({ children }) => {
127    console.dir(children);
128    return children;
129  };
130
131 PropTypeComponent.propTypes = {
132   children: PropTypes.arrayOf(PropTypes.element),
133 };
```

התוצאות בדף

בגלל שהעberman יותר מאלמנט אחד React הכניסה את האלמנטים לתוך מערך שככל אלמנט הוא איבר במערך

first child
second child

```
first child
second child

Array(2) 1 ←
  ▾ 0:
    $typeof: Symbol/react.element
    key: null
    ▶ props: {children: 'first child'}
    ref: null
    type: "div"
    ▶ _owner: FiberNode {tag: 0, key: null, stateNode: null, elementType: f, type: f, ...}
    ▶ _store: {validated: true}
    _self: undefined
    ▶ _source: {fileName: 'C:\\\\Users\\\\DELL\\\\Desktop\\\\HackerU\\\\lecturer-work\\\\Lesson...\\\\cli...'}
    ▶ [[Prototype]]: Object
  ▾ 1:
    $typeof: Symbol/react.element, type: 'p', key: null, ref: null, props: {...}, ...
    length: 2
  ▶ [[Prototype]]: Array(0)
```

משימת חלק א' PropTypes



Business-cards-app

- צור את הנתיב `src/cards/models/types`
- צור שלושה קבצים בנאטיב שיצרת:
 - `cardType.js` – פירוט על קובץ זה בעמוד הבא
 - `imageType.js`
- צור קבוע בשם `imageType` שייהי שווה ערך לערך שיחזור ממטודת `PropTypes.shape` אליה תעביר את אובייקט הkonfigורציות עם הערכים הבאים:
 - `url` – מסוג מחוץ תווים שדה חובה
 - `at` – מסוג מחוץ תווים שדה חובה
- יצא את הקבוע שיצרת באמצעות `export default addressType.js`
- צור קבוע בשם `addressType` שייהי שווה ערך לערך שיחזור ממטודת `PropTypes.shape` אליה תעביר את אובייקט הkonfigורציות עם הערכים הבאים:
 - `state` – מסוג מחוץ תווים
 - `country` – מסוג מחוץ תווים ערך חובה
 - `city` – מסוג מחוץ תווים ערך חובה
 - `street` – מסוג מחוץ תווים ערך חובה
 - `houseNumber` – מסוג מספר ערך חובה
 - `zip` – מסוג מספר
- יצא את הקבוע שיצרת באמצעות `export default`

משימת חלק ב' PropTypes



Business-cards-app

- cardType
- ייבא את הקבועים addressType imageType שיצרת למודול
- צור קבוע בשם cardType שיהיה שווה ערך לערך שיחזור ממוגנתה Props.shape אליה תעביר את אובייקט הקונפיגורציות עם הערך הבא:

 - id – מסוג מחרוזת תווים שדה חובה
 - title – מסוג מחרוזת תווים שדה חובה
 - subtitle - מסוג מחרוזת תווים שדה חובה
 - description - מסוג מחרוזת תווים שדה חובה
 - address – מסוג addressType שדה חובה
 - image – מסוג imageType שדה חובה
 - bizNumber – מסוג מספר שדה חובה
 - phone - מסוג מחרוזת תווים שדה חובה
 - likes – מסוג מערך של מחרוזות תווים שדה חובה
 - web - מסוג מחרוזת תווים או undefined שדה חובה
 - email - מסוג מחרוזת תווים שדה חובה
 - user_id - מסוג מחרוזת תווים שדה חובה
 - createdAt - מסוג מחרוזת תווים שדה חובה

- יצא את הקבוע שיצרת באמצעות export default

משימת חלק ג' PropTypes



Business-cards-app

- בקומponent card השתמש בPropTypes ובמודלים שיצרת בשקפים הקודמים על מנת לוודא כי כרטיס ביקור מסווג של cardType מועבר לקומponent Card
- וודא שכל הקומponentות שמרכיבות את קומponent Card מקבלות גם הן את הפרופס הדרושים להם כדי לעבוד כהלכה בעזרת המודלים שיצרת בשקפים הקודמים

Shared Components

יצירת קומפוננטות המשותפות למספר דפים ומספר פיצ'רים





PageHeader

יצירת קומפוננט של כותרות וכותרות משנה
לדף באתר



```
PageHeader.jsx X  
client > src > components > PageHeader.jsx > ...  
1 import React from "react";  
2 import { string } from "prop-types";  
3 import Typography from "@mui/material/Typography";  
4 import Divider from "@mui/material/Divider";  
5  
6 const PageHeader = ({ title, subtitle }) => { ←  
7   return (  
8     <>  
9       <Typography variant="h2" component="h1"> ←  
10      | {title}  
11      </Typography>  
12      <Typography variant="h5" component="h2"> ←  
13      | {subtitle}  
14      </Typography>  
15      <Divider sx={{ my: 2 }} /> ←  
16    </>  
17  );  
18};  
19  
20 PageHeader.propTypes = {  
21   title: string.isRequired,  
22   subtitle: string.isRequired,  
23 };  
24  
25 export default PageHeader;
```

PageHeader.jsx

- ניצור את הנתיב :
src/component/PageHeader.jsx
- הקומפוננט קיבל את המפתחות title
subtitle באובייקט הפרוופ
- ניצור אלמנט h1 עם עיצוב של h2
- ניצור אלמנט h2 עם עיצוב של h5
- ניצור אלמנט hr בעזרת הקומפוננט Divider
- נודא שהקומפוננט מקבלת את המפתחות
הדרושים לה באובייקט הפרוופ באמצעות
propTypes

CardsPage.jsx

```
client > src > cards > pages > CardsPage.jsx > ...
1  import React from "react";
2  import Container from "@mui/material/Container";
3  import Cards from "../../components/Cards";
4  import PageHeader from "../../../components/PageHeader";
5
6  const CardsPage = () => {
7    >   const cards = [ ... ];
77
78
79  >   return (
80    <Container sx={{ mt: 2 }}>
81      <PageHeader
82        title="Cards"
83        subtitle="On this page you can find all business cards from all
84        categories"
85      />
86
87      <Cards cards={cards} />
88    </Container>
89  );
90
91  export default CardsPage;
```

CardsPage.jsx

נוצר את הנתיב:

src/cards/pages/CardsPage.jsx

- נעביר את הCARTEISIM לkomponenT הZAT
- נעטוף את תוכן הדף בMICL של MUI
- נציב את הקומפוננט שיצרנו PageHeader בטור דף המועד לתצוגת crtisim עם הכותרות המתאימות
- נציב את הקומפוננט Cards וنعביר אליו את המשתנה cards כמפתח באובייקט הפרופו

Cards

Here you can find business cards from all categories



first

Business Headline

Phone: 0500000000

Address: STREET 1 Tel Aviv

Card Number: 6480165



second

Business Headline

Phone: 0500000000

Address: STREET 1 Tel Aviv

Card Number: 6480165



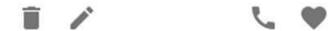
third

Business Headline

Phone: 0500000000

Address: STREET 1 Tel Aviv

Card Number: 6480165



התוצאה בדף

ניתן לראות כי בהתאם לתבנית שקבענו

- אנחנו רואים את כותרות הדף
- את החלק המועד לטקס המסביר על האפליקציה
- וראים את התמונה של הלקוח

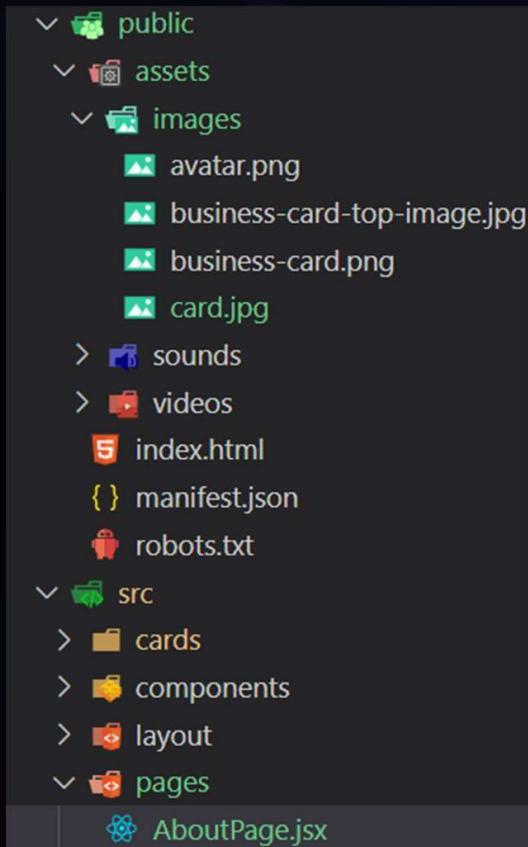
Static Folder

Using the Public Folder

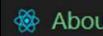
<https://create-react-app.dev/docs/using-the-public-folder/>



הכנות תשתית



- נסיף לנתיב `public/assets/images` תמונה של כרטיס ונקרא לה `card.jpg`
- בתוך תיקייה `src` ניצור תיקייה חדשה בשם `pages`
- בתוכה ניצור קובץ בשם `AboutPage.jsx`



AboutPage.jsx

```
client > src > pages > AboutPage.jsx > [default]
1 import React from "react";
2 import Container from "@mui/material/Container";
3 import PageHeader from "../../components/PageHeader";
4 import Grid from "@mui/material/Grid";
5
6 const AboutPage = () => {
7   return (
8     <Container maxWidth="lg"> ←
9       <PageHeader
10         title="About Page"
11         subtitle="On this page you can find explanations
12           about using the application"
13       />
14
15     <Grid container spacing={0}> ←
16       <Grid item xs={12} md={8} alignSelf="center"> ... ←
17         </Grid>
18         <Grid
19           item
20           xs={4}
21           sx={{
22             display: { md: "flex", xs: "none" },
23             justifyContent: "center",
24           }}>
25            ←
26         </Grid>
27       </Grid>
28     </Container>
29   );
30 }
31
32 export default AboutPage;
```

AboutPage.jsx

ניצור דף אודות שישתמש בתמונה שנשמר
בתיקייה public

- ניצור קומפוננט בשם AboutPage שתחזיר
- נשתמש בקומפוננט Container כדי לתת רווח
לתוכן בגדי מסך שונים
- אני מציב את הקומפוננט PageHeader עם
הכותרות המתאימות

• ניצור אזור רספונסיבי באמצעות הקומפוננט Grid

- נקבע כי החלק של המיל יתפוא את כל רוחב המסך
בגודל מסך קטן ויתפוא 8 עמודות מעל גודל מסך md
ונמרכז את התוכן לאמצע באמצעות "alignSelf="center"
- נקבע את החלק של התמונה כך שיוצג לגולש רק מוגדל
מסך md ונמרכז אותו
- מחרוזת התווים בערך של מאפיין src תתחיל בסימן
סלאש / כדי ש react תחליל את החיפוש של תיקיית
המבקשת

About Page

On this page you can find explanations about using the application

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aut ea quasi magnam rem velit cumque facilis minus iste, similique at placeat adipisci reiciendis! Quibusdant paratur voluptatibus suscipit, laboriosam earum sint asperiores, est velit voluptatem aspernatur quisquam modi quas, eligendi ad hic! Laborum deserunt quis, atque quam, sapiente maxime repellat voluptatem deleniti obcaecati aperiam ipsum! Iure, saepe! Voluptatibus harum, animi sapiente quas dolore, cum nam adipisci officiis inventore aperiam omnis aut fuga nemo perferendis tenetur? Debitis nihil facere quos? Debitis molestias quae voluptatum. Eius perferendis necessitatibus sed consequatur possimus ipsam odio, eos ab, enim corporis explicabo aspernatur consequuntur saepe quo facilis et voluptatem qui, ut quae! Reiciendis similique exercitationem ipsa. Aliquam quam eum ad, non delectus ducimus soluta numquam, molestiae fugiat sit odit! Repudiandae querat deserunt totam praesentium eaque voluptatem paratur neque porro, accusantium consequuntur, exercitationem quisquam? Itaque praesentium beatae consectetur, quisquam facilis qui laboriosam voluptate maxime cupiditate voluptas et nisi?



התוצאה בדף

- ניתן לראות כי בהתאם למבנה שקבענו
- אנחנו רואים את כותרות הדף
 - את החלק המועד לטקס המסביר על האפליקציה
 - וראים את התמונה של הלקוח

Hooks

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class.

<https://reactjs.org/docs/hooks-overview.html>



Introducing React Hooks

<https://www.youtube.com/watch?v=dpws4bM&t=2EHDh9>



Rules

Only Call Hooks at the Top Level

Don't call Hooks inside loops, conditions, or nested functions

Call Hooks from React function components

Call Hooks from custom Hooks

Call hooks from the lowest possible component in the component tree

A hook must start with the word "use" followed by the name

<https://reactjs.org/docs/hooks-rules.html>



useState

Hook שתפקידו הוא לנהל את האובייקט State



useState

בדוגמה שללן ניצור counter בעזרת hook

- נחלץ את מטודת useState מספרית react

- ניצור קומפוננט בשם SetCounter

- נעשה array destructor לערך שיחזור אליו מהפעלת מטודת useState עם הפרמטר 0 ונחלץ ממנו את המפתחות:

- counter – שם המשתנה

- setCounter – מטודה אחראית על שינוי ערכו של המשתנה

- נציב את הערך של המשתנה counter

- ניצור כפתור שבלחיצה עליו יפעיל פונקציה אונומית

- הֆונקציה האונומית תפעיל את מטודת setCounter שהילצנו

- הֆונקציה setCounter מקבלת פונקציה אונומית כאשר בפרמטר שלה היא מקבלת את ערכו של המשתנה counter והיא מעלה אותו באחד

- הֆונקציה setCounter מקבלת פונקציה אונומית כאשר בפרמטר שלה היא מקבלת את ערכו של המשתנה counter והיא מורידה אותו באחד

- הפעם אנחנו מעבירים לפונקציה setCounter את הערך שאנו מעוניינים שייהי למשתנה counter מבלי להעביר פונקציה אונומית כי אין אנחנו לא משנים את הערך על בסיס הערך הקודם אלא מAppending את הערך במספר אפס

```
SetCounter.jsx
src > components > SetCounter.jsx > ...
1 import { useState } from "react";
2
3 export const SetCounter = () => {
4   const [counter, setCounter] = useState(0);
5
6   return (
7     <div className="d-flex justify-content-center mt-2">
8       <div>
9         <div className="mx-2 text-center mb-2">
10           <Counter ${counter}>
11         </div>
12
13         <button
14           onClick={() => setCounter(counter => counter + 1)}
15           className="btn btn-outline-dark">
16           +
17         </button>
18
19         <button
20           onClick={() => setCounter(counter => counter - 1)}
21           className="btn btn-outline-dark mx-2">
22           -
23         </button>
24
25         <button
26           onClick={() => setCounter(0)}
27           className="btn btn-outline-dark">
28           reset counter
29         </button>
30       </div>
31     </div>
32   );
33 };
34 
```

useState with Function

בדוגמה שללן יוצרים פונקציה שבהתאם למה שנעביר לה היא תפעיל תשנה את ערכו של useState counter בעזרת counter

- ניצור קומפוננט בשם SetFunction
- נעשה array destructor לערך שיחזור אליו מהפעלת מטודת useState עם הפורמט `0` ונחלץ ממנו את המפתחות:
 - count – שם המשתנה
 - setCount – מטודה שאחראית על שינוי ערכו של המשתנה
- ניצור פונקציה בשם changeNum ש
 - קיבל בפורמט term
 - תתנה ואם הערך של term הוא מחרוזת התווים "increment" היא תבצע ותפעיל את מטודת setCount ותעלה את המספר באחד
 - תתנה ואם הערך של term הוא מחרוזת התווים "decrement" היא תבצע ותפעיל את מטודת setCount ותוריד את המספר באחד
 - ואם היא מקבלת משהו אחר היא תAppending את הערך של count
- הפעם שנלחץ על הכפתור נפעיל את הפונקציה שיצרנו עם הערך המתאים

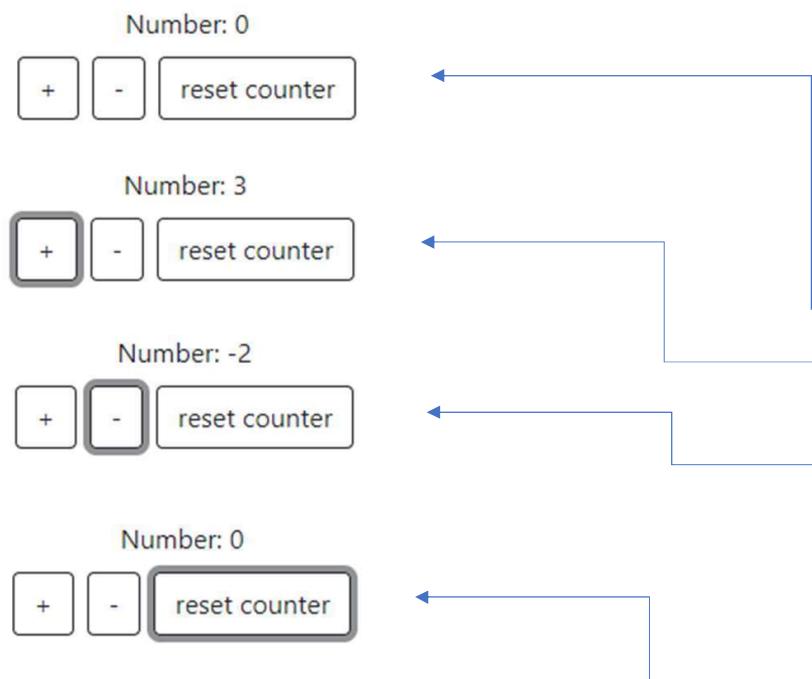
```
SetFunction.jsx

src > components > useState > SetFunction.jsx > ...
1 import { useState } from "react";
2
3 const SetFunction = () => {
4   const [count, setCount] = useState(0);
5
6   const changeNum = (term = "") => {
7     if (term === "increment") return setCount(count => count + 1);
8     if (term === "decrement") return setCount(count => count - 1);
9     setCount(0);
10  };
11
12  return (
13    <div className="d-flex justify-content-center mt-2">
14      <div>
15        <div className="mx-2 text-center mb-2">Number: {count}</div>
16
17        <button
18          onClick={() => changeNum("increment")}
19          className="btn btn-outline-dark">
20          +
21        </button>
22
23        <button
24          onClick={() => changeNum("decrement")}
25          className="btn btn-outline-dark mx-2">
26          -
27        </button>
28
29        <button onClick={changeNum} className="btn btn-outline-dark">
30          reset counter
31        </button>
32      </div>
33    </div>
34  );
35
36  export default SetFunction;
```

התוצאות בדף

ניתן לראות כי הערך של המשתנה count משתנה בהתאם ללחיצות על הכפתורים השונים

- בתחילת הוא מאפס
- כלחיצים על כפתור + הוא מעלה את המספר
- כלחיצים על כפתור - הוא מוריד את המספר
- כלחיצים על כפתור reset counter הוא מאפס את המספר



useState with objects

בדוגמה שלහן ניצור טופס שיבקש מהמשתמש את
השם הפרט**i** שלו ואת שם המשפחה בעזרת
`useState`

- נחלץ את מטודת `useState` מספרית `react`
- ניצור קומפוננט בשם `SetObject`
- ניצור קבוע בשם `initialName` שערכו יהיה אובייקט עם
המפתחות `first` `last`
- געזה array destructor למערך שיחזור אליו מהפעלת
מטודת `useState` עם הערך `initialName` ונחלץ
מןו את המפתחות:
 - `name` – שם המשתנה
 - `setName` – מטודה שאחרה על שינוי ערכו של
המשתנה
- נציב את ערכיו של המשתנה `name` כך שיוצגו לגולש
- ניצור אלמנטים מסוג `button` שכשיכניםו אליהם ערך ה-
תפוענה פונקציה אונומית ש
 - קיבל את האירוע
 - תפעיל את מטודת `setName` כאשר בפרמטר נעביר לה אובייקט
אליו אנו מעתיקים את כל המפתחות מאובייקט `name` ואז משנים
את הערך של המפתח הרלוונטי בהתאם לנ נתונים שהוכנו
לאלמנט `input`

```
SetObject.jsx x
src > components > useState > SetObject.jsx > ...
1 import { useState } from "react";
2
3 const SetObject = () => {
4   const initialValue = {
5     first: "",
6     last: ""
7   };
8
9   const [name, setName] = useState(initialValue);
10
11   return (
12     <div className="d-flex justify-content-center mt-4">
13       <form className="col-4 border p-2 rounded">
14         <h5>
15           Your Name Is:{" "}
16           <span className="fw-light">
17             {name.first} {name.last}
18           </span>
19         </h5>
20         <input
21           className="form-control mb-2"
22           placeholder="Enter First Name"
23           onChange={e => setName({ ...name, first: e.target.value })}
24         />
25         <input
26           className="form-control mb-2"
27           placeholder="Enter Last Name"
28           onChange={e => setName({ ...name, last: e.target.value })}
29         />
30       </form>
31     </div>
32   );
33 }
34
35 export default SetObject;
```

התוצאה בדף

ניתן לראות כי בהתאם לתבנית שקבענו מוצג לגולש

- את תפריט הניווט העליון
- לאחר מכן את תוכן הדף
- ולבסוף התפריט התחתון

Your Name Is: David Yakin

David

Yakin

useState with complex objects

בדוגמה שלහן ניצור טופס שיבקש מהמשתמש את השם הפרטី שלו ואת שם המשפחה בעזרת useState

- ניצור קומפוננט בשם SetComplexObject

- ניצור קבוע בשם INITIAL_USER שערך יהיה אובייקט עם המפתחות:

- name – מסוג אובייקט עם המפתחות first last שערכם הוא מחרוזות תווים

- email – מסוג מחרוזת תווים

- נעשה array destructor לערך שיחזור אלינו מהפעלת useState עם ה Parameter INITIAL_USER ונהלץ ממנו את המפתחות:

- user – שם המשתנה

- setUser – מетодה שאחראית על שינוי ערכו של המשתנה

- נציב את ערכיו של המשתנה user כך שיוצגו בגלוש ניצור אלמנט מסווג匿名 שכאשר יוכנסו אליו נתונים הוא יפעיל פונקציה אונומית שתקבל את האירוע ותפעיל את מетодת setUser ונעביר לה אובייקט ש:

- נועתק לתוכו את מפתחות האובייקט user

- נעשה השמה למפתח name ונשווה את ערכו לאובייקט שאליו נעתיק את המפתחות של האובייקט user.name כך

- ונעשה השמה למפתח first בתוך הערך של אובייקט name כך שערכו יהיה שווה לערך שיכנס לאלמנט input (e.target.value)

- ניצור אלמנט מסווג匿名 שיעתיק את המפתחות של user ויעשה שמה למפתח email

```
src > components > useState > SetComplexObject.jsx > ...
1  import { useState } from "react";
2
3  const SetComplexObject = () => {
4    const INITIAL_USER = {
5      name: {
6        first: "",
7        last: "",
8      },
9      email: ""
10    };
11
12  const [user, setUser] = useState(INITIAL_USER); ←
13
14  return (
15    <div className="d-flex justify-content-center mt-4">
16      <form className="col-4 border p-2 rounded">
17        <h5>
18          Your Name Is: {" "}
19          <span className="fw-light">
20            {user.name.first} {user.name.last}
21          </span>
22        </h5>
23        <h6>
24          Your email is:
25          <span className="fw-light"> {user.email}</span>
26        </h6>
27        <input
28          className="form-control mb-2"
29          placeholder="Enter First Name"
30          onChange={(e =>
31            setUser({ ...user, name: { ...user.name, first: e.target.value } })})←
32          }
33        />
34        <input
35          className="form-control mb-2"
36          placeholder="Enter Last Name"
37          onChange={(e =>
38            setUser({ ...user, name: { ...user.name, last: e.target.value } })})
39          }
40        />
41        <input
42          className="form-control mb-2"
43          placeholder="Enter Email"
44          onChange={(e => setUser({ ...user, email: e.target.value }))}←
45          />
46      </form>
47    </div>
48  );
49
50  export default SetComplexObject;
```

התוצאה בדף

ניתן לראות כי בהתאם לתבנית שקבענו מוצג לגולש

- את תפריט הניווט העליון
- לאחר מכן את תוכן הדף
- ולבסוף התפריט התחתון

Your Name Is: David Yakin

Your email is: david@gmail.com

David

Yakin

david@gmail.com

useState with array

הנתת התשתית

- ניבא את useState מספרית react
- ניצור קומפוננט בשם SetArray
- ניצור קבוע בשם INITIAL_TODO ונשווה את הערך שלו לאובייקט עם מפתחות וערכים ראשוניים
- פעיל את METHOD useState עם הקבוע שיצרנו ונחלץ ממנו את task setTask
- פעיל את METHOD useState עם מערך tasks setTasks
- ניצור פונקציה בשם createNewTask
- שתקבל אירוע
- תבצע את ההתנהגות הדיפולטיבית של שליחת הטופס על ידי הceptor
- שתפעיל את METHOD setTasks כشارוגומנט נתיק את מערך המשימות ונוסיף את המשימה
- נוצר את הפונקציה ונאפס את המשימה על ידי הפעלת METHOD setTask עם הקבוע של הערכים הראשוניים

```
SetArray.jsx M X
src > components > useState > SetArray.jsx > SetArray
1 import { useState } from "react";
2
3 export const SetArray = () => {
4   const INITIAL_TODO = { todo: "" };
5   const [task, setTask] = useState(INITIAL_TODO);
6   const [tasks, setTasks] = useState([]);
7
8   const createNewTask = e => {
9     e.preventDefault();
10    setTasks([...tasks, task]);
11    return setTask(INITIAL_TODO);
12  };
}
```

render

```
15 return [
16   <div className="d-flex justify-content-center mt-4">
17     <div>
18       <form className="col-12 border p-2 rounded">
19         <h5>
20           Task:
21           <span className="fw-light"> {task.todo}</span> ←
22         </h5>
23
24         <div className="input-group my-2">
25           <button
26             disabled={!task.todo} ←
27             onClick={createNewTask} ←
28             className="input-group-text"
29             id="inputGroup-sizing-default">
30             Create
31           </button>
32           <input
33             onChange={e => setTask({ ...task, todo: e.target.value })} ←
34             value={task.todo} ←
35             type="text"
36             className="form-control"
37             aria-label="Sizing example input"
38             aria-describedby="inputGroup-sizing-default"
39           />
40         </div>
41       </form>
42
43       <ul>
44         {tasks.map((todo, index) => (
45           <li key={index}>
46             {index + 1}. {todo.todo}
47           </li>
48         )));
49       </ul>
50     </div>
51   </div>
52 ];
53 };
```

- ניצור טופו
- בחלקן העליון נציג לגולש את המשימה
- ניצור כפטור ש:
- הוא יהיה מונטREL אם לא יהיה ערך במפתח todo של אובייקט task
- בלחיצה על הcpfטור הוא יפעיל את מטודת createNewTask
- ניצור input
- שתזין לאירוע onChange ותפעיל פונקציה אוניבריאט שתקבל את האירוע
- תפעיל את מטודת setTask כאשר בארגומנט נעתיק את המפתחות מתוך אובייקט task וונעשה השמה למפתח todo עם מה שיוצג בתוך האלמנט
- הערך של האלמנט יהיה הערך של המפתח task.todo
- ניצור רשיימה ש:
- על כל איבר במערך tasks היא תיצור רשומה עם מספר ועם פרטי המשימה

התוצאות בדפסן

ניתן לראות כי בהתאם למבנה שקבענו

- כשמלאים את הכתוב בשדה הטקסט מוצג בראש הטופס
- שלוחצים על הכפתור המשימה מוצגת כרשומה והשדה מתנתקה

Task: משימה שלישית

Create

משימה שלישית

- משימה ראשונה. 1.
- משימה שנייה. 2.
- משימה שלישית. 3.

Task:

Create

משימה שלישית

- משימה ראשונה. 1.
- משימה שנייה. 2.
- משימה שלישית. 3.

useState משימת



Hooks-Sandbox

• צור קומפוננט בשם SetPost

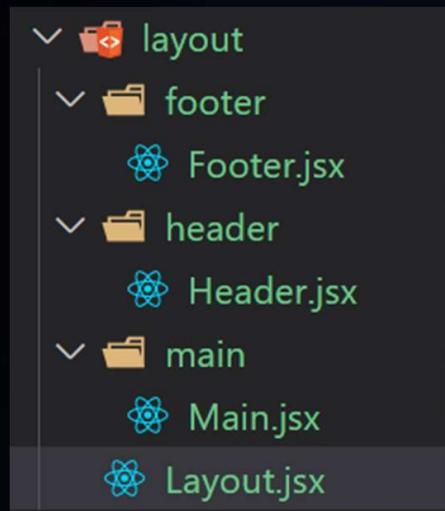
- ייבא את useState מモודול
- צור פונקציה בשם SetPost (קומפוננט)
- צור קבוע בשם POST_INITIAL שערך יהיה אובייקט עם המפתחות:
 - title – מסוג מחרוזת תווים
 - subtitle – מסוג מחרוזת תווים
 - author – מסוג מחרוזת תווים
 - createdAt – מחרוזת תווים
- הפעיל את מטודת useState פעמיים כאשר בפעם הראשונה עם הארגומנט INITIAL וחלץ מהערך שהוזר שוחזר את post setPost
- בפעם השנייה עם מערך ריק כארגומנט וחלץ מהערך שוחזר מהפעלת הפונקציה את posts setPosts
- הצג לגולש:
- טופס שבתוכו
 - צור שלוש אלמנטים מסוג `button` שהנתונים שיוצנו בתוכם יהיו מקושרים לערכים של שלושת המפתחות הראשונים של אובייקט post חדש (כמו שהוזג בשקפים הקודמים)
 - צור כפתור שלחיצה עליו תכניס פווט חדש למערך הפואטים
- טבלה שתציג רק אם יש פואטים במערך הפואטים

Layout

פריסת האפליקציה



הכנות תשתית



- בנתיב `src/layout` ניצור את התיקיות הבאות:
 - התיקייה `footer`
 - ובתוכה הקובץ `Footer.jsx`
 - התיקייה `header`
 - ובתוכה הקובץ `Header.jsx`
 - התיקייה `main`
 - ובתוכה הקובץ `Main.jsx`
 - הקובץ `Layout.jsx`

⚛️ Header.jsx ✎

client > src > layout > header > ⚛️ Header.jsx > ...

```
1 import React from "react";
2
3 const Header = () => {
4   return <div>Header</div>; ←
5 }
6
7 export default Header;
```

Header.jsx

בשלב זה ניצור את הקומפוננט `Header` שתשציג את התוכן שלה בתבנית הכללית

client > src > layout > footer > **Footer.jsx** > ...

```
1 import React from "react";
2
3 const Footer = () => {
4   return <div>Footer</div>;
5 }
6
7 export default Footer;
```

Footer.jsx

בשלב זה נוצר את הקומponent כרגע שציג את התוכן שלו בתבנית הכללית

Main.jsx

קומponent זה תהיה אחראית על תצוגת התוכן

- הקומponent קיבל בפתח של אובייקט ה -
props את המילה השמורה children
(כלומר אלמנט javascript/HTML/React)
data בין התגית הפתוחת לtagית הסגורה)
- התוכן הראשי יהיה עוטף באלמנט Box של
ועם שאני קבע גובה מינימלי וצבע רקע
- בນזודה זאת התוכן יוצג
- אני מודד בעזרת ספריית propTypes
שהקומponent אכן מקבלת אלמנט של React
בפתח children שבאובייקט הפורס

Main.jsx

```
client > src > layout > main > Main.jsx > ...
1  import { node } from "prop-types";
2  import Box from "@mui/material/Box";
3
4  const Main = ({ children }) => {
5    return (
6      <Box sx={{ minHeight: "85vh", backgroundColor: "#e3f2fd" }}>
7        {children}
8      </Box>
9    );
10 }
11
12 Main.propTypes = {
13   children: node.isRequired,
14 };
15
16 export default Main;
```

Layout.jsx

Layout.jsx

```
client > src > layout > Layout.jsx > ...
1  import React from "react";
2  import { node } from "prop-types";
3  import Header from "./header/Header";
4  import Main from "./main/Main";
5  import Footer from "./footer/Footer";
6
7  const Layout = ({ children }) => {
8    return (
9      <>
10      <Header />
11      <Main>{children}</Main>
12      <Footer />
13    </>
14  );
15};
16
17 Layout.propTypes = {
18   children: node.isRequired,
19 };
20
21 export default Layout;
```

בקומפוננט זה אסדר את התוכן כך שלכל דף ודף באתר יהיה Header Footer קבוע ורק התוכן של הדפים ישנה בהתאם לדף שהגולש נמצא בתוכו

- הקומפוננט קיבל בפתח של אובייקט הפוך את המילה השמורה children
 - עוטף את האלמנטים בקומפוננט באמצעות React.Fragment
 - ניציג גולש את הקומפוננטות:
 - Header – שתכלול תפריט ניוט בעתיד
 - Main – אליו נעביר את התוכן של הדף שברצוננו להציג
 - Footer – תכלול תפריט ניוט בעתיד לוגו וזכויות יוצרים
 - אני מודד בעזרה ספריית propTypes שהקומפוננט אכן מקבלת אלמנט של React.children שבאובייקט הפוך בפתח
- ! כרגע התוכן של הדף הוא עדין סטטי עד שנלמד בהמשך המציג להשתמש בניתובים

JS App.js M X

```
client > src > JS App.js > ...
1  import "./App.css";
2  import CardsPage from "./cards/pages/CardsPage";
3  import Layout from "./layout/Layout";
4
5  function App() {
6    return (
7      <div className="App">
8        <Layout>
9          <CardsPage /> ←
10         </Layout>
11       </div>
12     );
13   }
14
15  export default App;
```

App.js

- עטוף את הקומפוננט **CardsPage** כך שיוצג לנו תוכן הדף בקומפוננט **Layout** בלבד יחד עם **Header** ו**Footer** ב**CardsPage**

למעשה אנו מעבירים לקומפוננט **Layout** בפתח **children** שלו – אובייקט **props** אשר אלמנט של **React** בשם **CardsPage** אשר מוצב בתוך **Layout**.
בתוך הקומפוננט **Layout** מוצב הקומפוננט **main** שמקבל גם הוא את אלמנט **React** בפתח **children** שלו – **props** ומציג אותו!

התוצאה בדף

ניתן לראות כי בהתאם למבנה
שקבענו מוצג גולש

- את תפריט הנימוט העליון
- לאחר מכן את תוכן הדף
- ולבסוף התפריט התחתון

header ←

Cards

On this page you can find all business cards from all categories



first
subtitle

Phone: 050-0000000
Address: Shinkin 3 tel-aviv
Card Number: 1000000

subtitle

second

Phone: 050-0000000
Address: Shinkin 3 tel-aviv
Card Number: 2000000

third
subtitle

Phone: 050-0000000
Address: Shinkin 3 tel-aviv
Card Number: 3000000

trash edit heart trash edit heart

Footer ←

ErrorPage.jsx

**דף שגיאת 404 שיוצג במקרה והגיע לכתובת URL
שלא קיימת באפליקציה**

! יש לעבור למסך וUI לאחר navigation בטרם ממשיכים עם מצגת זאת



```
client > src > pages > ErrorPage.jsx > ErrorPage
1  import React from "react";
2  import Container from "@mui/material/Container";
3  import PageHeader from "../../components/PageHeader";
4  import Grid from "@mui/material/Grid";
5  import Typography from "@mui/material/Typography";
6  import Button from "@mui/material/Button";
7
8  const ErrorPage = () => {
9    return [
10      <Container>
11        <PageHeader title="Error 404" subtitle="Page not found" /> ←
12
13        <Grid container spacing={2}>
14          <Grid item xs={12} md={8}>
15            <Typography variant="h5" color="initial">
16              Oops... The requested URL was not found on this server ←
17            </Typography>
18            <Button variant="text" color="primary">
19              Click here to return to the home page... ←
20            </Button>
21          </Grid>
22          <Grid item xs={12} md={4} justifyContent="center">
23            
28          </Grid>
29        </Grid>
30      </Container>
31    ];
32  };
33
34  export default ErrorPage;
```

ErrorPage.jsx

ניצור את הנתיב jsx/ErrorPage.jsx

- ניצור את הקומפוננט ErrorPage
- הקומפוננט תציג לגולש כותרות מתאימות
- טקסט מתאים שיסביר לגולש שהדף שהוא ביקש לא נמצא
- כפתור שמאחור יותר יהיה לינק חזרה לדף הבית
- תמונה להמחשה שהגולש הגיע לדף 404

BCard ABOUT MY CARDS FAV CARDS

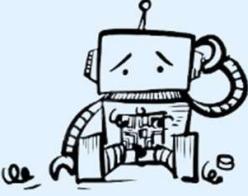
Search 

Error 404

Page not found

Oops... The requested URL was not found on this server

[CLICK HERE TO RETURN TO THE HOME PAGE...](#)



[!\[\]\(7fe18187a24413618d127488b9ffa806_img.jpg\) About](#) [!\[\]\(ccf9b6e58f510237340702e65ea248c1_img.jpg\) Favorites](#) [!\[\]\(7a6442fe2b26b9fabdaf3f58e68803fd_img.jpg\) My Cards](#)

התוצאה בדף

- ניתן לראות כי בהתאם לתבנית שקבענו
- אנחנו רואים את כוורות הדף
 - את החלק המועד לטקו המסביר על האפליקציה
 - וראים את התמונה של הלקוח

React Router Dom

ניתובים באפליקציה שהיא SPA

<https://reactrouter.com/en/main>





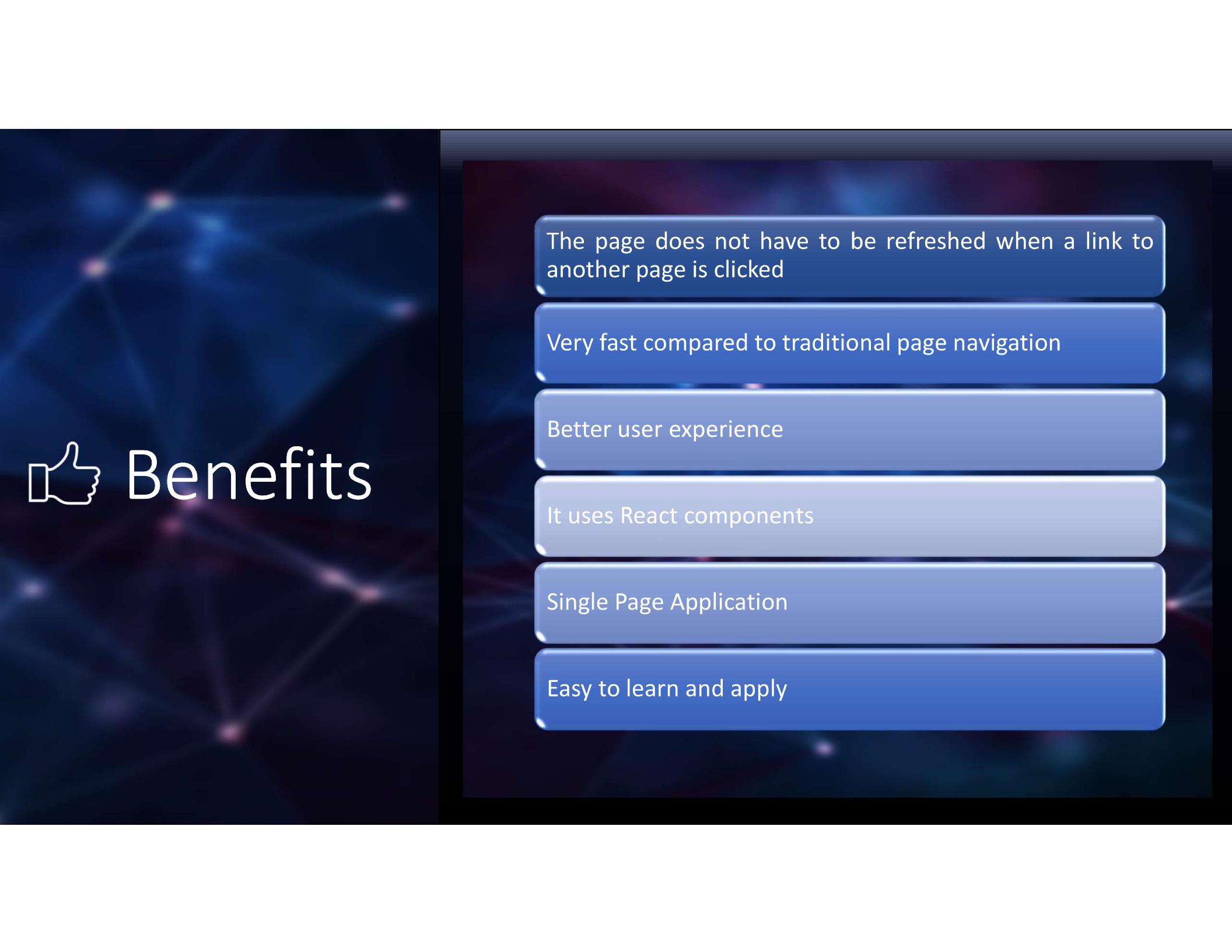
Q Definition

React Router DOM is an npm package that enables you to implement dynamic routing in a web app.

It allows you to display pages and allow users to navigate them.

It is a fully-featured client and server-side routing library for React.

React Router Dom is used to build single-page applications i.e. applications that have many pages or components but the page is never refreshed instead the content is dynamically fetched based on the URL.



👍 Benefits

The page does not have to be refreshed when a link to another page is clicked

Very fast compared to traditional page navigation

Better user experience

It uses React components

Single Page Application

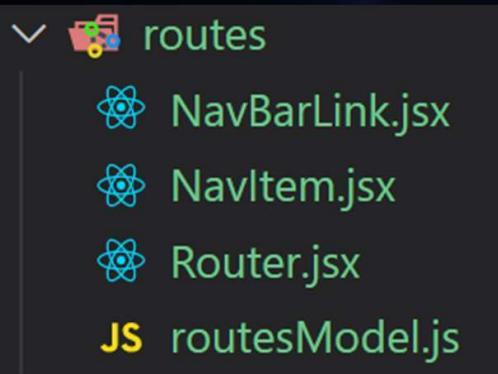
Easy to learn and apply



Installation

```
npm i react-router-dom
```

הכנות תשתית



- ניצור את הנתיב `src/routes` ובתוכו את הקבצים:

- `NavBarLink.jsx` – קומפוננט שيعוטף אלמנט כר שלחיצה עליו תעביר לכתובות ה – URL המוגדרת
- `NavItem.jsx` – קומפוננט שישמש כلينק בתפריט הניווט
- `Router.jsx` – הקובץ שינהל את התצוגה של הדפים
- `routesModel.js` – יכיל אובייקט שישמש כשפה משותפת לכל שמות הلينקים וערבים

routesModel.js

מודול זה ישמש כשפה משותפת לכל
שמות הLINKS וערוצים

- ניצור קבוע בשם ROUTES שערך יהיה
אובייקט שהמפתחות שלו יהיו הדפים
אליהם נרצה להגעה והערכים הם
הנתיב שנעביר לקומponent שיבקש
URL

- ניצא את הקבוע שיצרנו מהמודול

```
JS routeModel.js U X
client > src > routes > JS routeModel.js > ...
1 const ROUTES = {
2   ROOT: "/",
3   ABOUT: "/about",
4   CARDS: "/cards",
5 };
6
7 export default ROUTES;
```



Routes

קומפוננט של react-router-dom ש אחראי על
ניתובים



Router.jsx

קובץ זה ינהל את תצוגת הדפים לגולש

- נייבא את הקומפוננטות react-router-dom Route

- ניצור קומפוננט בשם Router

- נעתוף את קומפוננטות ה – Routes

- כל קומפוננט מסוג Route מקבל בקלט Routes

- בשלב זה שבי מאפיינים:

- path – כתובת URL

- element – הקומפוננט שנרצה להציג במידה ומגיעים לכתובת ה – URL של הקומפוננט

- הקומפוננט האחרון שנציג מקבל במאפיין path את הכתובת "*" וכך היא תירט את כל הכתובות שלא נתפסו בניתוביים הקודמים ותציג Komponent ErrorPage

! יש חשיבות לסדר הקומפוננטות של Route

Router.jsx

```
client > src > routes > Router.jsx > ...
1  import React from "react";
2  import { Route, Routes } from "react-router-dom"; ←
3  import CardsPage from "../../cards/pages/CardsPage";
4  import AboutPage from "../../pages/AboutPage";
5  import ErrorPage from "../../pages/ErrorPage";
6  import Sandbox from "../../sandbox/Sandbox";
7  import ROUTES from "./routesModel";
8
9  const Router = () => {
10    return (
11      <Routes> ←
12        <Route path={ROUTES.CARDS} element={<CardsPage />} />
13        <Route path={ROUTES.ABOUT} element={<AboutPage />} />
14        <Route path="/sandbox" element={<Sandbox />} />
15        <Route path="*" element={<ErrorPage />} /> ←
16      </Routes>
17    );
18  };
19
20  export default Router;
```



BrowserRouter

קומפוננט של react-router-dom שתנתב לדף
המתאים לפי הלוגיקה הרשומה בקומפוננט
Routes

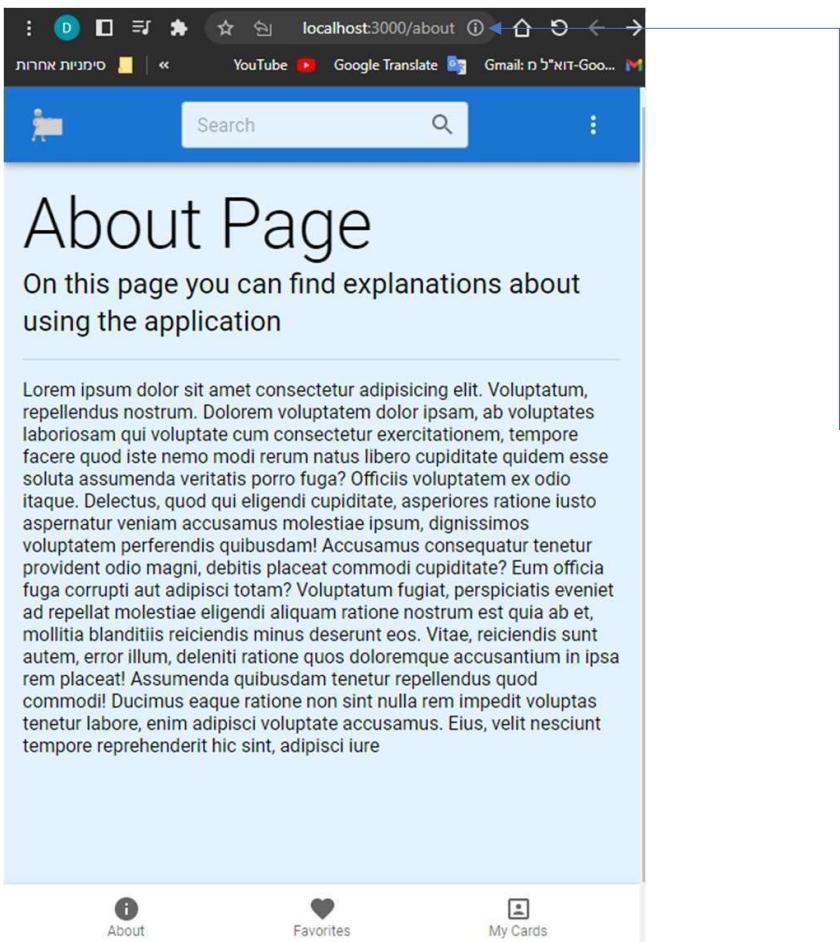


JS App.js

```
client > src > JS App.js > ...
1  import "./App.css";
2  import Layout from "./layout/Layout";
3  import { BrowserRouter } from "react-router-dom"; ←
4
5  import Router from "./routes/Router";
6  function App() {
7    return (
8      <div className="App">
9        <BrowserRouter> ←
10       <Layout>
11         <Router />
12       </Layout>
13     </BrowserRouter>
14   </div>
15 );
16
17
18 export default App;
```

BrowserRouter

- ניבא את BrowserRouter מספרית react-router-dom
- געטוף את האפליקציה שלנו בקומפוננט BrowserRouter



התוצאה בדף

אם נ קיש בשורת הכתובות לאחר כתובות
ה DNS והפורט את כתובת ה URL של
/about נ עבור לדף אודזות



Link & NavLink

קומפוננט של `react-router-dom` שתקבע את כתובת URL לפי הפרמטרים שנזין לתוכה



NavBarLink.jsx

NavBarLink.jsx

```
client > src > routes > NavBarLink.jsx > ...
1  import React from "react";
2  import { Link } from "react-router-dom"; ←
3  import { node, string } from "prop-types";
4
5  const NavBarLink = ({ to, color, children }) => {
6    return (
7      <Link to={to} style={{ color, textDecorationLine: "none" }}>
8        {children}
9      </Link>
10     );
11   };
12
13  NavBarLink.propTypes = {
14    children: node.isRequired,
15    to: string.isRequired,
16    color: string.isRequired,
17  };
18
19  NavBarLink.defaultProps = {
20    color: "#fff", ←
21  };
22
23  export default NavBarLink;
```

- קומponent שיעוטף אלמנט כר שלחיצה עליו תעביר לכתובת ה – URL המוגדרת
- NEYBA AT Link מספרית- dom
- NEYZOR KOMPONENT NAVBarLink בשם NAVBarLink בשם NAVBarLink
- NEYTKEV BAOBIKET PROPS AT HEMFOTOT:
 - URL – to
 - color – הצבע הכתוב של האלמנט שנעטוף
 - children – אלמנט שהקומponent תעטוף
- הקומponent תחזיר את הקומponent Link שיבאנו כאשר מאפיין to יהיה שווה ערך למפתח to אווביקט הProps, ונקבע כי המאפיין style יקבל את הצבע שנעביר לאובייקט הProps ונקבע שלא יהיה קו תחתון לאלמנט שאנו עוטפים
- נשתמש במספרית PropTypes כדי לוודא שהקומponent מקבל את כל מה שהוא צריך
- נקבע את הצבע הלבן כערך הדיפולטי של צבע הטקסט

```
client > src > layout > header > TopNavBar > Logo > Logo.jsx > ...  
1 import React from "react";  
2 import Typography from "@mui/material/Typography";  
3 import NavBarLink from "../../../../../routes/NavBarLink";  
4 import ROUTES from "../../../../../routes/routesModel";  
5  
6 const Logo = () => {  
7   return (  
8     <NavBarLink to={ROUTES.CARDS}>  
9       <Typography  
10         variant="h4"  
11         sx={{  
12           display: { xs: "none", md: "inline-flex" },  
13           marginRight: 2,  
14           fontFamily: "fantasy",  
15         }}>  
16       BCard  
17       </Typography>  
18     </NavBarLink>  
19   );  
20 };  
21  
22 export default Logo;
```

Logo.jsx

קומponeנט שתשמש בקומponeנט NavBarLink שיצרנו

- ניבא את NavBarLink
- ניבא את ROUTES מתחום המודול שיצרנו ל URL
- ניצור קומponeנט בשם Logo שתציג

• געטוף את הקומponeנט NavBarLink שיצרנו ונעביר לו בפרמטר את כתובת ה – URL המתאימה מתחור אובייקט ROUTES

• נקבע עיצוב מיוחד ללוגו

NavItem.jsx

קומפוננט המועדת לשימוש בתפריט ניוט

- ניבא את הקומפוננט NavBarLink
- נוצר את הפונקציה NavItem
- הפונקציה תחלץ מאובייקט הפרויקט את המפתחות:

- label – מסוג מחוזת תוים
- to – מחוזת תוים של URL
- color – מחוזת תוים

- נעוטף את קומפוננט הcptor בקומפוננט NavBarLink שתקבל מאפיינים שלה את כתובת ה – URL אליה היא צריכה להעביר את הgelsh ואת הצבע של הכיתוב על הcptor
- נודא שהקומפוננט מקבלת את כל המאפיינים שהיא צריכה כדי שהלוגיקה שלה תפעל כראוי בעזרת PropType

```
NavItem.jsx U X  
client > src > routes > NavItem.jsx > ...  
1 import React from "react";  
2 import { string } from "prop-types";  
3 import Typography from "@mui/material/Typography";  
4 import Button from "@mui/material/Button";  
5 import NavBarLink from "./NavBarLink";  
6  
7 const NavItem = ({ label, to, color }) => {  
8   return (  
9     <NavBarLink to={to} color={color}>  
10      <Button color="inherit">  
11        <Typography>{label}</Typography>  
12      </Button>  
13    </NavBarLink>  
14  );  
15};  
16  
17 NavItem.propTypes = {  
18   label: string.isRequired,  
19   to: string.isRequired,  
20   color: string,  
21 };  
22  
23 export default NavItem;
```

LeftNavigation.jsx

שימוש בkomponent NavItem שיצרנו

- ניבא את NavItem
- ניבא את ROUTES
- נציב את הקומponent במקום הרצוי
ונעביר לה את המאפיינים שהיא
צריכה

```
client > src > layout > header > TopNavBar > LeftNavigation.jsx > ...
1  import React from "react";
2  import Box from "@mui/material/Box";
3  import Logo from "./Logo/Logo";
4  import LogoIcon from "./Logo/LogoIcon";
5  import NavItem from "./NavItem";
6  import ROUTES from "../../routes/routesModel";
7
8  export const LeftNavigation = () => {
9    return (
10      <Box>
11        <LogoIcon />
12
13        <Logo />
14
15        <Box sx={{ display: { xs: "none", md: "inline-flex" } }}>
16          <NavItem label="About" to={ROUTES.ABOUT} />
17          <NavItem label="My Cards" to={ROUTES.MY_CARDS} />
18          <NavItem label="Fav Cards" to={ROUTES.FAV_CARDS} />
19        </Box>
20      </Box>
21    );
22  };

```

BCard ABOUT MY CARDS FAV CARDS

Search 

Cards

Here you can find business cards from all categories



title	subtitle
Phone: 050-0000000 Address: Shinkin 3 tel-aviv Card Number: 1000000	Phone: 050-0000000 Address: Shinkin 3 tel-aviv Card Number: 2000000
   	   

third	forth
Phone: 050-0000000 Address: Shinkin 3 tel-aviv Card Number: 3000000	Phone: 050-0000000 Address: Shinkin 3 tel-aviv Card Number: 4000000
   	   

 About  Favorites  My Cards

התווצה בדף

عصיו הלינקים שלנו בתפריט הניווט
עובדים, לחיצה עליהם תעביר את הגולש
לדף המבוקש

משימת react-router-dom



Business-cards-app

- שנה את תפריט הניווט כך שלחיצה על קישור תנסה את כתובות URL כדלהלן:

URL	ROUTES	label	lienק / אונ'
/signup	SINGUP	SINGUP	.1
/login	LOGIN	LOGIN	2.
/user-info	USER_PROFILE	profile	3.
/edit-user	EDIT_USER	Edit account	4.
/about	ABOUT	About	.5
/my-cards	MY_CARDS	MY CARDS	.6
/fav-cards	FAV_CARDS	FAV CARDS	.7
/sandbox	SANDBOX	SANDBOX	8.
/cards	CARDS	Logo	9.
/cards	CARDS	logolcon	10.



useNavigate

Hook של react-router-dom שעזר לנו בניתוב



useNavigate

בעזרת Hook זה נוכל לשמר על העיצוב של MUI
ולהשתמש בניתוב באמצעות react-router-dom

- ניבא את ROUTES למודול react-router-dom מ -
- נחלץ את useNavigate מ -
- ניצור קבוע בשם navigate שערך יהיה הערך שייחסור מהפעלת Methodת useNavigate
- ניצור קבוע בשם navigateTo
- שהערך שלו יהיה פונקציה שמקבלת בפרמטר to מסוג של מחוזת תווים
- ותפעיל את navigate עם מחוזת התווים שהועברה לפונקציה בפרמטר to
- בкомпонент שלחיצה אליה נרצה להעביר את הגולש לדף אחר נזין לאיירע onClick שיפעל פונקציה אונומית שתפעיל את פונקציית navigateTo שיצרנו עם כתובות ה – URL שתנתן את הגולש לדף הרלוונטי

```
client > src > layout > footer > Footer.jsx > Footer > navigate

1 import React from "react";
2 import BottomNavigation from "@mui/material/BottomNavigation";
3 import BottomNavigationAction from "@mui/material/BottomNavigationAction";
4 import FavoriteIcon from "@mui/icons-material/Favorite";
5 import Paper from "@mui/material/Paper";
6 import InfoIcon from "@mui/icons-material/Info";
7 import PortraitIcon from "@mui/icons-material/Portrait";
8 import ROUTES from "../../routes/routesModel"; ←
9 import { useNavigate } from "react-router-dom"; ←

10
11 const Footer = () => {
12   const navigate = useNavigate(); ←
13   const navigateTo = to => navigate(to); ←

14
15   return (
16     <Paper
17       sx={{ position: "sticky", bottom: 0, left: 0, right: 0 }}
18       elevation={3}>
19       <BottomNavigation showLabels>
20         <BottomNavigationAction
21           label="About"
22           icon={}
23           onClick={() => navigateTo(ROUTES.ABOUT)} />
24         <BottomNavigationAction
25           label="Favorites"
26           icon={}
27           onClick={() => navigateTo(ROUTES.FAV_CARDS)} />
28         <BottomNavigationAction
29           label="My Cards"
30           icon={}
31           onClick={() => navigateTo(ROUTES.MY_CARDS)} />
32       </BottomNavigation>
33     </Paper>
34   );
35
36   export default Footer;
37 };
38 
```



Navigate

קומponeנט של react-router-dom שתעזר לנו
בניתוב הגולש לדף אחר במידה והוא לא יעמוד
בתנאי שנקבע



SignupPage.jsx

```
client > src > users > pages > SignupPage.jsx > ...
1  import React from "react";
2  import { Navigate } from "react-router-dom"; ←
3  import ROUTES from "../../routes/routesModel"; ←
4  import Container from "@mui/material/Container";
5  import PageHeader from "../../components/PageHeader";
6
7  const SignupPage = () => {
8    const user = null; ←
9    //   const user = true;
10
11  if (user) return <Navigate replace to={ROUTES.CARDS} />; ←
12
13  return (
14    <Container maxWidth="lg"> ←
15      <PageHeader
16        title="Signup Page"
17        subtitle="In order to register, fill out the form
18        and click the submit button"
19      />
20      </Container>
21  );
22}
23 export default SignupPage;
```

- ניצור את הנתיב src/users/pages/SignupPage.jsx
- ניבא את Navigate מותוך react-router-dom
- ניבא את ROUTES
- ניצור משתנה בשם user ונשווה את הערך שלו פעם ל - null ופעם ל - true
- ניצור התנינה ואם הערך של user הוא לא פולסיבי נחזיר מהkomponenט את הקומוננט Navigate של react-router-dom ובעזרת המאפיין URL.replace נחליף את כתובת URL to לכתובת הרשמה במאפיין export default
- אם לא נכנס לתנינה נחזיר מהkomponent את דף ההתחברות

Router.jsx X

```
client > src > routes > Router.jsx > ...
1  import React from "react";
2  import { Route, Routes } from "react-router-dom";
3  import CardsPage from "../../cards/pages/CardsPage";
4  import AboutPage from "../../pages/AboutPage";
5  import ErrorPage from "../../pages/ErrorPage";
6  import Sandbox from "../../sandbox/Sandbox";
7  import ROUTES from "./routesModel";
8  import SignupPage from "../../users/pages/SignupPage";
9
10 const Router = () => {
11   return (
12     <Routes>
13       <Route path={ROUTES.CARDS} element={<CardsPage />} />
14       <Route path={ROUTES.ABOUT} element={<AboutPage />} />
15       <Route path={ROUTES.SIGNUP} element={<SignupPage />} /><-->
16       <Route path="/sandbox" element={<Sandbox />} />
17       <Route path="*" element={<ErrorPage />} />
18     </Routes>
19   );
20 };
21
22 export default Router;
```

Router.jsx

- ניבא את הקומponent SignupPage
- נוצר Route חדש שיעביר אותו לקומponent SignupPage במידה וshortcut הכתובות משתנה לכטובה אשר בפתח ROUTES.SIGNUP

מיפוי
Navigate



Business-cards-app

- צור את הkomponent LoginPage.jsx בנתיב
`src/users/pages`
- התנה ואם המשתמש מחובר העבר את הגולש
לדף הcartisim



useParams

Hook של react-router-dom שמחזיר את
האובייקט params משורת הכתובות



CardDetailsPage.jsx

ניצור את הנתיב
src/cards/pages/CardDetailsPage.jsx

- נייבא את useParams מספריית react-router-dom

נחלץ מהאובייקט שיחזור מהפעלת המטודה useParams את מפתח id

- נציג לגולש את תעודת הזהות של הלקוח שאות פרטיו נציג בקומponent

```
client > src > cards > pages > CardDetailsPage.jsx > ...
1  import { useParams } from "react-router-dom"; ←
2  import Container from "@mui/material/Container";
3  import PageHeader from "../../components/PageHeader";
4
5
6  const CardDetailsPage = () => {
7    const { id } = useParams(); ←
8
9    return (
10      <Container maxWidth="lg">
11        <PageHeader
12          title="Business Details"
13          subtitle="Here you can find more details about the business"
14        />
15        <div>Details of card: {id}</div> ←
16      </Container>
17    );
18  };
19
20  export default CardDetailsPage;
```

JS routesModel.js M X

```
client > src > routes > JS routesModel.js > ...
1  const ROUTES = {
2    ROOT: "/",
3    ABOUT: "/about",
4    CARDS: "/cards",
5    CARD_INFO: "/card-info", ←
6    MY_CARDS: "/my-cards",
7    FAV_CARDS: "/fav-cards",
8    SIGNUP: "signup",
9    LOGIN: "login",
10   USER_PROFILE: "/user-info",
11   EDIT_USER: "/edit-user",
12 };
13
14 export default ROUTES;
```

routesModel.js

- כוֹסֵף את המפתח CARD_INFO

Router.jsx

- ניצור **Route** חדש שיעביר אותנו לkomponent CardDetailsPage במידה ושורת הכתובות משתנה לנكتוב ROUTES.CARD_INFO אשר ב מפתח ROUTES.CARD_INFO/:id ונצפה לקבל בשורת הכתובות גם מפתח באובייקט params בשם id

```
client > src > routes > Router.jsx > ...
1  import React from "react";
2  import { Route, Routes } from "react-router-dom";
3  import CardsPage from "../../cards/pages/CardsPage";
4  import AboutPage from "../../pages/AboutPage";
5  import ErrorPage from "../../pages/ErrorPage";
6  import Sandbox from "../../sandbox/Sandbox";
7  import ROUTES from "./routesModel";
8  import SignupPage from "../../users/pages/SignupPage";
9  import CardDetailsPage from "../../cards/pages/CardDetailsPage";
10
11 const Router = () => {
12   return (
13     <Routes>
14       <Route path={ROUTES.ABOUT} element={<AboutPage />} />
15       <Route path={ROUTES.CARDS} element={<CardsPage />} />
16       <Route path={`${ROUTES.CARD_INFO}/:id`} element={<CardDetailsPage />} />
17       <Route path={ROUTES.SIGNUP} element={<SignupPage />} />
18       <Route path="/sandbox" element={<Sandbox />} />
19       <Route path="/" element={<ErrorPage />} />
20     </Routes>
21   );
22 };
23
24 export default Router;
```

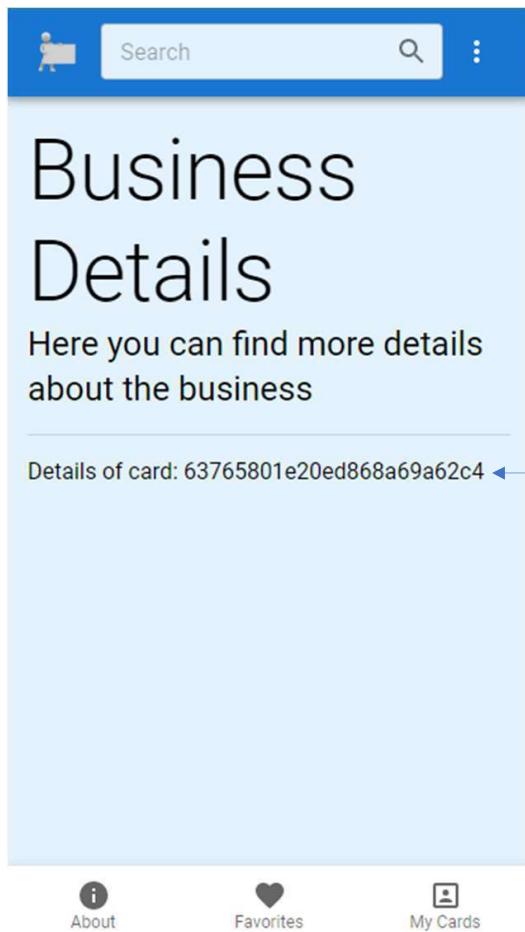
Card.jsx

בקומפוננט זה ניצור את הלוגיקהacr
שבלחיצה על כרטיס עסקי ספציפי נועבר
לדף עם הפרטים על הלקוח

- ניצור קבוע בשם navigate שערכו
יהה הערך שיחזור מהפעלת מетодת
useNavigate

- נאזרן ללחיצה על איזור זה בכרטיס
נפעיל את מетодת navigate כאשר
בפרמטר נעביר לה את כתובות URL
בנוסף את תעודת הזהות של הלקוח

```
12 const CardComponent = ({ card, handleCardDelete }) => {
13   const navigate = useNavigate(); ←
14
15   return (
16     <Card sx={{ minWidth: 280 }}>
17       <CardActionArea
18         onClick={() => navigate(`${
19           ROUTES.CARD_INFO}/ ${card._id}`)}>
20           <CardHead image={card.image} />
21           <CardBody card={card} />
22         </CardActionArea>
23         <CardActionBar
24           handleCardDelete={handleCardDelete}
25           bizNumber={card.bizNumber}
26         />
27       </Card>
28   );
}
```



התוצאה בדף

לחיצה על כרטיס הובילו אותנו לדף CardDetails.js

ומוצג לנו תעודת הזהות של הלקוח בתוכה אותה הקומponentן קacha מאובייקט ה `params` באמצעות המטודה `useParams`



Nested Routes

הדרך לייצר תפריט ניוט בתוך קומפוננט



```
client > src > routes > Router.jsx > Router
1 > import React from "react"; ...
20
21 const Router = () => {
22   return (
23     <Routes>
24       <Route path={ROUTES.ROOT} element={<CardsPage />} />
25       <Route path={ROUTES.ABOUT} element={<AboutPage />} />
26       <Route path={ROUTES.CARDS} element={<CardsPage />} />
27       <Route path={`${ROUTES.CARD_INFO}/:id`} element={<CardDetailsPage />} />
28       <Route path={ROUTES.SIGNUP} element={<SignupPage />} />
29       <Route path="/sandbox" element={<Sandbox />}> ←
30         <Route path="fetch" element={<DataFetch />} />
31         <Route path="custom-hook" element={<Counter />} />
32         <Route path="propTypes" element={<FatherPropTypes />} />
33         <Route path="props" element={<FatherComp />} />
34         <Route path="lifecycle" element={<LifeCycleHooks />} />
35         <Route path="use-callback" element={<UseCallBackComp />} />
36         <Route path="loops" element={<Loops />} />
37         <Route path="events" element={<OnClick />} /> ←
38         <Route path="use-memo" element={<UseMemo />} />
39         <Route path="axios" element={<AxiosComp />} />
40       </Route>
41       <Route path="*" element={<ErrorPage />} />
42     </Routes>
43   );
44 };
45
46 export default Router;
```

Router.jsx

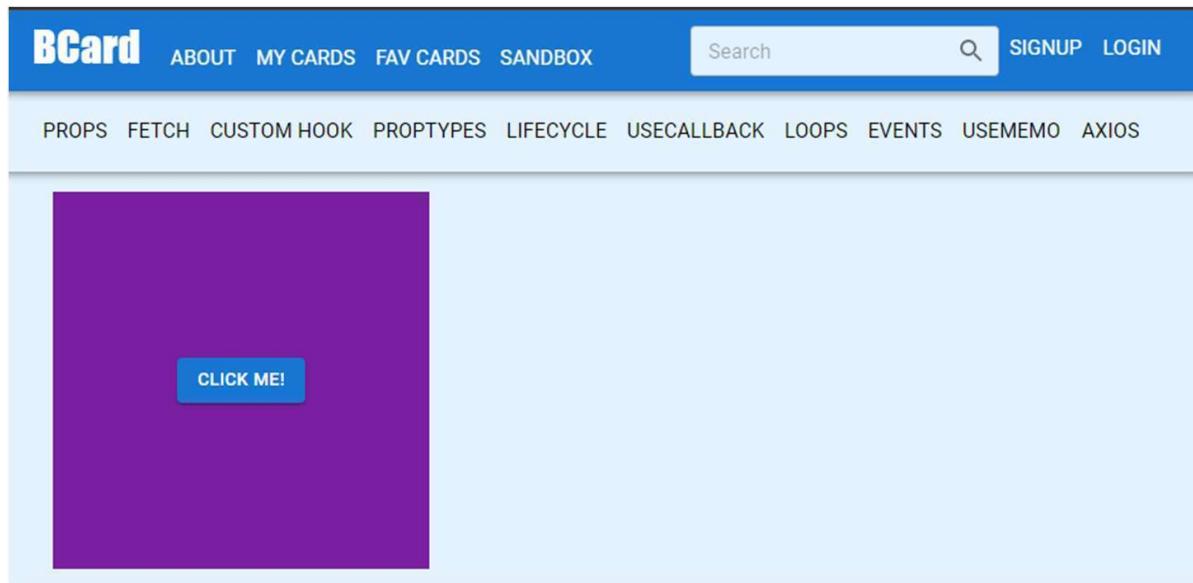
React-router-dom מותנת לנו דרך קלה ונוחה ליצור nested Route . ישן שלושה שלבים כאשר בשלב הראשון הוא עטיפת Route בקומפוננט Route הראשית. בדוגמה של להלן:

- יצירת תגית סגורה לקומפוננט Route שמאפיין ה – path שלו שווה ל – Sandbox ובקומפוננט שהוא מציג הוא Sandbox
- בקומפוננטות הבנים אין לי צורך לרשום את הכתובת המלאה אלא ארשום במאפיין path את הכתובת הירטטיבית שלהם (ללא סלאש בתחילת כתובות ה - url)

Sandbox.jsx

- בשלב שני ניבא את קומפוננט Outlet מתוכן react-router-dom וציב אותה במקום בו אנו רצים להציג את התוכן
- בשלב שלישי נוצר תפריט ניהול שינוי את הגולש לכתובות הרלוונטיות

```
client > src > sandbox > Sandbox.jsx > ...
1 import AppBar from "@mui/material/AppBar";
2 import Toolbar from "@mui/material/Toolbar";
3 import NavItem from "../../routes/NavItem";
4 import { Outlet } from "react-router-dom"; ←
5 import Container from "@mui/material/Container";
6
7 const Sandbox = () => {
8   return (
9     <>
10       <AppBar position="static" color="transparent">
11         <Toolbar>
12           <NavItem label="props" to="props" color="black" />
13           <NavItem label="fetch" to="fetch" color="black" />
14           <NavItem label="custom hook" to="custom-hook" color="black" />
15           <NavItem label="propTypes" to="propTypes" color="black" />
16           <NavItem label="lifecycle" to="lifecycle" color="black" />
17           <NavItem label="usecallback" to="use-callback" color="black" /> ←
18           <NavItem label="loops" to="loops" color="black" />
19           <NavItem label="events" to="events" color="black" />
20           <NavItem label="usememo" to="use-memo" color="black" />
21           <NavItem label="axios" to="axios" color="black" />
22         </Toolbar>
23       </AppBar>
24
25       <Container maxWidth="lg">
26         <Outlet /> ←
27       </Container>
28     </>
29   );
30 }
31
32 export default Sandbox;
```



התוצאה בדף

לחיצה קישור תוביל לkomponen שתציג את תוכנו וכל עוד אנו נשאים בקישור הראשי של sandbox תפריט הניות של הקומפוננט ישאר

משימת Nested Routs



Business-cards-app

- צורך טפריט ניוט בקומפוננט Sandbox כך של כל תיקייה בתוך תיקייה sandbox יש ליצור טפריט ניוט משלה עם התצוגה המתאימה לה כפי ש�示 בדוגמה מצד שמאל

Life Cycle Hooks

האזנה לאירועים של יצירת עדכון וחיסול קומפוננט והפעלת
לוגיקה מוגדרת בהתאם

<https://reactrouter.com/en/main>



Lifecycle Hooks

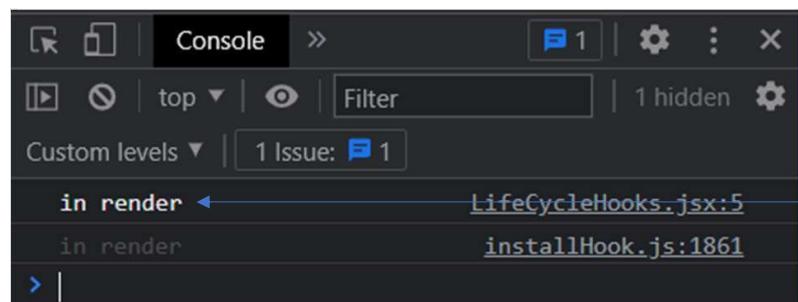


! כהמשתמשים ב React Classes יש עוד נקודות במחזור החיים של הקומפונט שנitin להאזין להם ולהפעיל מטודות !

! אומרים שנקודות אילו יהיו בעtid גם ב – react hooks על כן בקיישור הבא:

<https://reactjs.org/docs/hooks-faq.html#how-do-lifecycle-methods-correspond-to-hooks>

```
client > src > sandbox > LifeCycleHooks.jsx > ...
1  import React from "react";
2
3  const LifeCycleHooks = () => {
4      return (
5          <div>
6              {console.log("in render")}
7              LifeCycleHooks
8          </div>
9      );
10 }
11
12 export default LifeCycleHooks;
```



First Rendering

- בקומפוננט זה נדגים את הקונספט של rendering כולם עדכון התצוגה לגלש
- ניצור את הקומפוננט `LifeCycleHooks`
 - הפקציה תחזיר הדפסה של מחזורת תווים ואת הכיתוב `LifeCycleHooks`
 - התוצאה בדף
 - ניתן לראות שהדפסה בקונסול מחרוזת התווים שקבענו עם ייצירת הקומפוננט



Initial rendering

Placing an asynchronous method in useState



LifeCycleHooks.jsx

```
15 import { useState } from "react";
16 import Container from "@mui/material/Container";
17
18 const LifeCycleHooks = () => {
19   const [count, setCount] = useState(() => {
20     console.log("in useState function");
21     setTimeout(() => {
22       setCount(prev => prev + 1);
23     }, 2000);
24     return 0;
25   });
26
27   return (
28     <Container maxWidth="lg">
29       {console.log(new Date().toLocaleTimeString())}
30       Count: {count}
31     </Container>
32   );
33 };
34
35 export default LifeCycleHooks;
```

- ניבא את מטודת useState מספרית react
- נוצר קומפוננט בשם LifeCycleHooks
- נחלץ את המשתנה count ומטודה useState מהפעלת מטודה setCount אליה נעביר בארגומנט פונקציית callback שתופעל פעם אחת עם טיענת הקומפוננט. הפלוקציה תדפיס בקונסול מחרוזת תווים
- ולאחר שתי שינויים תנסה לשנות את ערכו של המשתנה count לערך הנוכחי פלוס אחד ולבסוף תעדכן את ערכו של המשתנה count על ידי החזרת הסירה 0
- הפלוקציה תחזיר את הקומפוננט Container של MUI ובתוכה
- נפתח איזור של javascript שם נדפיס את הזמן בו הקומפוננט נטען
- נדפיס את ערכו של המשתנה count

התוצאות בדף

ניתן לראות את הדפסת הטקסט שכתבנו
בתוך פונקציית ה – `callback` שהעבכנו
ארגומנט לMETHOD `useState`
כמו כן ניתן לראות שהקומפוננט נתענה
בשני זמנים שונים

The screenshot shows a browser developer tools console window. At the top, there's a search bar and a toolbar with icons for back, forward, and refresh. Below that, the text "Count: 1" is displayed. The main area shows a stack trace:

```
in useState function LifeCycleHooks.jsx:20
12:48:03      LifeCycleHooks.jsx:28
              react_devtools_backend.js:4066
in useState function
12:48:03      react_devtools_backend.js:4066
12:48:05      LifeCycleHooks.jsx:28
12:48:05      react_devtools_backend.js:4066
>
```

Blue arrows point from the text "ניתן לראות את הדפסת הטקסט שכתבנו" and "בשני זמנים שונים" to the "LifeCycleHooks.jsx:20" and "LifeCycleHooks.jsx:28" entries in the stack trace respectively.



useEffect

Hook that manages the side-effects in functional components.



useEffect as componentDidMount

בעזרת useEffect hook נוכל ליבא נתונים בצורה סינכרונית או אסינכרונית ולעדכן את המפתח הרלוונטי באובייקט ה-state עם הנתונים

- ניצור פונקציה בשם getTime שתחזיר מחרוזת תווים של הזמן שבו קראו לפונקציה

- נשנה את התוכן של הקומפוננט LifeCycleHooks שיצרנו
- נפעיל את מتدות useState עם הספירה אפס כפרמטר וnocház מהמערך שיחזור את המפתחות count setCount
- ניצור אפקט בעזרת useEffect ש:

- בפרמטר הראשון – יקבל פונקציית call back אוניבימית שתפעיל את הקוד הבא:
 - תדפיס את הזמן שמורכב מהשעה והמייל שניות שבה קוראים לפונקציה
 - תעלה את המשתנה count באחד בעזרת מtodot setCount
- בפרמטר השני – נעביר מערך ריק על מנת שפונקציה זאת תפעיל רק פעם אחת
- **הfonktsia תחזיר:**
 - הדפסה בקונסול של הזמן שבו הקומפוננט נתענת מחדש
 - תציג לגולש את הערך של count
 - ניצור שני כפתורים שאחד יעלה באחד והשני יורד באחד את הערך של count

```
43 const getTime = () => {  
44   const date = new Date();  
45   const time = date.toLocaleTimeString();  
46   const mili = date.getMilliseconds();  
47   return `${time}.${mili}`;  
48 };  
49  
50 const LifeCycleHooks = () => {  
51   const [count, setCount] = useState(0);  
52  
53   useEffect(() => {  
54     console.log(`in useEffect: ${getTime()}`);  
55     setCount(prev => prev + 1);  
56   }, []);  
57  
58   return (  
59     <Container maxWidth="lg">  
60       {console.log("in render " + getTime())}  
61       <Box>Count: {count}</Box>  
62       <div>  
63         <Button  
64           variant="outlined"  
65           color="primary"  
66           onClick={() => setCount(prev => prev + 1)}>  
67           +  
68         </Button>  
69         <Button  
70           variant="outlined"  
71           color="primary"  
72           onClick={() => setCount(prev => prev - 1)}>  
73           -  
74         </Button>  
75       </div>  
76     </Container>  
77   );  
78 };
```

התוצאות בדף

The screenshot shows the React DevTools Components tab with two state snapshots. The top snapshot has a count of 2 and the bottom snapshot has a count of 3. Both snapshots show the component tree with file names and line numbers. Arrows point from the text descriptions on the right to the corresponding log entries in each snapshot.

Count: 2

Count: 3

1 Issue: **F 1**

in render 20:51:16.992 [LifeCycleHooks.jsx:58](#) ←
in render 20:51:16.992 [react_devtools_backend.js:4066](#)
in useEffect: 20:51:17.13 [LifeCycleHooks.jsx:53](#) ←
in useEffect: 20:51:17.15 [LifeCycleHooks.jsx:53](#)
in render 20:51:17.18 ← [LifeCycleHooks.jsx:58](#)
in render 20:51:17.20 [react_devtools_backend.js:4066](#)

1 Issue: **F 1**

in render 20:51:16.992 [LifeCycleHooks.jsx:58](#)
in render 20:51:16.992 [react_devtools_backend.js:4066](#)
in useEffect: 20:51:17.13 [LifeCycleHooks.jsx:53](#)
in useEffect: 20:51:17.15 [LifeCycleHooks.jsx:53](#)
in render 20:51:17.18 [LifeCycleHooks.jsx:58](#)
in render 20:51:17.20 [react_devtools_backend.js:4066](#)
in render 20:51:57.838 [LifeCycleHooks.jsx:58](#) ←
in render 20:51:57.838 [react_devtools_backend.js:4066](#)

- ניתן לראות שהקומפוננט נטענת בפעם הראשונה בשעה 20:51:15.992
- מעבר מס' מייל שניות היא מבצעת את הלוגיקה שבתוך useEffect והמשתנה count עולה באחד (במצב של development ההוק useEffect של strict mode עושה ריצה אחת לפני הריצה המרכזית וכן רואים שהוא הולח את המספר בשניים ולא אחד כי שהינו מצלפים)
- הקומפוננט נטענת מחדש בשעה 20:51:17.18 בעקבות השינויים count
- כשלוחצים על כפתור שמעליה ספרה ניתן לראות שהקומפוננט נטענת מחדש אך useEffect לא מגיב לשינוי

useEffect with dependencies

ניתן להוסיף לפרמטר השני של Methodת useEffect לערך משתנה דינמי כך שבכל פעם שהוא משתנה הלוגיקה של useEffect תפעיל

- ניצור שני משתנים בעזרה useState

- count
- num

- כאשר Methodת useEffect תופעל היא תציג בקונסול מחרוזת תווים עם השעה שהיא הופעלה

- עבור לערך התלוויות של Methodת useEffect רק את המשתנה num כך שהיא תעקוף אחר שינויים בו בלבד

```
client > src > sandbox > LifeCycleHooks.jsx > LifeCycleHooks.jsx
38 import { useState, useEffect } from "react";
39 import Container from "@mui/material/Container";
40 import Button from "@mui/material/Button";
41 import Box from "@mui/material/Box";
42
43 const LifeCycleHooks = () => {
44   const [count, setCount] = useState(0); ←
45   const [num, setNum] = useState(0); ←
46
47   useEffect(() => {
48     const date = new Date();
49     const time = date.toLocaleTimeString();
50     const mili = date.getMilliseconds();
51     console.log(`in useEffect: ${time}.${mili}`); ←
52   }, [num]); ←
53
54   return [
55     <Container maxWidth="lg">
56       {console.log("in render " + new Date().toLocaleTimeString())}
57       <Box>Count: {count}</Box>
58     >
59     <div>...
60     </div>
61
62     <Box>Num: {num}</Box>
63   >
64     <div>...
65     </div>
66   </Container>
67   ];
68 };
69
70 export default LifeCycleHooks;
```

התווצהה בדף

The figure consists of three vertically stacked screenshots of a browser's developer tools DevTools panel. Each screenshot shows a component tree on the left and a console log on the right.

- Top Screenshot:** Shows 'Count: 0' in the component tree. The console log shows two 'in render' entries at 14:39:36 and two 'in useEffect' entries at 14:39:36.561 and 14:39:36.562. The 'react_devtools_backend.js' file is mentioned in the logs.
- Middle Screenshot:** Shows 'Count: 1' in the component tree. The console log shows four 'in render' entries at 14:39:36, 14:39:36, 14:40:19, and 14:40:19. It also shows two 'in useEffect' entries at 14:39:36.561 and 14:39:36.562.
- Bottom Screenshot:** Shows 'Count: 1' in the component tree. The console log shows five 'in render' entries at 14:39:36, 14:39:36, 14:40:19, 14:40:36, and 14:40:36. It shows three 'in useEffect' entries at 14:39:36.561, 14:39:36.562, and 14:40:36.828.

- ניתן לראות כי הקומponent נטענה בשעה 14:39:36
- קצת אחרי הופעה מטודת `useEffect`
- ובגלל שלא היה שינוי באף משתנה דינמי שבאובייקט ה `state` הקומponent לא נטענה מחדש
- כארש אנו לוחצים על הכפתור שמשפיע על משתנה `count` (שלא שמננו אותו במערך התלוויות של `useEffect`) אנו רואים כי הקומponent נטענת מחדש אבל `useEffect` אינו מופעל
- אולם כארש אנו משים את ערכו של המשתנה ועת שכך נמצא במערך התלוויות גם הקומponent נטענת מחדש וגם `useEffect` מופעל.

useEffect as componentWillUnmount

- נוכל לקרוא ל `useEffect` רגע לפני שהkomponent מתחלס וنוכל להפעיל את הלוגיקה שלו בנקודת הזאת
- ניצור את מетодת `useEffect` המטודת האנונימית בתוכה תדפיס בקונסול מחרוזת תווים עם השעה שבה `useEffect` הופעלה
- מה שיקרה לאחר המילה `return` יקרה רגע לפני שהkomponent מתחלס וידפס לי בקונסול מחרוזת תווים עם השעה שבה הופעל הקוד.

```
95 import { useEffect } from "react";
96 import Container from "@mui/material/Container";
97
98 const LifeCycleHooks = () => {
99   useEffect(() => {
100     const date = new Date();
101     const time = date.toLocaleTimeString();
102     const mili = date.getMilliseconds();
103     console.log(`in useEffect: ${time}.${mili}`);
104     return () => console.log(`in useEffect return: ${time}.${mili}`);
105   }, []);
106
107   return (
108     <Container maxWidth="lg">
109       {console.log("in render " + new Date().toLocaleTimeString())}
110       in LifeCycleHooks
111     </Container>
112   );
113 };
114
115 export default LifeCycleHooks;
```

התוצאות בדף

ניתן לראות שכארר אנו נכנסים לkomponent

- komponent נתענת
- יש ריצה מקדימה על useEffect על מה שהפונקציה מחזירה
- לאחר מכן יש את ביצוע הקוד ב – useEffect
- רגע לפני שאנו עוברים לkomponent אחר מבוצע הקוד שמחזר מפונקציית useEffect

in LifeCycleHooks

Components

Console

Custom levels

1 Issue: 1

in render 18:08:01 LifeCycleHooks.jsx:108
in render 18:08:01 react_devtools_backend.js:4066
in useEffect: 18:08:01.773 LifeCycleHooks.jsx:103
in useEffect return: 18:08:01.773 LifeCycleHooks.jsx:104
in useEffect: 18:08:01.774 LifeCycleHooks.jsx:103

Error
404
Page not found

Components

Console

Custom levels

1 Issue: 1

in render 18:08:01 LifeCycleHooks.jsx:108
in render 18:08:01 react_devtools_backend.js:4066
in useEffect: 18:08:01.773 LifeCycleHooks.jsx:103
in useEffect return: 18:08:01.773 LifeCycleHooks.jsx:104
in useEffect: 18:08:01.774 LifeCycleHooks.jsx:103
in useEffect return: 18:08:01.774 LifeCycleHooks.jsx:104

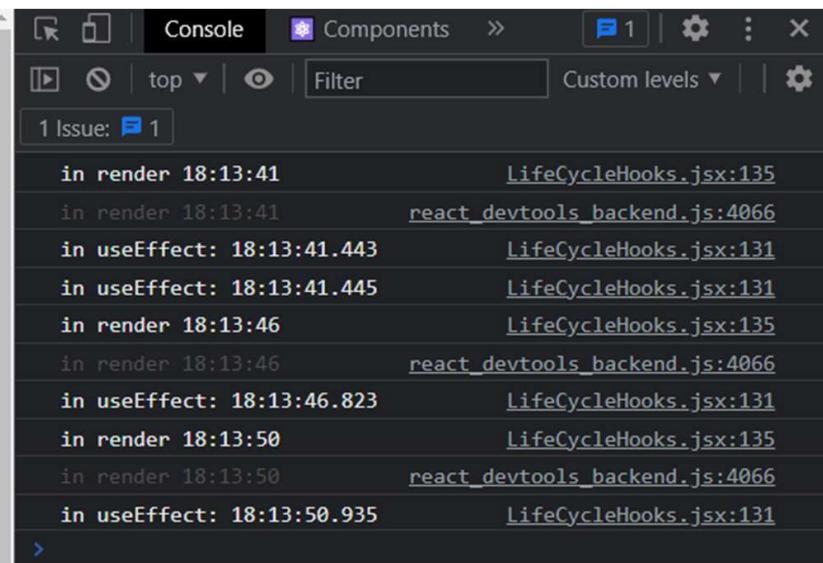
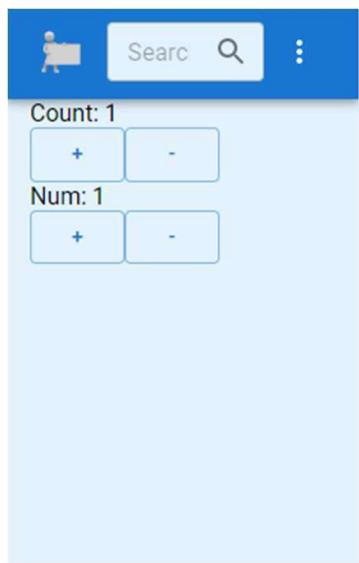
```
118 import { useState, useEffect } from "react";
119 import Container from "@mui/material/Container";
120 import Button from "@mui/material/Button";
121 import Box from "@mui/material/Box";
122
123 const LifeCycleHooks = () => {
124   const [count, setCount] = useState(0); ←
125   const [num, setNum] = useState(0);
126
127   useEffect(() => {
128     const date = new Date();
129     const time = date.toLocaleTimeString();
130     const mili = date.getMilliseconds();
131     console.log(`in useEffect: ${time}.${mili}`); ←
132   });
133
134   return (
135     <Container maxWidth="lg">
136       {console.log("in render " + new Date().toLocaleTimeString())}
137       <Box>Count: {count}</Box>
138     >
139       <div>...
140       </div>
141
142       <Box>Num: {num}</Box>
143     >
144       <div>...
145       </div>
146     </Container>
147   );
148 };
149
150 export default LifeCycleHooks;
```

useEffect as componentDidUpdate

- נהפוך למתודה `useEffect` פרמטר שני
היא תפעול בכל פעם שתהיה טעינה חדשה
של הדף בלי קשר למשתנה ספציפי
שהשתנה
 - נחזיר למשתנים הדינמיים `num` כערך
ששינויו בהם יטען מחדש את הדף בעזרת
متודת `useState`
 - הפעם לא נהפוך פרמטר שני למתודה
`useEffect` ונגידיר שכשהיא תפעול היא
תדפיס בקונסול את השעה שהפעילו אותה
 - הפונקציה מחזירה כפותורים ותצוגה של
המשתנים הדינמיים

התווצה בדף

ניתן לראות כי כל טעינה חדשה של הדף בעקבות שינוי ערכו של משתנה דינמי מפעילה את METHOD useEffect.



The screenshot shows the React DevTools interface. On the left, the Components tab displays a component tree with a single component having a 'Count' prop set to 1. On the right, the Console tab shows a log of events:

```
in render 18:13:41          LifeCycleHooks.jsx:135
in render 18:13:41          react_devtools_backend.js:4066
in useEffect: 18:13:41.443    LifeCycleHooks.jsx:131
in useEffect: 18:13:41.445    LifeCycleHooks.jsx:131
in render 18:13:46          LifeCycleHooks.jsx:135
in render 18:13:46          react_devtools_backend.js:4066
in useEffect: 18:13:46.823    LifeCycleHooks.jsx:131
in render 18:13:50          LifeCycleHooks.jsx:135
in render 18:13:50          react_devtools_backend.js:4066
in useEffect: 18:13:50.935    LifeCycleHooks.jsx:131
```

Custom hooks

Building your own Hooks lets you extract component logic into reusable functions

<https://reactjs.org/docs/hooks-custom.html>





Rules

All the rules that apply to hooks that we import from libraries like React apply to custom hooks

The name of the hook must start with use

useCounter

JS useCounter.js U X

```
client > src > sandbox > custom-hook > JS useCounter.js > ...
1 import { useState } from "react";
2 import { number } from "prop-types";
3
4 const useCounter = (initialCount = 0) => {
5   const [count, setCount] = useState(initialCount);
6
7   const increment = () => setCount(prev => prev + 1);
8   const decrement = () => setCount(prev => prev - 1);
9   const reset = () => setCount(initialCount);
10  return [count, increment, decrement, reset];
11};
12
13 useCounter.propTypes = {
14   initialCount: number,
15 };
16
17 export default useCounter;
```

פונקציית hook שתנהל לנו את ה - counter של ה state

- ניבא את useState
- נוצר את הפונקציה useCounter שתתקבל בפרמטר initialCount מסוג של מספר עם ערך דיפולטיבי של אפס
- נפעיל את מتدת useState עם initialCount כפרמטר ונחלץ מהמערך shizor את המפתחות count setCounet שתעלה את count של count באחד
- נוצר את מتدת increment שתוריד את count של count באחד
- נוצר את מتدת decrement שתוריד את count של count באחד
- נוצר את מتدת reset שתאפשר את initialCount לערך של count
- נציג מהפונקציה מערך עם המשתנה count ושאר הפונקציות שייצרנו count
- נודא בעזרת סדרית propTypes שאר מקבלים בפרמטר של הפונקציה count מסוף.

Counter.jsx

```
client > src > sandbox > custom-hook > Counter.jsx > Counter > mt
1 import React from "react";
2 import useCounter from "./useCounter";
3 import Button from "@mui/material/Button";
4 import Box from "@mui/material/Box";
5 import Typography from "@mui/material/Typography";
6 import Paper from "@mui/material/Paper";
7
8 const Counter = () => {
9   const [count, increment, decrement, reset] = useCounter();
10
11   return (
12     <Box sx={{ display: "flex", justifyContent: "center" }}>
13       <Paper sx={{ width: 500, mt: 2 }}>
14         <Box>
15           <Typography align="center">Count: {count}</Typography>
16
17           <Box>
18             <Button onClick={increment} variant="outlined" sx={{ m: 2 }}>
19               Increment
20             </Button>
21             <Button onClick={decrement} variant="outlined" sx={{ m: 2 }}>
22               decrement
23             </Button>
24             <Button onClick={reset} variant="outlined" sx={{ m: 2 }}>
25               reset
26             </Button>
27           </Box>
28         </Box>
29       </Paper>
30     </Box>
31   );
32 }
33
34 export default Counter;
```

שימוש ב custom hook

אם לא נervoir למетодה `useEffect` פרמטר שני
היא תפעל בכל פעם שתהיה טעינה מחדש
של הדף בלי קשר למשתנה ספציפי
שהשתנה

- ניבא את `useCounter`
- נפעיל את `useCounter` שהוא מוחזיר האיברים במערך שהוא מוחזיר
- נציג את ערכו של המשתנה `count`
- בלחיצה על הceptor נפעיל את מетодת `increment`
- בלחיצה על הceptor נפעיל את מетодת `decrement`
- בלחיצה על הceptor נפעיל את מетодת `reset`

התווצה בדף

ניתן לראות שה counter שלנו עובד
מצוין ולמרות שאמנו מנהלים את המפתח
count שבתוך אובייקט ה – state מחוץ
לקומponent הולך עובד חלק

Count: 1

INCREMENT DECREMENT RESET

משימת custom hook



Business-cards-app

- צור custom hook בשם `useName`
- הhook יקבל פרמטר `initialName`
- הוא יחלץ מהhook `useState` את המשתנה `name`
- והמטודה `setName`
- יצא מהmethode אובייקט עם המשתנה והmethode שיצרת
- השתמש בהוק `useName` בкомпонент בשם `CustomName.jsx`
- יבא את `useName` חלץ ממנו את המפתחות
- השתמש באלמנט מסווג `input` כדי לשנות את השם
- הצג את השם והשינויים בו

Memoization

שמירת פונקציות ומשתנים ב – cache כך שלא יצרו אותם
בכל פעם שהקומפוננט נטען מחדש





Definition

an optimization technique used primarily to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again



useCallback

A hook that receives a callback function and an array of dependencies and return a memoized version of the callback that only changes if one of the dependencies has changed.

<https://reactjs.org/docs/hooks-reference.html#usecallback>



ButtonComp.jsx

ButtonComp.jsx U X

```
client > src > sandbox > Memoization > ButtonComp.jsx > ...
1  import { func, string } from "prop-types";
2  import Button from "@mui/material/Button";
3
4  const ButtonComp = ({ handleClick, children }) => {
5    console.log(`Rendering Button: ${children}`);
6    return (
7      <Button variant="outlined" onClick={handleClick} color="primary">
8        {children}
9      </Button>
10    );
11  };
12
13  ButtonComp.propTypes = {
14    handleClick: func.isRequired,
15    children: string.isRequired,
16  };
17
18  export default ButtonComp;
```

ראשית נראה את בעיית האופטימיזציה
ולאחר מכן נראה איך נפתר אותה
בעזרת `useCallback`

- ניצור קומponent בשם `ButtonComp`
 - הקומponent מקבל מפתחות לאובייקט `handleClick` ו- `children` מסוג מחרוזת תווים
 - היא תדפיס בקונסול מחרוזת תווים
 - היא תחזיר כפטור של `func.isRequired` שלחיצה עליו `handleClick` את מетодת `onClick` של הפעונציה
- גודא בעזרת ספרית `propTypes` שהפונקציה מקבלים בפרמטר של `children`.

UseCallBackComp.jsx

```
client > src > sandbox > Memoization > UseCallBackComp.jsx > ...
1 import { useState } from "react";
2 import Box from "@mui/material/Box";
3 import Typography from "@mui/material/Typography";
4 import Paper from "@mui/material/Paper";
5 import ButtonComp from "./ButtonComp"; ←
6
7 const UseCallBackComp = () => {
8   const [age, setAge] = useState(1);
9   const [height, setHeight] = useState(0); } ←
10
11 const incrementAge = () => setAge(age + 1); ←
12 const incrementHeight = () => setHeight(height + 1); ←
13
14 return (
15   <Box sx={{ display: "flex", justifyContent: "center" }}>
16     <Paper sx={{ width: 350, mt: 2 }}>
17       <Box>
18         <Typography align="center">Age: {age}</Typography>
19         <Typography align="center">Height: {height}</Typography> } ←
20
21         <Box sx={{ display: "flex", justifyContent: "space-between", m: 2 }}>
22           <ButtonComp handleClick={incrementAge}>age</ButtonComp>
23           <ButtonComp handleClick={incrementHeight}>height</ButtonComp> } ←
24         </Box>
25       </Box>
26     </Paper>
27   </Box>
28 );
29 }
30
31 export default UseCallBackComp;
```

UseCallBackComp.jsx

- ניבא את הקומפוננט שיצרנו
- נוצר את הקומפוננט UseCallBackComp
 - נפעיל פעמים את מетодת useState ונהלץ ממנה את המשתנים והMETHODOT
 - ניצור קבוע בשם incrementAge שווה ערך לפונקציה אונומית שתפעיל את METHODOT setAge שתעלה את המשתנה age באחד
 - ניצור קבוע בשם incrementHeight שווה ערך לפונקציה אונומית שתפעיל את METHODOT setHeight שתעלה את המשתנה height באחד
 - נציג את המשתנים age height
 - נציב את ButtonComp כאשר לכפתור אחד נעביר את incrementAge ולכפתור השני את incrementHeight

התוצאות בדף

The figure consists of two vertically stacked screenshots of a web application interface. Both screenshots show a search bar at the top with placeholder text 'Search' and a magnifying glass icon. Below the search bar are two buttons labeled 'AGE' and 'HEIGHT'. In the first screenshot, the text 'Age: 1' and 'Height: 0' is displayed. In the second screenshot, the text 'Age: 2' and 'Height: 0' is displayed. To the right of each button, there is a vertical stack of four lines of text in a monospaced font, representing the output of a browser's developer tools Console tab. The text shows the rendering of 'Button: age' and 'Button: height' from 'ButtonComp.jsx:5' and 'react_devtools_backend.js:4066'.

```
Rendering Button: age      ButtonComp.jsx:5
Rendering Button: age      VM826 installHook.js:1861
Rendering Button: height   ButtonComp.jsx:5
Rendering Button: height   VM826 installHook.js:1861
```



```
Rendering Button: age      ButtonComp.jsx:5
Rendering Button: age      react_devtools_backend.js:4066
Rendering Button: height   ButtonComp.jsx:5
Rendering Button: height   react_devtools_backend.js:4066
```

- ניתן לראות שברנדור הראשוני מודפס ליש בקונסול שני הcptors הופעלו
- אולם כאשר אני לוחץ על אחד מהם שני הcptors נטען מחדש
- הסיבה לכך היא שכל עם שהקומפוננט נטען מחדש, react ייצור את כל הfonkציות שבתוך הקומפוננט מחדש וכך גם מעולם לא נוצרו אותן מחדש שלחן זהה. והfonkציות נאליות שלחן זהה.

ButtonComp.jsx

```
20 import { memo } from "react"; ←
21 import { func, string } from "prop-types";
22 import Button from "@mui/material/Button";
23
24 const ButtonComp = ({ handleClick, children }) => {
25   console.log(`Rendering Button: ${children}`);
26   return (
27     <Button variant="outlined" onClick={handleClick} color="primary">
28       {children}
29     </Button>
30   );
31 };
32
33 ButtonComp.propTypes = {
34   handleClick: func.isRequired,
35   children: string.isRequired,
36 };
37
38 export default memo(ButtonComp); ←
```

- הדרך לפטור בעיה זאת מתחילה בЛОמר לריקט כי הקומפוננט מקבלת באובייקט ה - props שלה מפתחות שאת ערכם אנו מעוניינים לטעון מחדש רק כאשר יש שינוי זהה שיצירר את יצירת הפונקציה שקומפוננט קיבלה מחדש
- NEYBA AT meno MATORE SPERIYAT react
- הקומפוננט נשארת אותו דבר בלבד החלק שאנו מיצאים
- געטוף את יצא הקומפוננט בערך שיחזור מהפעלת מטודת meno שתגיד לריקט לבדוק מטודה שהקומפוננט קיבלה באובייקט ה - props השנתנה בצויה צאת שצירר להעביר את הפונקציה מחדש לקומפוננט

```

33 import { useState, useCallback } from "react";
34 import Box from "@mui/material/Box";
35 import Typography from "@mui/material/Typography";
36 import Paper from "@mui/material/Paper";
37 import ButtonComp from "./ButtonComp";
38
39 const UseCallBackComp = () => {
40   const [age, setAge] = useState(1);
41   const [height, setHeight] = useState(0);
42
43   const incrementAge = useCallback(() => {
44     setAge(age + 1);
45   }, [age]);
46
47   const incrementHeight = useCallback(() => {
48     setHeight(height + 1);
49   }, [height]);
50
51   return (
52     <Box sx={{ display: "flex", justifyContent: "center" }}>
53       <Paper sx={{ width: 350, mt: 2 }}>
54         <Box>
55           <Typography align="center">Age: {age}</Typography>
56           <Typography align="center">Height: {height}</Typography>
57
58           <Box sx={{ display: "flex", justifyContent: "space-between", m: 2 }}>
59             <ButtonComp handleClick={incrementAge}>age</ButtonComp>
60             <ButtonComp handleClick={incrementHeight}>height</ButtonComp>
61           </Box>
62         </Box>
63       </Paper>
64     </Box>
65   );
66 }
67
68 export default UseCallBackComp;

```

useCounter

cut ניבא את מетодת
מתוך ריאקט

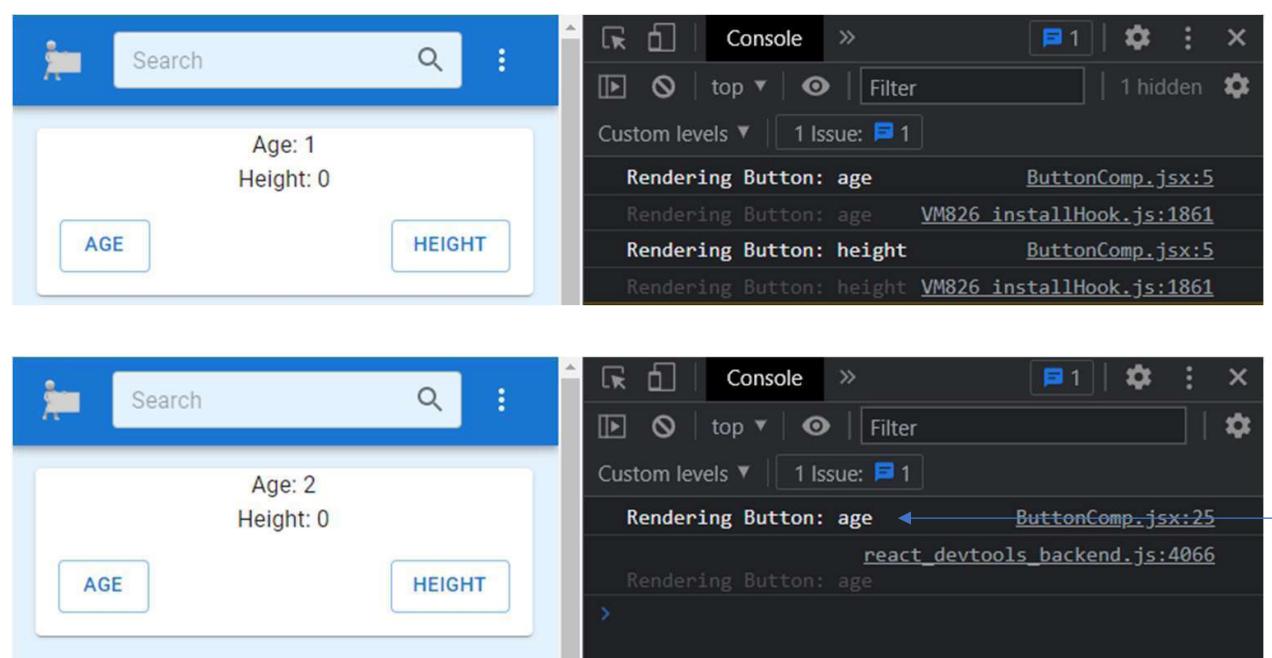
- געטוף את המethodות incrementAge incrementHeight בMETHOD שמקבלת שני פרמטרים:

- הראשון הוא פונקציה
- השני מערך של תלויות ששינוייהם יגרום ליצירת הפונקציה מחדש

- כל השאר נשאר אותו הדבר

התווצה בדף

ניתן לראות כי עכשו לאחר הטעינה הראשונית וניקיון הקונסול לחיצה על אחד הcptורים יתען מחדש רק אותו ואת הפונקציות שהוא מקבל





useMemo

React Hook that allows you to memoize expensive functions so that you can avoid calling them on every render.

useMemo hook accepts as an argument a function for which it will perform a process of memoize and dependent array

useMemo will only recompute the memoized value when one of the inputs has changed.

<https://reactjs.org/docs/hooks-reference.html#usememo>



UseMemo.jsx

ראשית נגידר את הפעולה

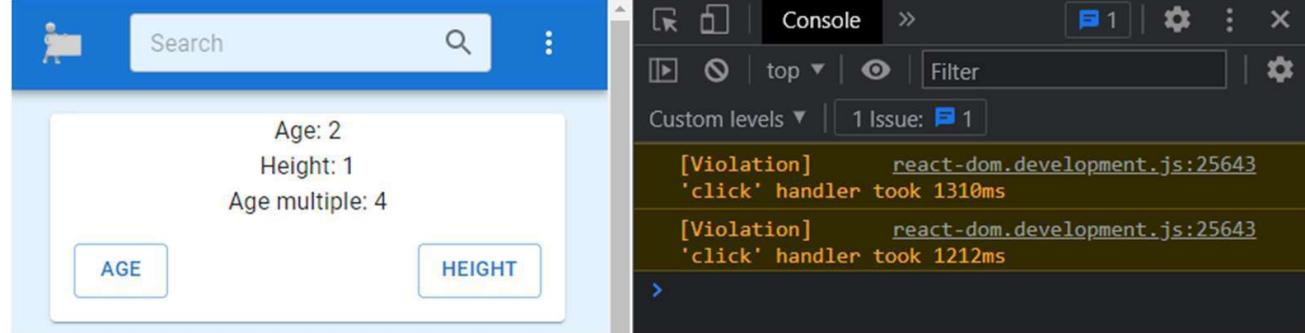
• בקומponent **UseMemo**

- נפעיל פעמים את מטודת `useState` ונהלץ ממנה את המשתנים והMETHODOT
- ניצור קבוע בשם `incrementAge` שיהיה שווה ערך לפונקציה אונימית שתפעיל את מטודת `setAge` שתעלה את המשתנה `age` באחד
- ניצור קבוע בשם `incrementHeight` שיהיה שווה ערך לפונקציה אונימית שתפעיל את מטודת `setHeight` שתעלה את המשתנה `height` באחד
- ניצור פונקציה בשם `slowFunction` שכךüber משך של לולאה מאוד ארוכה מחזירה את `age` כפול שתיים. מצב זה נועד כדי לדמות פונקציה "יקרה" שמעמיסה על משאבי מערכת.
- נציג את המשתנים `age` `height`
- נציג בתוך ערך את הפעלת מטודת `slowFunction` ונוביר לה כารוגמנט את המשתנה `age`

```
client > src > sandbox > Memoization > UseMemo.jsx > ...  
1 import Button from "@mui/material/Button";  
2 import Box from "@mui/material/Box";  
3 import { useState } from "react";  
4 import Typography from "@mui/material/Typography";  
5 import Paper from "@mui/material/Paper";  
6  
7 const UseMemo = () => {  
8   const [age, setAge] = useState(1);  
9   const [height, setHeight] = useState(0); }  
10  
11 const incrementAge = () => setAge(age + 1); ←  
12 const incrementHeight = () => setHeight(height + 1); ←  
13  
14 const slowFunction = () => { ←  
15   for (let i = 0; i < 2_000_000_000; i++) {}  
16   return age * 2;  
17 };  
18  
19 return (  
20   <Box sx={{ display: "flex", justifyContent: "center" }}>  
21     <Paper sx={{ width: 350, mt: 2 }}>  
22       <Box>  
23         <Typography align="center">Height: {height}</Typography>  
24         <Typography align="center">Age multiple: {slowFunction()}</Typography>  
25  
26         <Box sx={{ display: "flex", justifyContent: "space-between", m: 2 }}>  
27           <Button variant="outlined" onClick={incrementAge}>  
28             | age  
29             |</Button>  
30           <Button variant="outlined" onClick={incrementHeight}>  
31             | height  
32             |</Button>  
33         </Box>  
34       </Box>  
35     </Paper>  
36   </Box>  
37 );  
38  
39 export default UseMemo;
```

התוצאות בדף

לחיצה על כל אחד מהכפתורים לוקחת לפחות 1212ms וזו את בגין שבכל פעם שימושה דינامي משתנה, הקומפוננט `slowFunction` נטענת מחדש והפונקציה `slowFunction` נוצרת מחדש ומופעלת



UseMemo.jsx

הקוד נשאר זהה מלבד השינויים הבאים

- ניבא את `useMemo`
- נסנה את הערך של הקבוע `slowFunction` להפעלת מטודת `useMemo` שתתקבל בפרמטר הראשון את הפונקציה הארוכה שיצרנו ובפרמטר השני מערך תלויות כך שהפונקציה תבנה מחדש רק כאשר יהיה שינוי במשתנה שנמצא בתוך מערך התלויות.

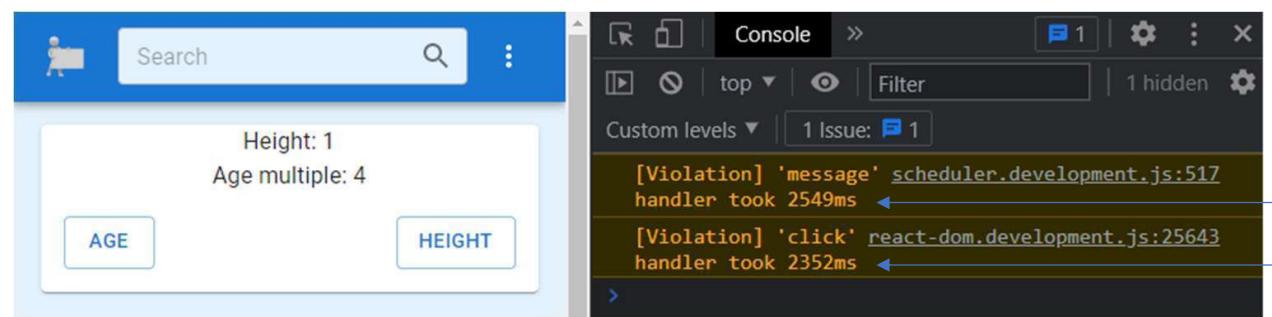
- נציג לגולש את הקבוע `slowFunction` שערכו הוא הערך שיחזור ממתודת `useMemo`

! לא ניתן להעביר לפונקציה שבתוכה `useMemo` ערכים

```
42 import Button from "@mui/material/Button";
43 import Box from "@mui/material/Box";
44 import { useState, useMemo } from "react"; ←
45 import Typography from "@mui/material/Typography";
46 import Paper from "@mui/material/Paper";
47
48 const UseMemo = () => {
49   const [age, setAge] = useState(1);
50   const [height, setHeight] = useState(0);
51
52   const incrementAge = () => setAge(age + 1);
53   const incrementHeight = () => setHeight(height + 1);
54
55   const slowFunction = useMemo(() => { ←
56     for (let i = 0; i < 2_000_000_000; i++)
57       return age * 2;
58   }, [age]);
59
60   return (
61     <Box sx={{ display: "flex", justifyContent: "center" }}>
62       <Paper sx={{ width: 350, mt: 2 }}>
63         <Box>
64           <Typography align="center">Height: {height}</Typography>
65           <Typography align="center">Age multiple: {slowFunction}</Typography> ←
66
67           <Box sx={{ display: "flex", justifyContent: "space-between", m: 2 }}>
68             <Button variant="outlined" onClick={incrementAge}>
69               age
70             </Button>
71             <Button variant="outlined" onClick={incrementHeight}>
72               height
73             </Button>
74           </Box>
75         </Box>
76       </Paper>
77     </Box>
78   );
79 }
80
81 export default UseMemo;
```

התווצה בדף

הפעם הטעינה הראשונית לוקחת 2549
לחיצה על כפתור age לוקחת 2352
אולם לחיצה על כפתור HEIGHT לא
נרשמת פעולה שלקחה זמן רב



axios

Promise based HTTP client for the browser and node.js

<https://axios-http.com/docs/intro>

לפני החלק זה יש להוריד את השרת של node.js ולבור על המציגת של התקנת db
mongodb





Q Definition

Axios is a simple promise-based HTTP client for the browser and node.js. Axios provides a simple to use library in a small package with a very extensible interface.



Benefits

supports older browsers

has a way to abort a request

has a way to set a response timeout

has built-in Cross-Site Request Forgery protection

supports upload progress

performs automatic JSON data transformation.

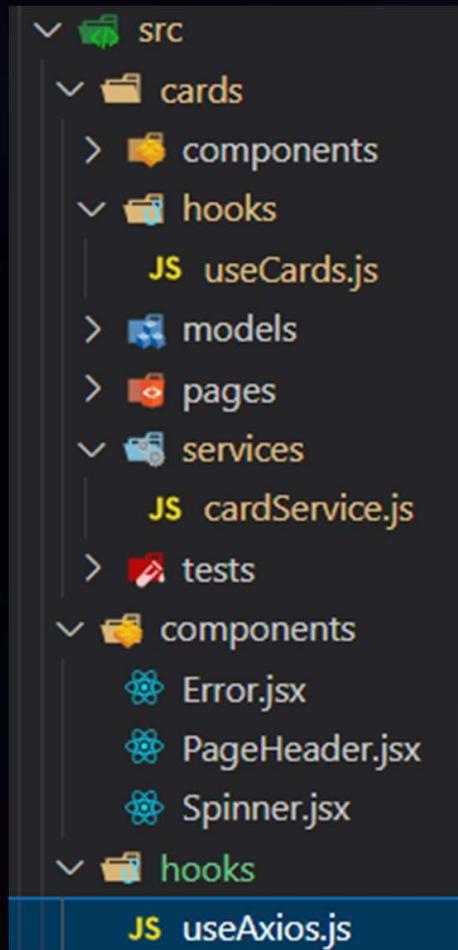


Installation

npm i axios

הכנות תשתית

- ניצור את הנתיב `src/cards/hooks/useCards.js`
- ניצור את הנתיב `src/cards/service/card ApiService.js`
- ניצור את הנתיב `src/component/Error.jsx`
- ניצור את הנתיב `src/component/Spinner.jsx`
- ניצור את הנתיב `src/hooks/useAxios.js`





cardApiService

מודול שירכז את הקריאהות מצד לקוח לצד
שירות לצורך ביצוע פעולות ה – CRUD על
כרטיסים/ כרטיסים



cardService.js

- ניבא מופע מסכנית axios לתוכה משתנה בשם axios
- נקשר בין המודול הזה למודול של axiosService על ידי יבוא המודול ניצור קבוע בשם apiUrl שערך יהיה או הערך שיחזור מ process.env.REACT_APP_API_URL או הכתובת הרשומה
- ניצור קבוע בשם getCards ווניצא אותו כמפתח באובייקט exports שערך היה פונקציה אסינכרונית
- שתחלץ את המשתנה data מתוך הערך שיחזור מהפעלת המטודה האסינכרונית של axios.get אליה נעביר כארגומנט את הכתובת אליה תשלח הבקשה
- נחזיר את הקבוע date
- אם תהיה שגיאה נתפוא איתה באמצעות מנגנון try & catch ונחזיר הודעת השגיאה עם הורדת השגיאה באמצעות Promise.reject

```
js cardService.js ×  
client > src > cards > services > js cardService.js > ...  
1 import axios from "axios";  
2 import "../../services/axiosService"; ←  
3  
4 const apiUrl = process.env.REACT_APP_API_URL || "http://localhost:8181"; ←  
5  
6 export const getCards = async () => { ←  
7   try { ←  
8     const { data } = await axios.get(` ${apiUrl}/cards`); ←  
9     return data; ←  
10  } catch (error) { ←  
11    return Promise.reject(error.message); ←  
12  }  
13};
```

Error.jsx

יצירת קומponent למקירה ויש שגיאות שמתקבלות מהשרת

- הקומponent מקבל בפרמטר errorMessage מסוג טקסט
- מציג את הודעת השגיאה לגולש
- לצד תמונה של רובוט שבור מתוך תיקית images

Error.jsx X

client > src > components > Error.jsx > [x] Error

```
1 import React from "react";
2 import { string } from "prop-types";
3 import Container from "@mui/material/Container";
4 import Grid from "@mui/material/Grid";
5 import Typography from "@mui/material/Typography";
6
7 const Error = ({ errorMessage }) => {
8   return (
9     <Container>
10       <Grid container spacing={2}>
11         <Grid item xs={12} md={8}>
12           <Typography variant="h5" color="initial">
13             Oops... something went wrong: {errorMessage}
14           </Typography>
15         </Grid>
16         <Grid item xs={12} md={4} justifyContent="center">
17           
22         </Grid>
23       </Grid>
24     </Container>
25   );
26 }
27
28 Error.propTypes = {
29   errorMessage: string.isRequired,
30 };
31
32 export default Error;
```

Spinner.jsx

בקומפוננט זה אציג אнимציה שתפעל כל עוד לא התקבלה תשובה מהשרת

- ניצור קומפוננט בשם Spinner שיקבל באובייקט הפרופ

- color – מחרוזת תווים עם ערך דיפולטיבי
- Size – מספר עם ערך דיפולטיבי
- height – מסוג של מספר או מחרוזת תווים

- הפונקציה תחזיר קומפוננט שבתוכו האנימציה של UI M בשם CirularProgress אליו אני מעביר את הפרמטרים שהקומפוננט קיבל במידה ואני מעוניין לשנות את הערכים הדיפולטיבים שהציבתי לו

Spinner.jsx X

```
client > src > components > Spinner.jsx > ...  
1 import React from "react";  
2 import { string, number, oneOfType } from "prop-types";  
3 import CircularProgress from "@mui/material/CircularProgress";  
4 import Box from "@mui/material/Box";  
5  
6 const Spinner = ({ color = "primary", size = 40, height = "50vh" }) => {  
7   return (  
8     <Box  
9       sx={{  
10         display: "flex",  
11         justifyContent: "center",  
12         minHeight: { height },  
13       }}>  
14     <CircularProgress ←  
15       color={color}  
16       size={size}  
17       sx={{ alignSelf: "center" }}  
18     />  
19   </Box>  
20 );  
21 };  
22  
23 Spinner.propTypes = {  
24   color: string,  
25   size: number,  
26   height: oneOfType([string, number]),  
27 };  
28  
29 export default Spinner;
```

CardsPage.jsx

קומponent שמשתמש ב - cardService

- ניבא את getCards מתוך המודול שיצרנו cardService
- פעיל את מетод useState שלוש פעמים ונחלץ ממנה את המשתנים cards error isPending setError setPending

- פעיל את מетод useEffect שתפעל פעם אחת עם יצירת הקומponent ותפעל את פונקציית ה – callback :
 - תנסה את ערכו של המשתנה isPending ל – true

- פעיל את מетод then. שתפעיל פונקציה שתקבל בפרמטר את data
 - תנסה את ערכו של המשתנה isPending ל – false
 - תעשה השמה למשתנה cards לערך data
- נשרר אליה את מетод catch. כך שאם תחזיר שגיאה false
 - תנסה את ערכו של המשתנה isPending ל – false
- נעשה השמה למשתנה error עם השגיאה שקיבלנו

דף לוגישן

- אם הערך של isPending הוא לא פולסיבי נציג את קומponent Spinner שיצרנו
- אם הערך של error הוא לא פולסיבי נציג את הקומponent Error ונעביר לו errorMessage באובייקט הפורוס את השגיאה
- נבדוק ואם מערך הCARDSים ריק נציג לוגישן את מחזורת התווים הבאה
- אם יש CARDSים להציג

```
client > src > cards > pages > CardsPage.jsx
1 import Container from "@mui/material/Container";
2 import Cards from "../../components/Cards";
3 import PageHeader from "../../components/PageHeader";
4 import { useEffect, useState } from "react";
5 import { getCards } from "../../services/cardService";
6 import Error from "../../components/Error";
7 import Spinner from "../../components/Spinner";
8
9 const CardsPage = () => {
10   const [cards, setCards] = useState();
11   const [error, setError] = useState(null);
12   const [isPending, setPending] = useState(false);
13
14   useEffect(() => {
15     setPending(true);
16     getCards()
17       .then(data => {
18         setPending(false);
19         setCards(data);
20       })
21       .catch(error => {
22         setPending(false);
23         setError(error);
24       });
25   }, []);
26
27   return (
28     <Container>
29       <PageHeader
30         title="Cards"
31         subtitle="Here you can find business cards from all categories"
32       />
33       {isPending && <Spinner />}
34       {error && <Error errorMessage={error} />}
35       {cards && !cards.length &&
36         <p>Oops, there are no business cards in the database that match<br>the parameters you entered</p>}
37       {cards && !cards.length && <Cards cards={cards} />}
38     </Container>
39   );
40 }
41
42 export default CardsPage;
```



CardsFeedback.jsx

קומפוננט לחישוי מצב הלקוחות



CardsFeedback.jsx

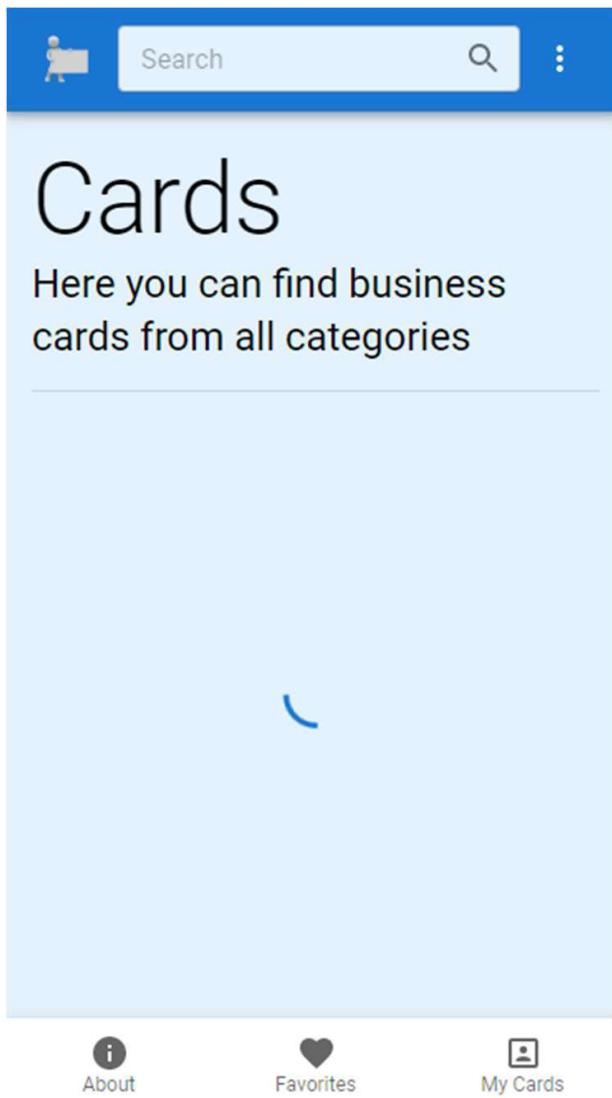
- ניבא את המטודות הkomponenot והמשתנים שדרושים לנו לצורך הלוגיקה של הקומפוננט
 - ניצור את komponent Feedback Cards
 - הקומפוננט מקבל את המפתחות הלל באובייקט הprofop
 - נתנה ואם המשתנה isLoading עם ערך true חיובי נציג את הקומפוננט Spinner
 - נתנה ואם המשתנה error עם ערך true נציג את komponent Error
 - נתנה ואם אין אורך למרכז crtisim נציג לגולש את הכתוב הבא
 - אם יש אורך למרכז crtisim נציג את komponent Cards
 - אם הקומפוננט לא עומדת באף התניריה נחריר null
 - בעזרת ספריות PropTypes נוודא שהקומפוננט אכן קיבלת בפרמטרי שלה את מה שהיא צריכה על מנת לבצע את הלוגיקה שלה

```
client > src > cards > components > CardsFeedback.jsx > ...
1 import React from "react";
2 import { object, arrayOf, bool, string, func } from "prop-types";
3 import Spinner from "../../components/Spinner";
4 import Error from "../../components/Error";
5 import Cards from "./Cards";
6
7 const CardsFeedback = ({ isLoading, error, cards, onDelete }) => {
8   if (isLoading) return <Spinner />;
9   if (error) return <Error errorMessage={error} />;
10  if (cards && !cards.length)
11    return (
12      <div>
13        {" "}
14        Oops, there are no business cards in the database that match the
15        parameters you entered
16      </div>
17    );
18  if (cards) return <Cards cards={cards} onDelete={onDelete} />;
19  return null;
20};
21
22 CardsFeedback.propTypes = {
23   isLoading: bool.isRequired,
24   error: string,
25   cards: arrayOf(object),
26   onDelete: func.isRequired,
27 };
28
29 export default CardsFeedback;
```

```
❶ CardsPage.jsx M X
client > src > cards > pages > ❷ CardsPage.jsx > ...
1 import Container from "@mui/material/Container";
2 import PageHeader from "../../components/PageHeader";
3 import { useEffect, useState } from "react";
4 import CardsFeedback from "../components/CardsFeedback";
5 import { getCard } from "../services/cardService";
6
7 const CardsPage = () => {
8   const [cards, setCards] = useState([]);
9   const [error, setError] = useState(null);
10  const [isLoading, setLoading] = useState(false);
11
12  > useEffect(() => { ...
13    }, []);
14
15  const onDeleteCard = () => {};
16
17  return (
18    <Container>
19      <PageHeader
20        title="Cards"
21        subtitle="Here you can find business cards from all categories"
22      />
23
24      <CardsFeedback <-->
25        isLoading={isLoading}
26        error={error}
27        cards={cards}
28        onDelete={onDeleteCard}
29      />
30    </Container>
31  );
32};
33
34
35
36
37
38
39
40
41
42
43
44
45
46 export default CardsPage;
```

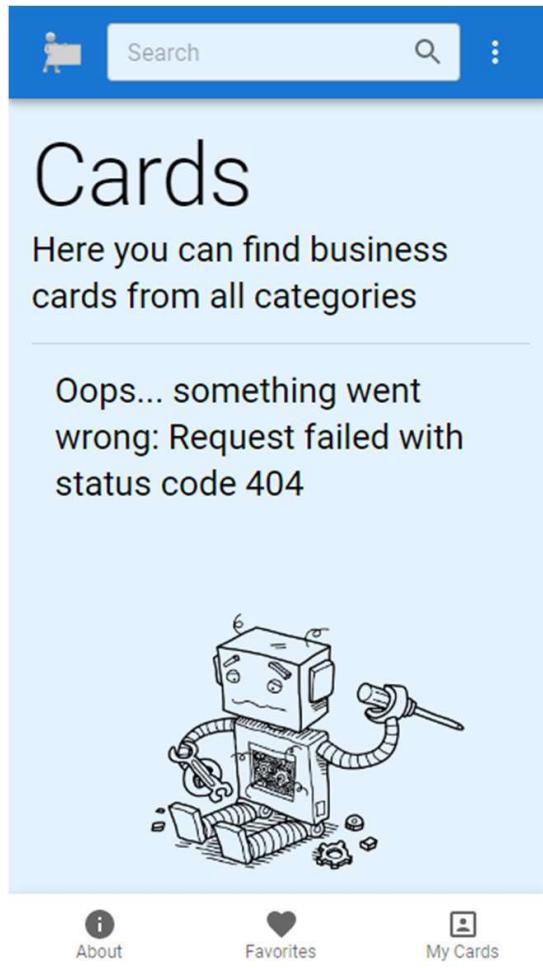
Cards.jsx

- נציג את הקומונט **CardsFeedback** שהוא
ונעביר לה את המשתנים שהוא
מבקש לקלבל **props** מפתחות באובייקט ה



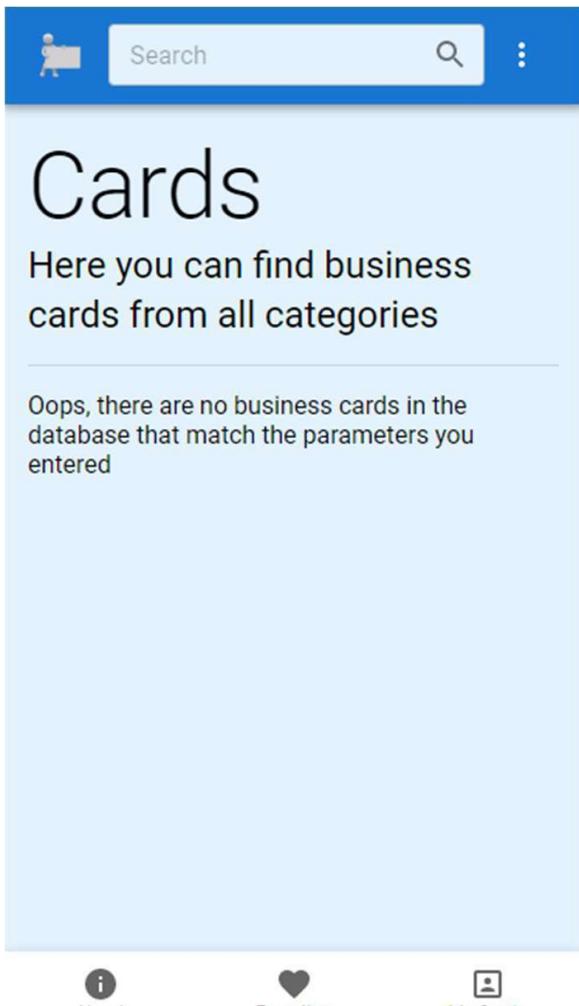
התווצה בדף

כשהקומפוננט מתחילה למידע והקומפוננט spinner עובד



התווצה בדף

כשמקבלת שגיאה שמיורה בשרת



התווצה בדף

אם מתתקבל מערך ריק ממאגר המידע או שעקב חיפוש כרטיס עם פרמטרים מסוימים אף כרטיס לא עמד בקריטריון של הפרמטרים שהזנו (למשל בשדה החיפוש) יוצג לגולש הכיתוב הבא

BCard ABOUT MY CARDS FAV CARDS SANDBOX Search SIGNUP LOGIN

Cards

Here you can find business cards from all categories



first card
this is the first card

Phone: 050-0000000
Address: test 3 test
Card Number: 1482923

second card
this is the second card

Phone: 050-0000000
Address: test 3 test
Card Number: 5968240

third card
this is the third card

Phone: 050-0000000
Address: test 3 test
Card Number: 1529634

 About  Favorites  My Cards



משימת cardApiService חלק א'



Business-cards-app

- צור הפקציות האсинכרוניות הבאות בנתיב :src/cards/services/cardApiService.js

getCard

- הפקציה מקבל בפרמטר cardId מסוג של מחרוזת תווים שתציג תעודת זהות של כרטיס
- פועל את METHOD axios.get עם הנטיב בשרת שאחראי על היבאת כרטיס בודד וחלץ ממנו את מפתח data.
- החזר את data
- אם מתבלט שגיאה תפום אותה באמצעות מנגןן try & catch

getMyCards

- פועל את METHOD axios.get עם הנטיב בשרת שאחראי על היבאת כל כרטיסי הביקור של המשתמש וחלץ ממנו את מפתח data.
- החזר את data
- אם מתבלט שגיאה תפום אותה באמצעות מנגןן try & catch



חלק ב'

- **creactCard** •
 - הfonקציה מקבל בפרמטר card מסוג של אובייקט המיצג כרטיס ביקור לעסק
 - הפעל את מетодת axios.post כasher ברגומנט הראשון הצב את כתובות ה – URL ליצירת כרטיס ובפרמטר השני העבר את card אותו הfonקציה קיבלה בפרמטר וחלץ ממנו את מפתח data.
 - החזר את data מהfonקציה
 - אם מתקיים שגיאה תפום אותה באמצעות מנגןן try & catch ו诙זר Promise.reject עם הודעה השגיאה
- **editCard** •
 - הfonקציה מקבל בפרמטר card מסוג של אובייקט המיצג כרטיס ביקור לעסק
 - הפעל את מетодת axios.put כasher ברגומנט הראשון הצב את כתובות ה – URL לעריכת כרטיס ביקור ובפרמטר השני העבר את card וחלץ ממנו את מפתח data.
 - החזר את data
 - אם מתקיים שגיאה תפום אותה באמצעות מנגןן try & catch ו诙זר Promise.reject עם הודעה השגיאה

חלק ג'



changeLikeStatus •

- הֆונקציה מקבל בפרמטר cardId מסוג של מחוץ תווים המייצגת תעודה זהות של כרטיס ביקור
- הפעל את מטודת axios.patch והעביר בארגומנט את כתובות ה – URL שהרואית על הוספה או הסרת ליק מכרטיס ביקור וחלץ ממנו את מפתח data.
- החזר את data מהפונקציה
- אם מתקבלת שגיאה תפום אותה באמצעות מנגןן try & catch ו诙זר Promise.reject עם הודעה שהשגיאה

deleteCard •

- הֆונקציה מקבל בפרמטר cardId מסוג של מחוץ תווים המייצגת תעודה זהות של כרטיס ביקור
- הפעל את מטודת axios.delete והעביר בארגומנט את כתובות ה – URL שהרואית על מחיקת כרטיס ממגר המידע וחלץ ממנו את מפתח data.
- החזר את data
- אם מתקבלת שגיאה תפום אותה באמצעות מנגןן try & catch ו诙זר Promise.reject עם הודעה שהשגיאה



useCards

בגלל שדים רבים באפליקציה יהיו מעוניינים לקבל את כרטיסי הביקור ולהשפיע על תוכנם נוצר `Custom hook` לניהול הCARTEISIM



useCards.js

הברת הלוגיקה של ניהול הCARTEISIMS תבוצע בשלוש שלבים.
הראשון הוא ייצרת custom hook `useCards` בנתיב
`src/cards/hooks/useCards`

- ניבא את `useState` מ `react`
- ניבא את מטודת `getCards` מהשירות שיצרנו
- ניצור את המטודה `useCards`
- **בעזרת מטודה `useState` ה – hook ינהל את מצב:**

- cards – CARTEISIM הביקור
- card – CARTEISIM ביקור ספציפי
- pending request – ניהול מצב isLoading
- error – ניהול שגיאות

- ניצור את הפונקציה האсинכרונית `handleGetCards`
- שתקבע את ערכו של המשתנה `isLoading` ל – `false`
- תפעיל את מטודת `getCards` ותשמר את הערך שיחזור בתוך קבוע בשם `cards`
- נעדכן את ערכו של המשתנה `isLoading` לחיווי
- נעדכן את המשתנה `error` ל – `null`
- נעדכן את המשתנה `cards` עם מערך הCARTEISIMS שחרזר אלינו ממטודה `getCards`
- במידה והייתה שגיאה היא תיתפס במנגנון `try & catch` ונעדכן את המשתנים בהתאם

• **לבסוף ניצא את המשתנים ואת מטודה `handleGetCards`**

```
JS useCards.js 1, M X
client > src > cards > hooks > JS useCards.js > ...
1 import { useState } from "react";
2 import { getCards } from "../../services/cardService";
3
4 const useCards = () => {
5   const [cards, setCards] = useState(null);
6   const [card, setCard] = useState(null);
7   const [isLoading, setLoading] = useState(true);
8   const [error, setError] = useState(null);
9
10  const handleGetCards = async () => {
11    try {
12      setLoading(true);
13      const cards = await getCards();
14      setLoading(false);
15      setError(null);
16      setCards(cards);
17    } catch (error) {
18      setLoading(false);
19      setError(error);
20      setCards(null);
21    }
22  };
23
24  return {
25    card,
26    cards,
27    isLoading,
28    error,
29    handleGetCards,
30  };
31};
32
33 export default useCards;
```

useCards.js

בגלל שאנחנו מתוכונים לעדכן בכל הfonקציות העתידיות שלנו את המשתנים שבתור useState ואין בראצנו לרשום כל פעם את הפעלת המethodות שארחראיות על השינוי. ניצור פונקציה שתרכז לנו את ניהול המשתנים.

- ניצור משתנה בשם requestStatus שייהי שווה ערך לפונקציה שתתקבל את המשתנים הבאים
 - הפונקציה תפעיל את method setLoading עם הערך loading שיתקבל במשתנה setLoading
 - תפעיל את method setCards עם הערך שיתקבל cards במשתנה cards
 - תפעיל את method setCard עם הערך שיתקבל card במשתנה card (אם לא יתקבל ערך נקבע ערך דיפולטיבי של null)
 - נפעיל את method setError עם הערך יגיע מהמשתנה errorMessage
- בmethod handleGetCards נפעיל את method requestStatus כאשר מקבל מידע עם כרטיסי ביקור
- כאשר מקבלת שגיאה מהשרת

```
37 import { useState } from "react";
38 import { getCards } from "../../services/cardService";
39
40 const useCards = () => {
41   const [cards, setCards] = useState(null);
42   const [card, setCard] = useState(null);
43   const [isLoading, setLoading] = useState(true);
44   const [error, setError] = useState(null);
45
46   const requestStatus = (loading, errorMessage, cards, card = null) => {
47     setLoading(loading);
48     setCards(cards);
49     setCard(card);
50     setError(errorMessage);
51   };
52
53   const handleGetCards = async () => {
54     try {
55       setLoading(true);
56       const cards = await getCards();
57       requestStatus(false, null, cards); ←
58     } catch (error) {
59       requestStatus(false, error, null); ←
60     }
61   };
62
63   return {
64     card,
65     cards,
66     isLoading,
67     error,
68     handleGetCards,
69   };
70 };
71
72 export default useCards;
```

CardsPage.jsx

דוגמה לשימוש ב `useCard hook` שיצרנו

- ניבא את `useCards`
- נפעיל אותו בתוך הקומפוננט ונחלץ ממנו את הערךים שאנו מעוניינים בהם
- נפעיל את מетодת `handleGetCards` בתוך `useEffect` כדי שתישלח בקשה להבאת הלקוחות שנכתבו ב – `useCards` תפעל
- נעדכן את הגולש בהתאם למידע שמתתקבל

```
client > src > cards > pages > CardsPage.jsx > CardsPage
1 import useCards from "./../hooks/useCards";
2 import Container from "@mui/material/Container";
3 import PageHeader from "../../components/PageHeader";
4 import { useEffect } from "react";
5 import CardsFeedback from "../../components/CardsFeedback";
6
7 const CardsPage = () => {
8   const { cards, error, isLoading, handleGetCards } = useCards();
9
10  useEffect(() => {
11    handleGetCards();
12  }, []);
13
14
15  return (
16    <Container>
17      <PageHeader
18        title="Cards Page"
19        subtitle="Here you can find business cards from all categories"
20      />
21
22      <CardsFeedback
23        isLoading={isLoading}
24        error={error}
25        cards={cards}
26        onDelete={() => {}}
27      />
28    </Container>
29  );
30
31
32 export default CardsPage;
```

useCards חלק א'



Business-cards-app

צור את המethodות הבאות בנתיב `src/cards/hooks/useCard.js`

`handleGetCard`

- הפקציה תקבל בפרמטר `cardId` מסוג של מחרוזת תווים שמייצגת תעודה זהה של כרטיס ביקור לעסק
- היא תנסה את ערכו של המשתנה `isLoading` ל – `false`
- צור קבוע בשם `card` שערך יהיה הפעלת המethodה `getCard` האסינכרונית והעבר לה בארגומנט את `cardId`
- הפעל את מטודת `requestStatus` כך שתעדכן את המפתחות באובייקט `state` שהטעינה הסתיימה, אין הודעה שגיאה, אין כרטיסים ועדיין את המפתח שאחראי על הkartis הבודד בערך של הקבוע `card` שיצרת
- החזר את הקבוע `card` שיצרת מהפקציה
- במידה ומתקבלת שגיאת תפוא אותה באמצעות מנגנון `try & catch` והפעל את מטודת `requestStatus` כך שתעדכן את הגולש בשגיאת

שימוש בחلك ב'



handleGetMyCards •

- הפונקציה תשנה את ערכו של המשתנה isLoading ל – false
- צור קבוע בשם cards שערך יהיה הפעלת המטודה האсинכרונית getMyCards
- הפעיל את מטודת requestStatus כר שתעדכן את ה – state שהטעינה הסטיימה, אין הודעת שגיאה, ותעדכן את המשתנה שאחראי על כרטיסים במידע שהתקבל בקבוע שיצרת במידה ומתקבלת שגיאה תפוס אותה באמצעות מנגןן & try catch והפעיל את מטודת requestStatus כר שתעדכן את הגולש בשגיאה

handleCreateCard •

- הפונקציה תקבל בפרמטר cardFromClient מסוג של אובייקט המייצג כרטיס ביקור של עסק
- היא תשנה את ערכו של המשתנה isLoading ל – false
- צור קבוע בשם card שערך יהיה הפעלת המטודה האсинכרונית createCard והעביר לה כארגומנט את cardFromClient
- הפעיל את מטודת requestStatus כר שתעדכן את ה – state שהטעינה הסטיימה, אין הודעת שגיאה, אין כרטיסים ותעדכן את המשתנה שאחראי על כרטיס במידע שהתקבל בקבוע שיצרת cardFromClient
- במידה ומתקבלת שגיאה תפוס אותה באמצעות מנגןן & try catch והפעיל את מטודת requestStatus כר שתעדכן את הגולש בשגיאה

שימוש בחلك ג'



handleUpdateCard •

- הֆונקציה תקבל בפרמטר `cardFromClient` מסוג של אובייקט המציג כרטיס ביקור של עסק
- היא תנסה את ערכו של המשתנה `isLoading` ל – `false`
- צור קבוע בשם `card` שערך יהיה הפעלת המטודה האסינכרונית `editCard` והעביר לה ארגומנט `cardFromClient`
- הפעיל את מטודת `requestStatus` כר שיתעדן את ה – `state` שהטינה הסטיימה, אין הودעת שגיאה, אין כרטיסים ותעדן את המשתנה שאחראי על כרטיס במידע שהתקבל בקבוע `card` שיצרת
- במידה ומתקבלת שגיאה תפום אותה באמצעות מנגןון & `try` `catch` והפעיל את מטודת `requestStatus` כר שיתעדן את הגולש בשגיאה

handleDeleteCard •

- הֆונקציה תקבל בפרמטר `cardId` מסוג של מחרוזת תווים המציגת תעודה זהות של כרטיס ביקור לעסק
- היא תנסה את ערכו של המשתנה `isLoading` ל – `false`
- הפעיל את המטודה האסינכרונית `deleteCard` והעביר לה ארגומנט את `cardId`
- במידה ומתקבלת שגיאה תפום אותה באמצעות מנגןון & `try` `catch` והפעיל את מטודת `requestStatus` כר שיתעדן את הגולש בשגיאה

useCards חלק ד'



handleLikeCard •

- הפקציה תקבל בפרמטר cardId מסוג של מחרוזת תווים המיצגת תעודה זהות של כרטיס ביקור לעסן
- היא תנסה את ערכו של המשתנה isLoading ל – false
- צור קבוע בשם card שערך יהיה הפעלת המתודה האсинכרונית changeLikeStatus והעביר לה ארגומנט את cardId
- צור קבוע בשם cards שערך יהיה הפעלת המתודה getCards האсинכרונית
- הצל את מתודת requestStatus כך שתעדכן את ה – state שהטינה הסתיימה, אין הודעת שגיאה, עדכן את המשתנה שאחראי על הCARDSים במידע שמתתקבל מהקבוע cards שיצרת, ותעדכן את המשתנה שאחראי על CARDS במידע שהתקבל בקבוע card שיצרת
- במידה ומתקבלת שגיאה תפוס אותה באמצעות מגנון try & catch והצל את מתודת requestStatus כך שתעדכן את הגולש בשגיאה
- יצא את כל המתודות חדשות מהמודול



axios interceptors

ירוט האובייקטים של `request` ו-`response` על ידיו של axios לפני ואחרי שליחת בקשה `http` לשרת ואפשרות לבצע שינויים באובייקטים אלו / או להפעיל לוגיקה בהתאם לערכיהם

<https://axios-http.com/docs/interceptors>



useAxios.js

בנתיב js.js src/hooks/useAxios.js

```
JS useAxios.js U X
client > src > hooks > JS useAxios.js > ...
1   import axios from "axios"; ←
2
3   const useAxios = () => {
4     axios.interceptors.request.use(data => { ←
5       console.log("in useAxios request interceptor");
6       return Promise.resolve(data);
7     }, null); ←
8
9   axios.interceptors.response.use(
10    data => {
11      console.log("in useAxios response interceptor");
12      return Promise.resolve(data);
13    },
14    error => {
15      console.log("in useAxios response interceptor");
16      return Promise.reject(error);
17    }
18  );
19}
20
21 export default useAxios;
```

- נייבא את מטודת axios.interceptors.request.use עם שני ארגומנטים:
- פונקציית call back ש – axios תשפוך אליה את אובייקט ה – request (פעיל את מטודת request עם מחזורת תווים כדי לסמן שאין במצבים בחלק זה של הקוד ונחזירPromise.resolve(data) כדי שהבקשה תעבור להלאה אל השרת גם בארגומנט השני null !
- אם עביר פונקציה בארגומנט השני, במידה ותהייה שגיאה axios תשפוך את אובייקט השגיאה של הזאת ותשפוך את אובייקט השגיאה של axios אל הארגומנט שלו. לאחר מכן נctrן להעביר להלאה את השגיאה באמצעותPromise.reject : axios.interceptors.response.use
- גם היא מקבלת שני ארגומנטים כאשר אל הראשון היא תשפוך את אובייקט ה – response ובשי את אובייקט השגיאה וגם פה ניתן ח'ו' שאמנו בחלק הזה בקוד ונעביר להלאה את הבקשה בעזרת מחלקהPromise

useCards.js

בגלל על מנת שנוכל לראות את axios interceptors שיצרנו מתוך useCard hook מידע מה קומפוננט שתצרוך ל- axios – ה- interceptors תפעיל את ה-

- ניבא את useAxios
- נפעיל את מетодת useAxios בתוך ה- useCards של hook – scope

! להזכיר ייתן להפעיל hook רק בתוך קומפוננט מסווג פונקציה או hook אחר

```
JS useCards.js M ×  
client > src > cards > hooks > JS useCards.js > ...  
1 import { useState } from "react";  
2 import {  
3   getCards,  
4   getCard,  
5   deleteCard,  
6   createCard,  
7   editCard,  
8   getMyCards,  
9   changeLikeStatus,  
10 } from "../../services/cardService";  
11 import useAxios from "../../hooks/useAxios"; ←  
12  
13 const useCards = () => {  
14   const [cards, setCards] = useState(null);  
15   const [card, setCard] = useState(null);  
16   const [isLoading, setLoading] = useState(true);  
17   const [error, setError] = useState(null);  
18  
19   useAxios(); ←
```

The screenshot shows a web-based application for managing business cards. At the top, there's a header with a user icon, a search bar, and a menu icon. Below the header, the word "Cards" is prominently displayed. A sub-header below it says "Here you can find business cards from all categories". On the left, there's a large thumbnail image of a business card featuring a line graph on a grid background. To the right of the thumbnail, the text "first card" is displayed, followed by the subtitle "this is the first card". Underneath, there are four lines of contact information: "Phone: 050-0000000", "Address: test 3 test", and "Card Number: 2249541". At the bottom of the card view, there are three navigation icons: "About" (with an info icon), "Favorites" (with a heart icon), and "My Cards" (with a user icon).

The screenshot shows a browser developer tools console window. It displays a warning message: "[Violation] 'message' scheduler.development.js:517 handler took 233ms". Below this, two entries are listed under "in useAxios request interceptor": one at line 5 and another at line 11. There is also a small arrow pointing downwards.

התווצה בדף

במידה והميدע מגיע ניתן לראות ש axios interceptors הפעלו ולפניהם שליחת הבקשה הדפיסו בקונוסל את מחרוזת התווים הבאה

The screenshot shows a web application interface. At the top, there is a header with a user icon, a search bar containing 'Search', and a menu icon. Below the header, the word 'Cards' is displayed in large letters. Underneath 'Cards', a sub-header says 'Here you can find business cards from all categories'. A message below states 'Oops... something went wrong: Request failed with status code 404'. At the bottom left is a cartoon illustration of a robot holding a wrench.

The screenshot shows a browser's developer tools console. The title bar includes 'Console', '2 issues', and '1 hidden'. The main area shows a list of errors under '1 Issue: 1':

- [Violation] scheduler.development.js:517 'message' handler took 181ms
- ④ in useAxios request interceptor useAxios.js:5 ←
✖ ▶ GET http://localhost:8181/cards xhr.js:247 ↗
fff 404 (Not Found)
- ② in useAxios request interceptor useAxios.js:15 ←
✖ ▶ GET http://localhost:8181/cards xhr.js:247 ↗
fff 404 (Not Found)
- ② in useAxios request interceptor useAxios.js:15 ←
▶

התוצאות בדף

במידה שיש שגיאה

- אז קודם כל הבקשה מירטת על ידי axios interceptors
- וכשחזרת תשובה עם שגיאה גם היא מירטת

axios משימת interceptors



Business-cards-app

צור את המethodות הבאות בנתיב `src/cards/hooks/useCard.js`

`handleGetCard` •

- הפקציה תקבל בפרמטר `cardId` מסוג של מחרוזת תווים שמייצגת תעודת זהות של כרטיס ביקור לעסק
- היא תנסה את ערכו של המשתנה `isLoading` ל-`false`
- צור קבוע בשם `card` שערך יהיה הפעלת המethode `getCard` האסינכרונית והעבר לה בארגומנט את `cardId`
- הפעל את מетодת `requestStatus` כך שתעדכן את המפתחות באובייקט `state` – שהטעינה הסתיימה, אין הודעה שגיאה, אין כרטיסים ועדכן את המפתח שאחראי על הkartis הבודד בערך של הקבוע `card` שיצרת
- במידה ומתקבלת שגיאה תפום אותה באמצעות מנגןן `try & catch` והפעל את מетодת `requestStatus` כך שתעדכן את הגולש בשגיאה

Context

"Context provides a way to pass data through the component tree without having to pass props down manually at every level"

reactjs.org

<https://reactjs.org/docs/context.html>





When to use context

Data that can be considered “global”

Themes

User data (current authenticated user)

Location-specific

preferred language

Using React Context

Create Context

By using the `React.createContext` method

Wrap the component tree
with `context.Provider`

Use the prop value on the
`context.Provider` to pass data
that you want to share

*Use `context.Consumer` to get
the shared value*

If you transfer an object and
inside, it a function that can
affect the value of the prop
value in `context.Provider` you
can use it to change the value
from the component tree

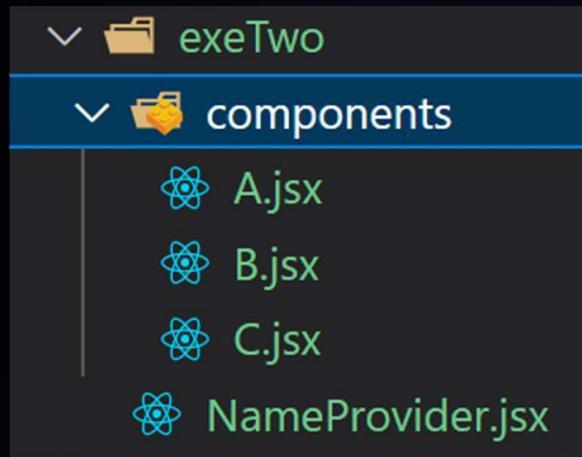


example

דוגמא ליצירה ושימוש ב - context



הכנות תשתיית



• נוצר את הנתיב `exeTwo`

ובתוכו את הקובץ:

`NameProvider.jsx` •

• את התקינה `components` שבתוכה הקבצים:

`A.jsx` •

`B.jsx` •

`C.jsx` •

NameProvider.jsx

```
client > src > sandbox > use-context > exeTwo > NameProvider.jsx > ...
1 import React, { useState, useEffect, useContext } from "react";
2 import { node } from "prop-types";
3
4 const NameContext = React.createContext(null); ←
5 // NameContext.displayName = "ProviderName";
6
7 export const NameProvider = ({ children }) => { ←
8   const [name, setName] = useState(); ←
9
10  useEffect(() => { ←
11    setName("david");
12  }, [ ]); ←
13
14  return (
15    <NameContext.Provider value={{ name, setName }}> ←
16      {children} ←
17    </NameContext.Provider>
18  );
19};
20
21 export const useName = () => { ←
22   const context = useContext(NameContext);
23   if (!context) throw new Error("useName must be used within a NameProvider");
24   return context;
25 };
26
27 NameProvider.propTypes = {
28   children: node.isRequired,
29 };
```

- בדוגמה הבאה אציגים איך ניתן להביר לעצם הקומפוננטות מטודה שתשנה את ערכו של Context.Provider ב- המשנה המועבר ב-
- ניצור קבוע בשם NameContext ונשווה את ערכו לערך שיכון מהפעלת מטודה React.createContext
- ניצור ונייצא קבוע בשם NameProvider שערכו יהיה פונקציה שתחלץ מאובייקט ה-props את המילה השמורה children
- נפעיל את מטודת useState ונחלץ ממנה את setName
- נפעיל את מטודת useEffect פעם אחת שתעדכן את הערך של name עם מחזורת התווים
- נחזיר מהפונקציה את ה- Provider של NameContext וגעיר לו בערך גם את המשנה name וגם את המטודה שאחראית לשינוי ערכו setName
- ניצור ונייצא את הקבוע useName שערכו יהיה פונקציה אנונימית ש:

 - תיצור קבוע בשם context שערכו יהיה הפעלת מטודת useContext וגעיר לה בפרמטר את NameContext
 - תבדוק האם הערך של context הוא פוליסיבי תזרוק שגיאה
 - אם הערך של context הוא חיובי היא תחזיר אותו (כלומר תחזיר את האובייקט שהעבರנו ב- value (במאפיין NameContext.Provider)

```
❖ A.jsx  M X
client > src > sandbox > use-context > exeTwo > components > ❖ A.jsx
1 import B from "./B";
2 import { NameProvider } from "../NameProvider";
3
4 const A = () => {
5   return (
6     <NameProvider> <
7       <B />
8     </NameProvider>
9   );
10};
11
12 export default A;
```

A.jsx

נ鼬ף את הקומפוננט B בקומפוננט
NameProvider
מהקומפוננט A

B.jsx

```
client > src > sandbox > use-context > exeTwo > components > B.jsx
1 import React from "react";
2 import C from "./C";
3 import { useState } from "../NameProvider"; ←
4
5 const B = () => {
6   const { name } = useState(); ←
7   return (
8     <>
9       <p>name in B: {name}</p> ←
10      <C /> ←
11    </>
12  );
13};
14
15 export default B;
```

- ניבא את useState שיצרנו
- בתוך הקומפוננט B
 - נפעיל את METHOD useState ונקחץ ממנו את המשתנה name
 - נציג לגולש שהוא נמצא בקומפוננט B עם ערכו של המשתנה name
 - ונציג את קומפוננט C

C.jsx

C.jsx

client > src > sandbox > use-context > exeTwo > components > C.jsx > ...

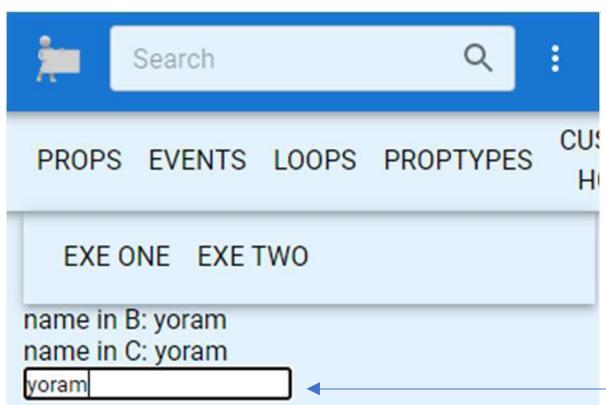
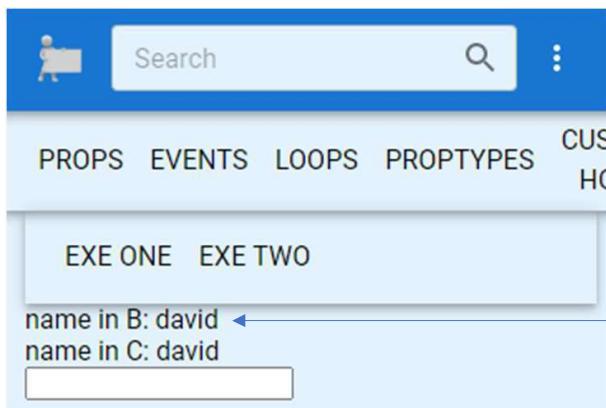
```
1 import React from "react";
2 import { useState } from "../NameProvider"; ←
3
4 const C = () => {
5   const { name, setName } = useState(); ←
6   return (
7     <>
8       <div>name in C: {name}</div> ←
9       <input type="text" id="name" onChange={e => setName(e.target.value)} /> ←
10      </>
11    );
12  };
13
14 export default C;
```

- ניבא את useState שיצרנו
- בתוך הקומponent C

- פעיל את方法ת useState ווחלץ ממנה את המשתנה name ואתmethodה setName
- נציג לגולש שהוא נמצא בקומponent C עם ערכו של המשתנה name
- נוצר אלמנט מסוג input שכל שינוי בו יפעיל אתmethodת setName שჩילנו מ – useState ותשנה את ערכו של name בהתאם לכתוב באלמנט (e.target.value)

התוצאה בדף

- כאשר קוראים לkomponent useEffect נכנס לפעולה ומעדכן את הערך של name ל – david
- כאשר כתבים אלמנט input שבkomponent C כל שאר הקומponentות מתעדכנות בערך



משימת חלק א'



! המשך המשימה בעמוד הבא

Business-cards-app

- צור את הlient `src/providers/ThemeProvider.jsx`
- יבוא לתוך המודול את `React useState useContext`
- יבא את `createTheme` `ThemeProvider` `as MuiThemeProvider`
- (כלומר לחליף לקומפוננט `ThemeProvider` של `MUI` את השם כדי שהשם לא יתנגש עם השם של הקומפוננט)
- צור קבוע בשם `ThemeContext` שערךו יהיה שווה להפעלת מטודת `React.createContext`
- צור ייצא את הקומפוננט `ThemeProvider` שתתקבל `children` מסווג `node` כמפתח באובייקט הפרויקט
- הפעיל את מטודת `useState` עם הערך `false` וחילץ ממנו את `isDark` `setDark`
- צור קבוע בשם `toggleDardMode` שייהי שווה ערך לפונקציה אונונימית שתפעיל את מטודת `setDark` ותשנה את ערכו של המשתנה `dark` לערך הנגדי שלו
- צור קבוע בשם `theme` שערךו יהיה שווה להפעלת מטודת `createTheme` שייבנו מ - `MUI` עם אובייקט `Konfigurazioni` כפרמטר, כאשר אנו משים את המפתח `palette` לאובייקט שהמפתח `mode` בתוכו יהיה שווה למחוזות התווים "dark" במידה והמשתנה `isDark` אינו פוליסיבי, אחרת ערכו יהיה שווה ערך למחוזות התווים "light"
- החזר מהקומפוננט את הילדים שהקומפוננט קיבלה עטופים ב - `ThemeContext.Provider` עם הערך של המשתנה `isDark` זו ומטודה `MuiThemeProvider` `toggleDardMode` שגם היא תהיה עוטפה עם `isDark` כאשר העבר למפתח `theme` באובייקט הפרויקט שלה את `theme`

שימוש ב'Context



! המשך המשימה בעמוד הבא

- עדין בנתיב `src/providers/ThemeProvider`
- צור ויצא קבוע בשם `useTheme` שיהיה שווה ערך לפונקציה אוניברסלית

- שטיוצר קבוע בשם `context` שערך יהיה שווה ערך להפעלת מטודת `useContext` בפרמטר `ThemeContext`
- הmethodת תבדוק אם יש ערך פולסיבי קבוע `context` היא תזרוק `ThemeProvider` שגיאה כי `useTheme` חייב להיות בתוך `ThemeProvider`
- בודק באמצעות סדרית `PropTypes` שהקומפוננט קיבלת את מה שהיא צריכה כדי לפעול כהכליה.

בונוס

- העבר תהליך של Memoization למשתנה `isDark` לפונקציה `toggleDarkMode`

בקובץ `src/App.js`

- יבא את `ThemeProvider`
- עוטף את הקומפוננט `Layout` בה
- בתפריט הנויוט העליון ימני `useTheme`
- יבא את `useTheme` וחלץ ממנו את המשתנים `isDark` `toggleDarkMode`
- צור כפתור `IconButton` שלחיצה עליו תפעיל את מטודת `toggleDarkMode`
- צור התניה בתוך הכפתור כך שם ערכו של המשתנה `isDark` הוא לא פולסיבי הפתור יראה את הקומפוננט `LightModelcon` של MUI אחריו הוא יראה את הקומפוננט `DarkModelcon`

משימת Context

חלק ג'



- בנתיב `src/layout/main/Main.js`
- ייבא את מتدת `useTheme` שיצרנו בשקפים הקודמים
- הפעיל את מتدת `useTheme` וחלץ ממנה את המשתנה `isDark`
- החלף את המיכל ל – MUI Paper
- בהגדרות העיצוב קבוע שם למשתנה `isDark` יש ערך פולסיבי צבע הרקע יהיה "#e3f2fd" אחרית ערכו יהיה "#333333"

משימת התוצאה בדף

The image displays two side-by-side screenshots of a web application interface for managing business cards. The top screenshot shows the interface in light mode, and the bottom screenshot shows it in dark mode. Both screenshots feature a header with the logo 'BCard' and navigation links for 'ABOUT', 'MY CARDS', 'FAV CARDS', 'SANDBOX', a search bar, and 'SIGNUP' and 'LOGIN' buttons. Below the header, the word 'Cards' is displayed, followed by the sub-instruction 'Here you can find business cards from all categories'. Three business cards are listed, each with a thumbnail image of a stock market chart, a title ('first card', 'second card', 'third card'), a subtitle ('this is the first card', etc.), and a detailed section containing placeholder contact information: 'Phone: 050-0000000', 'Address: test 3 test', and 'Card Number: 2249541' for the first card, and similar entries for the second and third cards. Each card also includes edit, delete, and favorite icons. At the bottom of each screenshot, there are three buttons: 'About', 'Favorites', and 'My Cards'. The dark mode screenshot has a black background and white text, while the light mode screenshot has a white background and black text.



Snackbar

מתן חיוני ויזואלי לגולש על הצלחה/ כישלון
בפעולות ה - CRUD



SnackbarProvider.jsx

ניצור את הנתיב `src/providers/SnackbarProvider.jsx`

- ניצור קבוע בשם `SnackbarContext` ונשווה את ערכו לערך שיחזור `React.createContext(null)` מהפעלת `createContext`
- ניצור וニיצא קבוע בשם `SnackbarProvider` שערכו יהיה פונקציה `setSnack` מוצelix מאובייקט ה- `props` את המילה `children` השומרה ממנה את `children`
- נפעיל את `useState` ארבע פעמים ונחלץ ממנה את המשתנים והметודות הבאות
- ניצור את `setSnack` שערך יהיה הפונקציה שתבצע מהפעלת `useCallback` כאשר בפונקטר הראשון מקבל פונקציה אונומית ש:

- מקבל בפונקטר את המשתנים `color message variant`
- תקבע את ערכו של הקבוע `isSnackOpen` ל- `true` על מנת שהקומפוננט `Snackbar` של MUI תופיע במקום שנגדר לה.
- תנסה את צבע הקומפוננט `Snackbar` לצלבע שיתקבל בהפעלת הפונקציה
- תנסה את ההודעה שתוצג לגולש להודעה שתתקבל בהפעלת הפונקציה
- תנסה את תצורת הקומפוננט או לערך הדיפולטיבי "filled" או לערך אחר שיתקבל בהפעלת הפונקציה

- ובפונקטר השני מעניק של תלויות עם שמות המתוודות לשינוי `state` המשתנים ב-

הקומפוננט תצהיר:

- את הקומפוננט `Snackbar` עם העיצובים שקבעם `setSnack`
- ואת הקומפוננט `SnackbarContext.Provider` עם הערך של `setSnack` קיבל שועופת את הקומפוננטותשה – `Provider`

- בעזרה ספרית `propTypes` נבדוק שהקומפוננט אכן קיבל `children`

The diagram illustrates the prop flow in the `SnackbarProvider` component. It starts with the component definition at the top, followed by its body which contains several `const` declarations and a `return` block. The `return` block contains two main parts: a `Snackbar` component and a `SnackbarContext.Provider`. The `Snackbar` component receives `isSnackOpen`, `onClose`, `autoHideDuration`, `severity`, and `variant` props. The `SnackbarContext.Provider` component receives the `setSnack` function as its `value`. A large curly brace on the right side groups all these props together, indicating they are passed down to the `children` of the provider. Below the provider, another curly brace groups the `children` of the provider itself, which are the `Alert` and `Snackbar` components. Arrows point from the `setSnack` prop to the `setSnack` in the `Snackbar` component, and from the `severity` and `variant` props to their respective properties in the `Alert` component.

```
client > src > providers > SnackbarProvider.jsx > ...
1 import React, { useState, useContext, useCallback } from "react";
2 import { node } from "prop-types";
3 import Alert from "@mui/material/Alert";
4 import Snackbar from "@mui/material/Snackbar";
5
6 const SnackbarContext = React.createContext(null);
7
8 export const SnackbarProvider = ({ children }) => {
9   const [isSnackOpen, setOpenSnack] = useState(false);
10  const [snackColor, setSnackColor] = useState("success");
11  const [snackMessage, setSnackMessage] = useState("in snackbar!");
12  const [snackVariant, setSnackVariant] = useState("filled");
13
14  const setSnack = useCallback(
15    (color, message, variant = "filled") => {
16      setOpenSnack(true);
17      setSnackColor(color);
18      setSnackMessage(message);
19      setSnackVariant(variant);
20    },
21    [setOpenSnack, setSnackColor, setSnackMessage, setSnackVariant]
22  );
23
24  return (
25    <>
26      <Snackbar
27        anchorOrigin={{ vertical: "top", horizontal: "right" }}
28        open={isSnackOpen}
29        onClose={() => setOpenSnack(prev => !prev)}
30        autoHideDuration={6000}
31        <Alert severity={snackColor} variant={snackVariant}>
32          {snackMessage}
33        </Alert>
34      </Snackbar>
35
36      <SnackbarContext.Provider value={setSnack}>
37        {children}
38      </SnackbarContext.Provider>
39    </>
40  );
41};
42
43 SnackbarProvider.propTypes = {
44   children: node.isRequired,
45};
```

SnackbarProvider.jsx

המשך הקוד של המודול מהשקייף הקודם

- ניצור ונייבא קבוע בשם `useSnackbar` שערך יהיה פונקציה אוניברסלית ש:

- ניצור קבוע בשם `context` ונשווה את `useContext` להפעלת מטודת `SnackbarContext` ועביר לה אגרומנט את `SnackbarContext`

- נתנה ואם לקבוע `context` שייצרנו יש ערך פולסיבי נעצור ונזורך שגיאה על כך שיש להשתמש בMETHOD רק בקומפוננטות בנים של הקומפוננט `SnackbarProvider`
- אם לא נכנסנו לתנינה המשמעות היא שיש ערך לקבוע `context` שייצרנו ואני מחזיר אותו מהפונקציה

```
43 export const useSnackbar = () => {
44   const context = useContext(SnackbarContext); ←
45   if (!context) ←
46     throw new Error("useSnackbar must be used within a SnackbarProvider");
47   return context;
48 }
```

JS App.js

```
client > src > JS App.js > ...
1  import "./App.css";
2  import Layout from "./layout/Layout";
3  import { BrowserRouter } from "react-router-dom";
4  import Router from "./routes/Router";
5  import { ThemeProvider } from "./providers/ThemeProvider";
6  import { SnackbarProvider } from "./providers/SnackbarProvider";
7
8  function App() {
9    return (
10      <div className="App">
11        <BrowserRouter>
12          <ThemeProvider>
13            <SnackbarProvider> ←
14              <Layout>
15                <Router />
16              </Layout>
17            </SnackbarProvider>
18          </ThemeProvider>
19        </BrowserRouter>
20      </div>
21    );
22  }
23
24  export default App;
```

App.js

- ניבא את SnackbarProvider
- ניצור נטעוף את האפליקציה בקומפוננט SnackbarProvider שיצרנו

useAxios.js

```
23 import axios from "axios";
24 import { useSnackbar } from "../providers/SnackbarProvider"; ←
25
26 const useAxios = () => {
27   const snack = useSnackbar(); ←
28
29   axios.interceptors.request.use(data => {
30     return Promise.resolve(data);
31   }, null);
32
33   axios.interceptors.response.use(null, error => { ←
34     const expectedError = error.response && error.response.status >= 400;
35     if (expectedError) snack("error", error.message); ←
36     return Promise.reject(error); ←
37   });
38 };
39
40 export default useAxios;
```

שימוש בkomponenot Snackbar שיצרנו

- ניבא את מטודת useSnackbar שיצרנו
- ניצור קבוע בשם snack שערך יהיה useSnackbar מטודה
- השתמש ב – snack בתוך מטודה axios.interceptors.response
 - במידה ומתקבלת שגיאה מהשרת שהסתטואס קוד שלה הוא 400 ומעלה
 - נפעיל את snack שיצבע אותו באדום ובפרמטר הראשון של הצבע שיצבע אותו באדום ובפרמטר השני את הودעת השגיאה מ – axios
- לבסוף נחזיר Promise.reject עם השגיאה כדי שהkomponenot/hook שצרך את הלוגיקה שב useAxios יוכל לעדכן את הגולש בצורה נוספת מלבד ה – snackbar -

Snackbar חלק א'



Business-cards-app

- במודול `useCards.js`
- יבא את המטודה `useSnackbar`
- צור קבוע בשם `snack` שערך יהיה הפעלת מטודת `useSnackbar`
- הוסיף למטרות הבאות בחלק של הצלחת פעולה ה – CRUD את הפעלת מטודת `snack` כאשר הפרמטר הראשון שלו יהיה "success" ובפרמטר השני שלה הודעה שמתאימה לפעולה ה – CRUD שבוצעה בהצלחה:
 - `handleCreateCard`
 - `handleUpdateCard`
 - `handleDeleteCard`