

React

A JavaScript Library for building user interfaces

Created By David Yakin 2023

תוכן טנינים

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

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

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 את ה - **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.

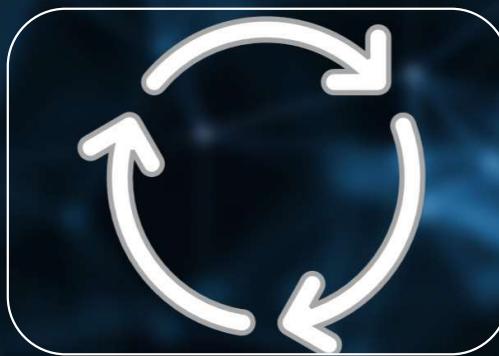
<https://reactjs.org/docs/faq-internals.html>



Virtual DOM implementations



State



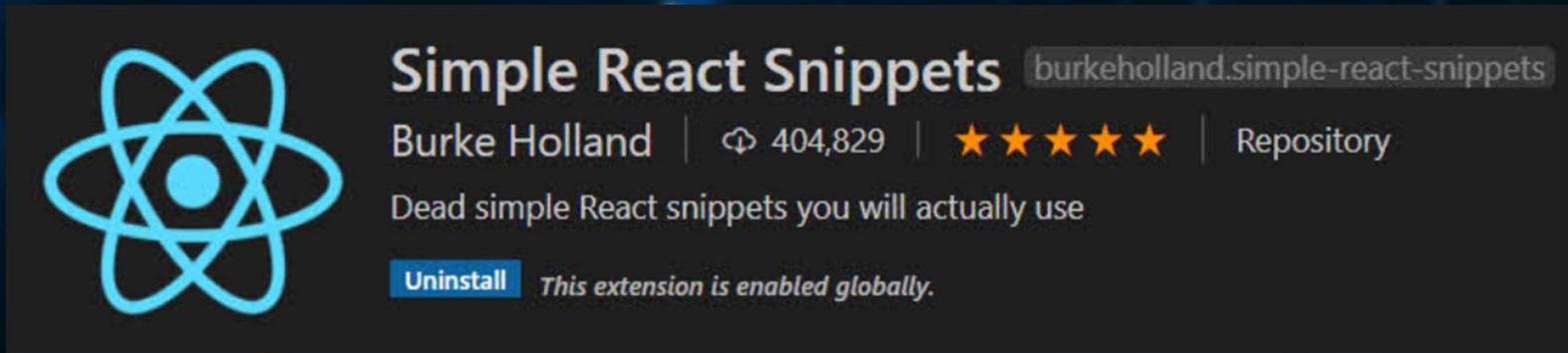
Life cycle
hooks



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

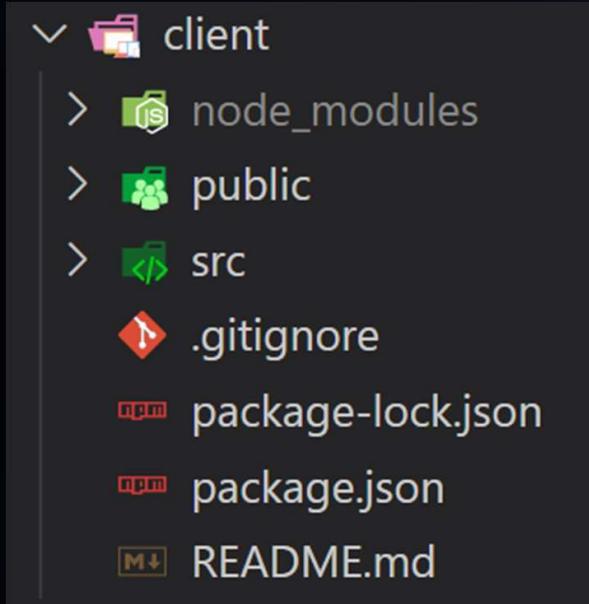
- נכנס לתוך התקינה בה אנו מעוניינים ליצור פרויקט חדש של React

• נקיש בטרמינל את הפקודה create-react-app client

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

תשתיית הקבצים שה - CLI יצר

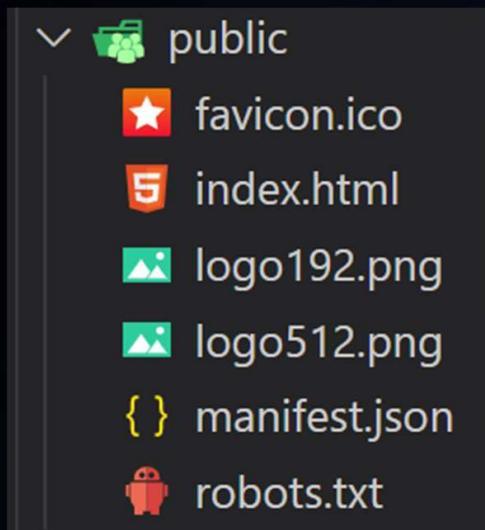
- **Node_modules** – תיקייה השומרת בתוכה את הספריות שהורدنנו לפרויקט
- **public** – תיקייה השומרת את הקבצים הסטטיסיים
- **src** – תיקייה בה רוב הקוד של הפרויקט יכתב
- **.gitignore** – קובץ קונפיגורציות של git ובתוכו הוראות מיילותיקיות וקבצים להタルם ולא להעלות ל github
- **package.json** – קובץ קונפיגורציות הכלל 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 שרייקט משתמש 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 txt
<https://www.youtube.com/watch?v=fzm-zYHlgY>



index.html

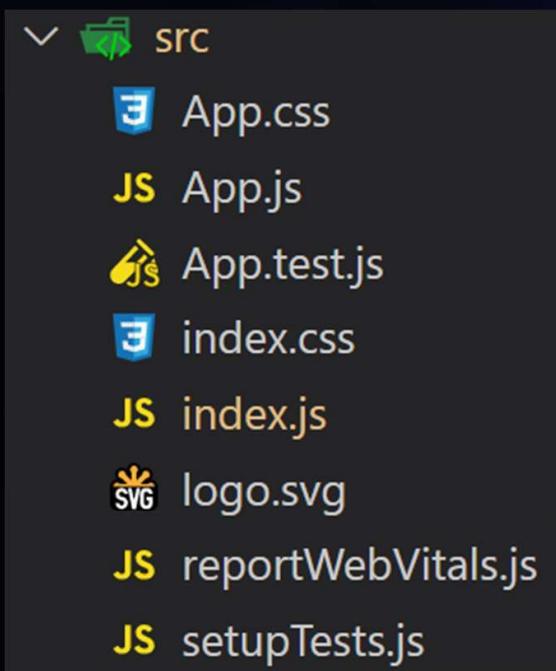
- קישור למיקום תמונה ה – favicon
- קישור למיקום התמונה ל – apple
- קישור מיקום קובץ manifest.json
- כותרת האפליקציה
- האלמנט אליו יזרקו כל שאר הקומponentות

```
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=00RoZflFE34>



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

- יבוא קובץ ה – logo לkomponent
- יבוא קובץ העיצוב לkomponent
- יצירת komponent App
- התצוגה לגולש (מה שהפונקציה מחזירה)

index.js

קובץ הלוגיקה המרכזי של האפליקציה

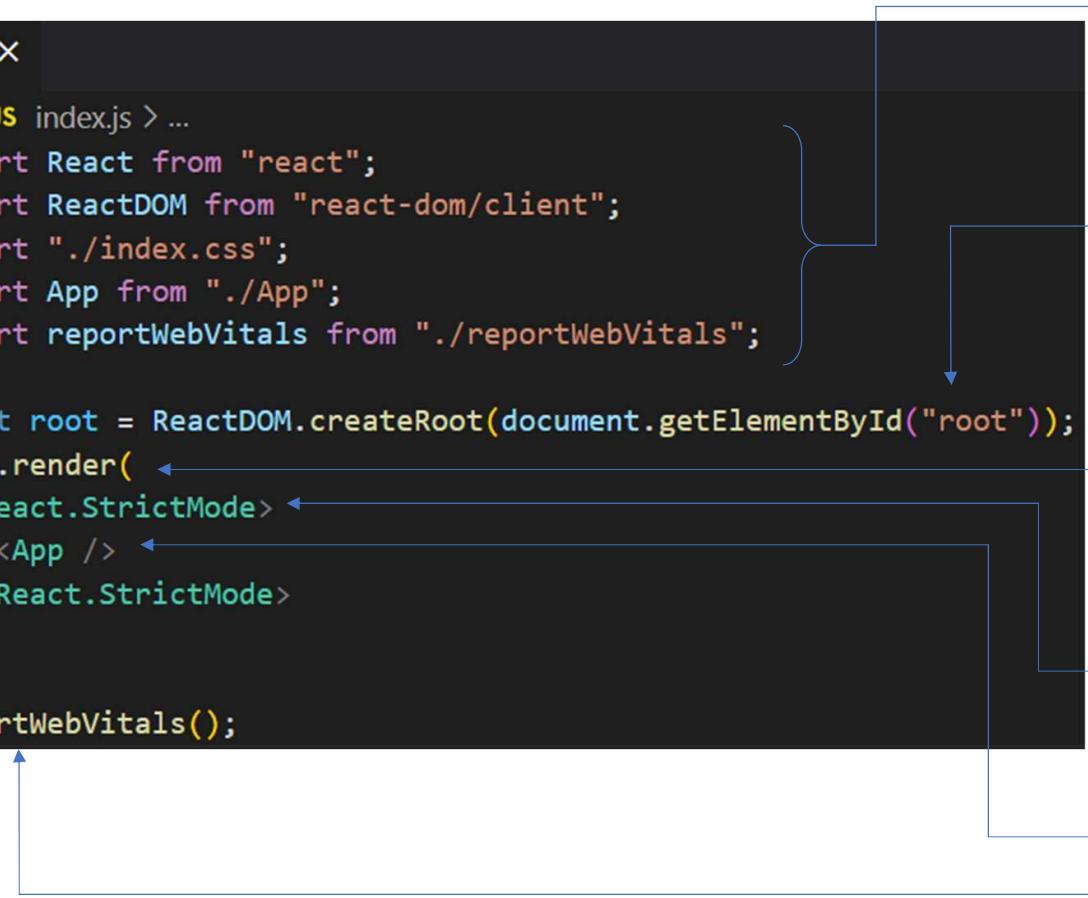
- **יובא מודולים:**
 - מופע מהספרייה React לkomponent
 - קובץ העיצוב של komponent
 - komponent App בדיקות האופטימיזציה לkomponent
- **יצירת קבוע בשם root שערך שווה ערך להפעלה מטודת ReactDOM.createRoot שתתפוך את האלמנט זהה נמצאת בתוך הקובץ (אלמנט זהה נמצא בתיקיה index.html) index.html**
 - הפעלה מטודת root.render אשר תזירק לתוך האלמנט שתפסנו root את הקומונוטות (ידובר בהרחבת בהמשך)
 - עטיפת האפליקציה בkomponent של React strictMode שייכפה על הקוד שייכתב בתוכו (ידובר על כר בהרחבת בהמשך המציגת)
 - הצבת komponent App כר שתוצג לגולש reportWebVitals()
 - הפעלה 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 השירות הזמן של ריאקט ונוכל לראות את התצוגה הראשונית של האפליקציה



npm start

נפעיל את השרת הזמני של React
שגם יאזור לשינויים בקוד ויתן לנו
תצוגת אמת של הקוד שלנו

- נכנס לתיקייה הרלוונטייה בה נמצא פרויקט ה – React

• וקליד את הפקודה npm start

- ה – CLI יודיע לנו ובמידה ולא תהיה שגיאה בתהליך compile נראה את הכתוב הבא

- ה – CLI יעדכן אותנו כי אנו מازינים 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 הרשמיים

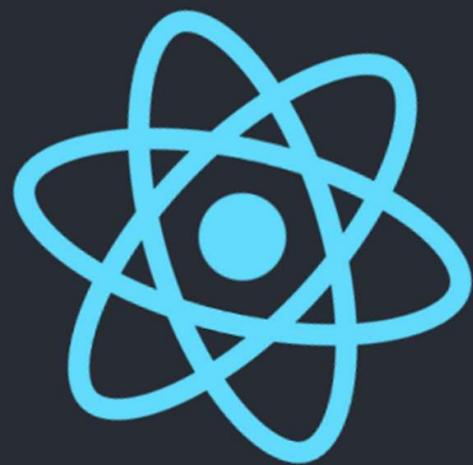
```
Compiled successfully!
```

```
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.

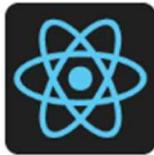
[Learn React](#)

התוצאה בדף

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

React Developer Tools

כלי שיעזור לנו בשלבי הפיתוח ב - React



React Developer Tools

Featured

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

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



Components

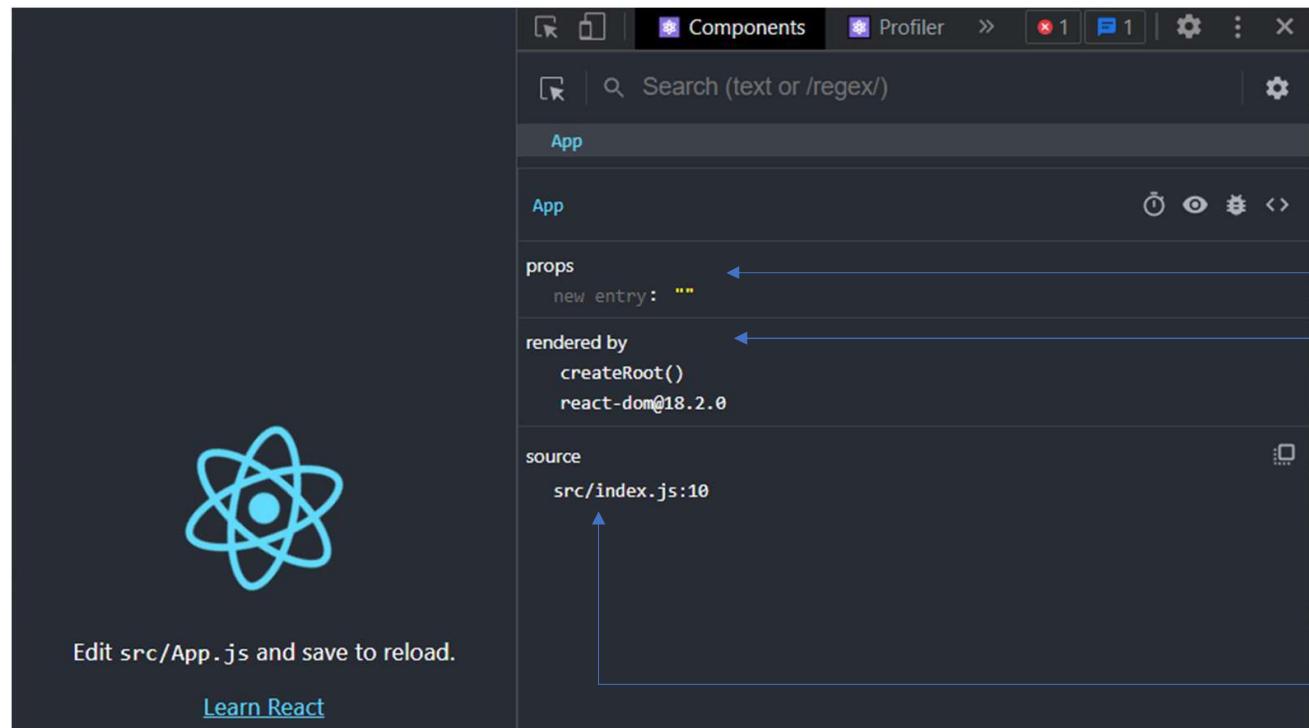
לשונית זאת מראה לנו נתונים נוספים על קומponent נבחרת

- 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 – הפונקציה שאחראית על טעינה וטעינה מחדש של הקומponent
- source – קוובץ המקור

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



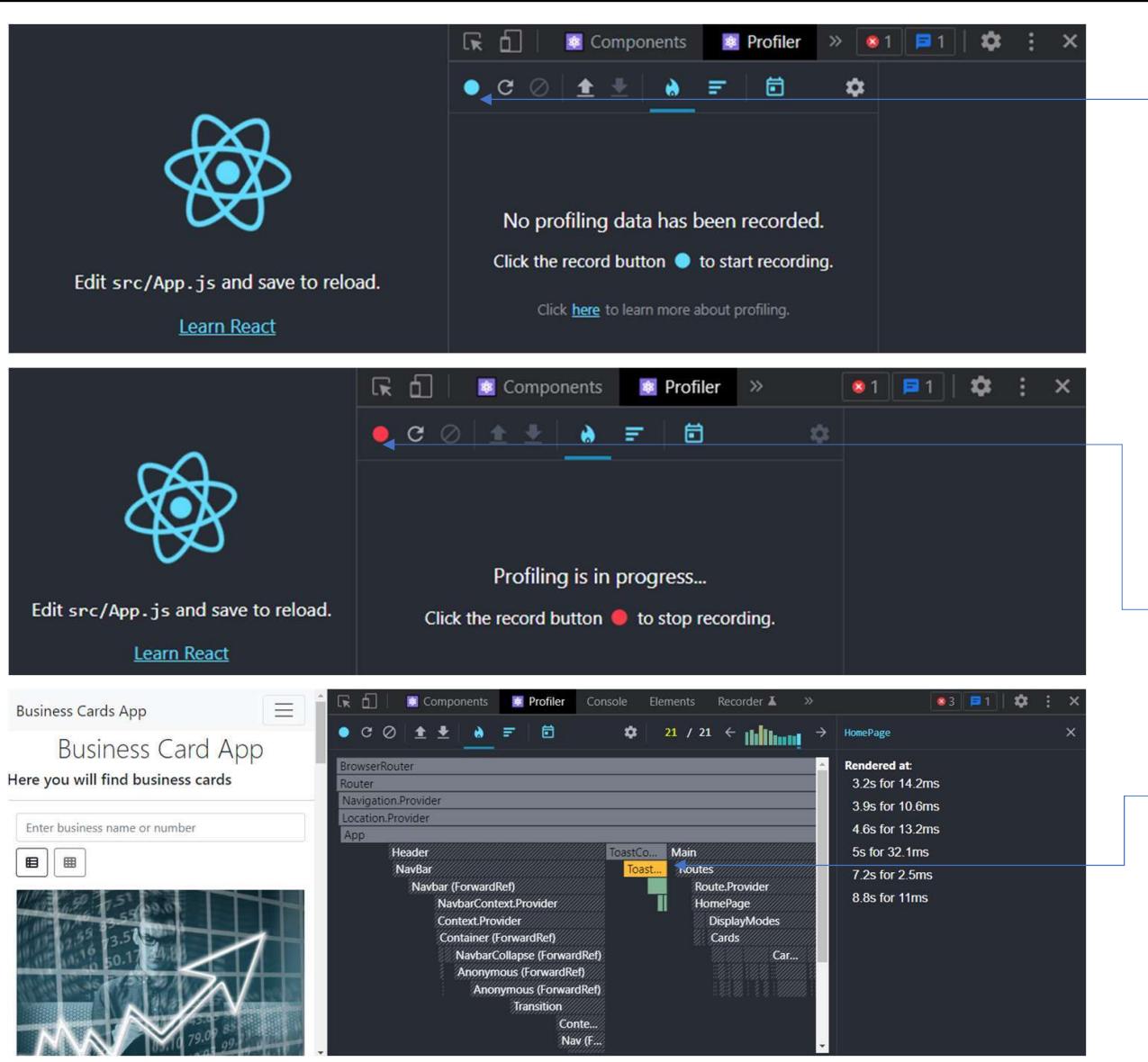
Profiler

בלשונית זאת נוכל לעשות בדיקות אופטימיזציה

- לחיצה על כפתור record תחילת להאזין לאירועים שוקרים ב – DOM

- לחיצה נוספת תעצור את ההקלטה ובמידה ויהיו נתונים להציג (כמו משך הזמן שלקח לכל אירוע לפחות) החלקים הללו יוצגו לנו.

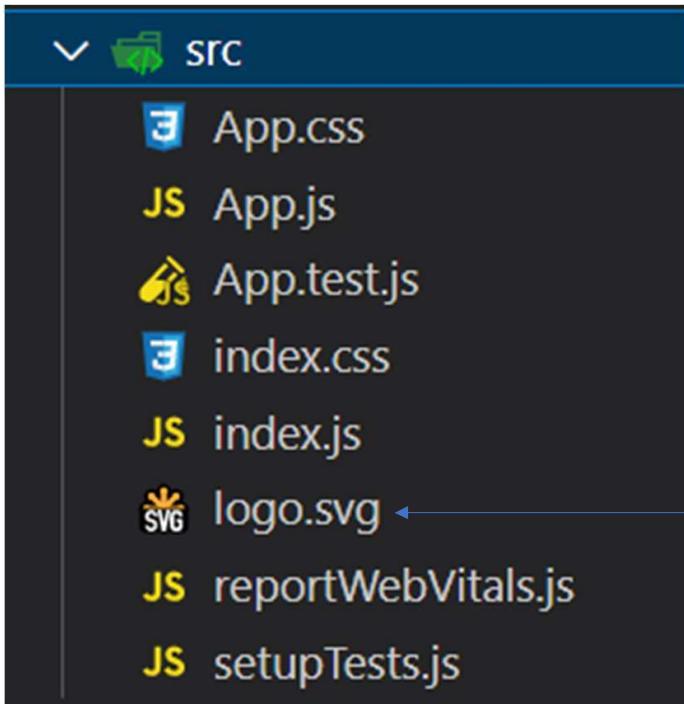
- לחיצה על אחד מהאירועים תיתן לנו פרטים עליון



תחילה עבודה

尼克י ראשוני של האפליקציה מתונות וערכים דיפולטיביים של
create-react-app





Src

- מחיקת קובץ הלוגו של ריאקט

App.js

- מחדיקת יבואקובץ הלוגו של ריאקט
- מחדיקת 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

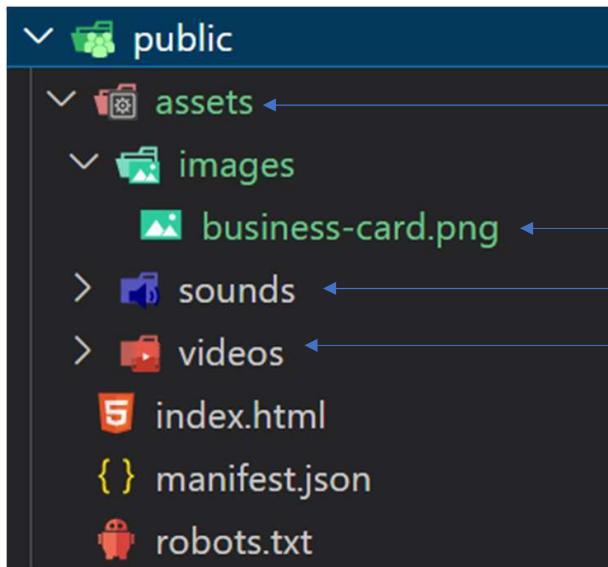
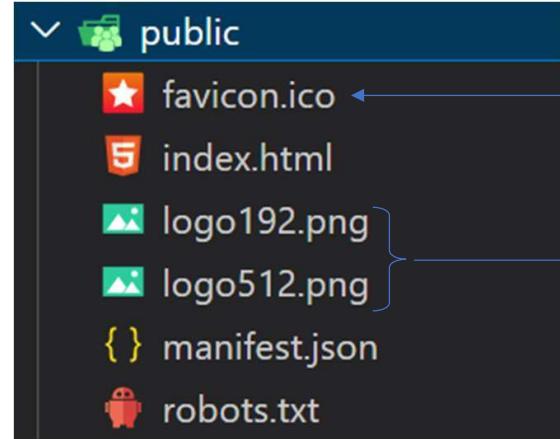
- מחיקת תוכן הקובץ
- יצירת של מחלקות עיצוב משלנו
- מחיקת התוכן של הקובץ של App.css

```
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 }
```

App.css M X

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



public

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

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      />
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>
```

index.html

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

```
{ } 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 מקום 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

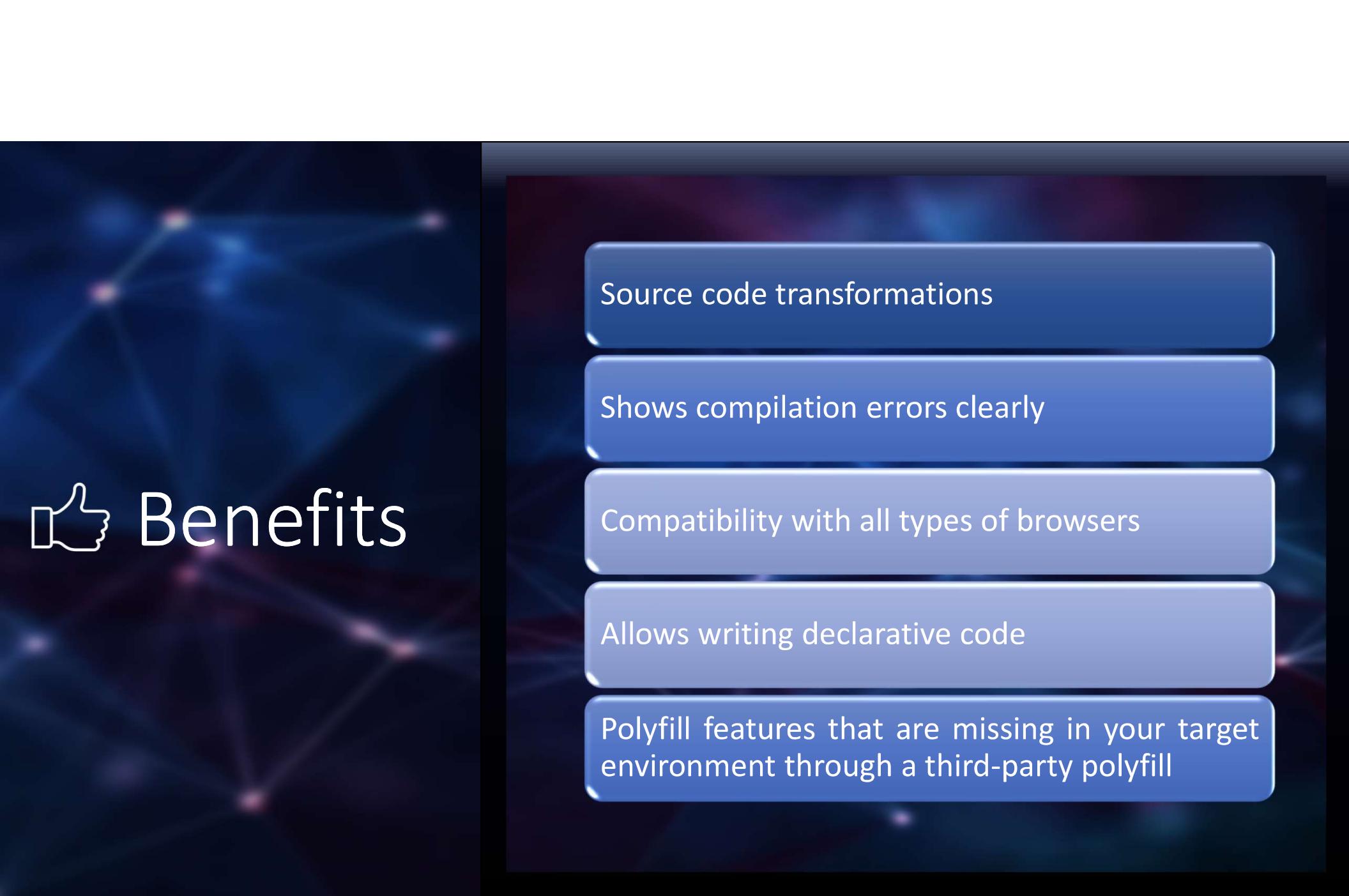
<https://babeljs.io>





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 <https://babeljs.io/> תחת הלשונית **Try it out** של

- ארגז המשחקים זהה בנו שלושה חלקים:

- מסך ימני – מציג את הקוד לאחר היליך הקומpileציה
- מסך אמצעי – משמש לכתיבה קוד javascript דקלרטיבי ועכני
- מסך צידי – ובו אפשרות שנותן 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 following code:1 "use strict";
2 hallo
3 </>Arrows from the text 'TARGETS' and 'TARGETS' in the sidebar point to the bottom of the left code pane. The right pane contains this code:1 "use strict";
2
3 /*#__PURE__*/React.createElement(React.Fragment,
 null, "hallo");Arrows from the text 'TARGETS' and 'TARGETS' in the sidebar point to the bottom of the right code pane.

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

! **לינק להרצאה המסביר איך js babel.js עובדת מאחוריו 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

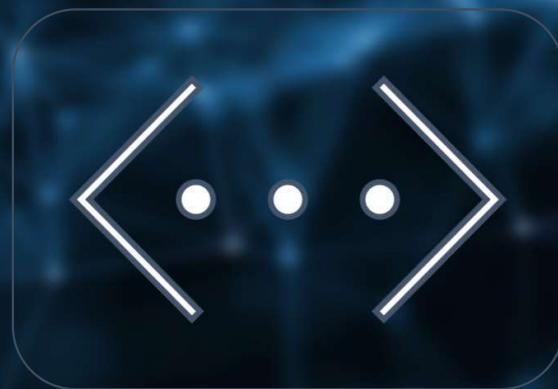


LOGIC
JAVASCRIPT



STYLES
CSS

Components Types



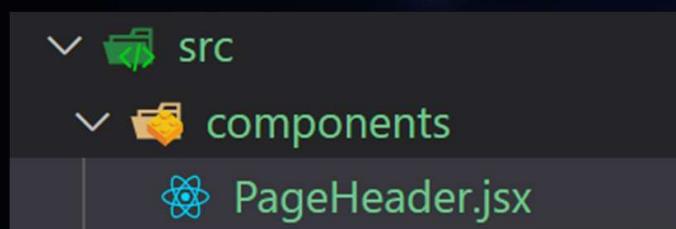
Function



Class

הכנות לשתייה

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



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



Template

יצור קומפוננט מסווג פונקציה שתחזיר
אלמנט של HTML אותו נציג לגולש



PageHeader

```
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;
```

- PageHeader.jsx •
 - נוצר קבוע בשם **PageHeader** שערך יהיה שווה לפונקציה אונימית
 - הפונקציה תחזיר אלמנט של **HTML** מסוג **H2** עם הכיתוב בתוכו
 - לבסוף ניצאת הפונקציה מהמודול באמצעות **export default**
- App.js •
 - ניבא את הקומפוננט שיצרנו
 - נציב אותה בתוך החלק של ה **HTML** אותה הפונקציה של הקומפוננט **App** מחרירה.
- ! במצגת זאת נתמקד בקומפוננטות מסוג **functional components** !
חויה לעתוף את כל האלמנטים שהפונקציה מחרירה באלמנט או של **React** או של **HTML**

התווצאה בדף

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

pageHeader works!



Compilation Error

במידה ותהי שגיאה בקוד Babel תתריע
לי על כר במספר מקומות



איתור שגיאות

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

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

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

On the right, the code editor displays the content of `PageHeader.jsx`:

```
PageHeader.jsx 2, U X
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 list items on the right to specific parts of the code editor interface: one arrow points to the file tree, another to the code editor title bar, and two arrows point to the code itself, highlighting the first two lines of the JSX component.

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 two main sections highlighted by red arrows:

- TERMINAL:** Shows the command-line output. It includes:
 - A local host message: "Local: http://localhost:3000".
 - An error message: "Failed to compile.".
 - A detailed syntax error from a file named "PageHeader.jsx":

```
SyntaxError: C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT
cent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fr
2 |   return (
3 |     <h2>pageHeader works!</h2>
4 |     <p>hallo world</p> ←
5 |     ^
6 |   );
7 | }
```
 - An "ERROR" message: "ERROR in ./src/components/PageHeader.jsx".
 - Module build failed (from ./node_modules/babel-loader/lib/index.js):
 - Another syntax error from the same file: "SyntaxError: C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT
cent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fr".
- PROBLEMS:** Shows a list of errors for a file named "PageHeader.jsx" located at "src\components".
 - A collapsed section header for "PageHeader.jsx" with "src\components".
 - Two expanded error items:
 - "JSX expressions must have one parent element. ts(2657) [Ln 3, Col 3]".
 - "Parsing error: Adjacent JSX elements must be wrapped in an en... eslint [Ln 4, Col 2]".

בדפסן

- בקונסול תופיע השגיאה
- בمسך התצוגה הראשי יופיעו פרטי השגיאה

! בمسך התצוגה הראשי ניתן ללחוץ על הסימן X ולחזור לתצוגת האפליקציה אך מומלץ לתקן את השגיאה בקוד במקום

Compiled with problems:

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.jsxParseElement (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)

Custom levels 2 hidden

2 Issues: 2

✖ Uncaught 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:12:1)

App.js:12

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.jsxParseElement (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)

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;
```

The screenshot shows a browser developer tools window with the 'Elements' tab selected. The page content is displayed above the tree view:

pageHeader works!
hallo world

The DOM tree below shows:

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body cz-shortcut-listen="true">
    <div id="root">
      <div class="App"> == $0
        <h2>pageHeader works!</h2>
        <p>hallo world</p>
      </div>
    </div>
  </body>
</html>
```

A green bracket on the right side of the slide points to the curly braces around the `div class="App"` element in the DOM tree.

במקרה זהה מקור השגיאה היה
שניסיתי להחזיר למעלה אלמנט
HTML אחד מהקומponent

- אם אני לא מונין לעתוף את שני
האלמנטים באלמנט עיצובי של
HTML כמו `div` ריאקט `pseudo element` משלה שנקרא `React.Fragment`

שבעזרתו אוכל
לעתוף 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

הדרך המוקוצרת של כתיבת האלמנט
`<></>` היא `React.Fragment`



Logic

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

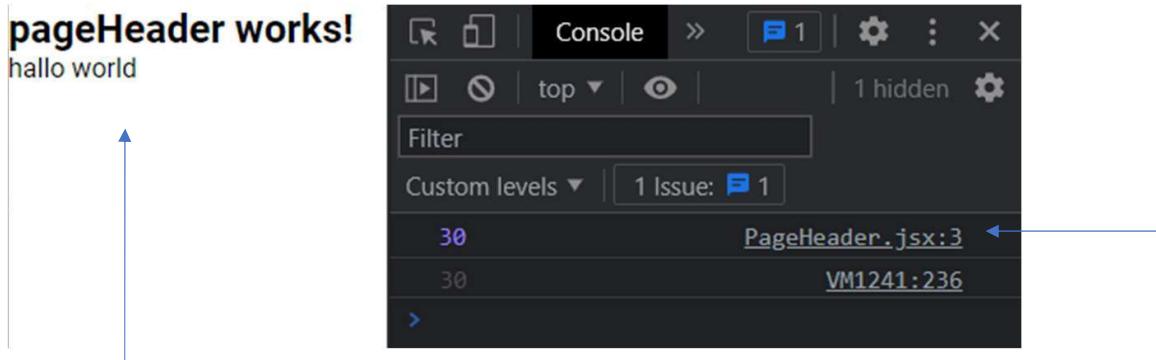


הוספה לוגיקה

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

- 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 אזור JAVASCRIPT באזורי המוגדר
HTML בקומפוננט



String interpolation example

פתחת אזור JAVASCRIPT באזורי המועד
ל – HTML מתבצעת על ידי פתיחה
ושגירה של סוגרים מסולסלים
בדוגמה של להלן:

```
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

- בתוך ה – scope של הקומפוננט יוצרתי קבוע בשם text והשוויתי את הערך שלו למחזרות תווים
- באזורי המועד לשפת HTML פתוחתי אזור של JAVASCRIPT בעזרת פתיחה סוגרים מסולסלים ובתוכם הצבתי את שם הקבוע שיצרתי
- בעזרת string interpolation נוסיף פתיחה אזור JAVASCRIPT גם בתוך אלמנט נוסף של HTML והפעם ביצעתו חישוב כפי ששפת JAVASCRIPT יודעת לעשות

הטוצאה בדפסן:

- כפי שניתן לראות הטקסט הוצב במקום שהגדרתי לו
- החישוב בוצע במקום שהגדרתי לו



Styles

הוסף עיצוב לkomponenT



Styles Types



INLINE



IMPORT STYLES FROM
MODULE



EXTERNAL LIBRARIES



Inline style

הדריך להזrik inline style לאלמנט HTML בקומפוננט של ריאקט היא על ידי הוספת המאפיין style ולהשוו את הערך שלו לאזרע javascript שלו 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 שלו הם המאפיינים העיצובים שאמו מעוניינים לשנות וערכיהם כמו בכל אובייקט JAVASCRIPT יוופיעו לאחר הנקודותם

- ניצור בתגית HTML הפותחת מאפיין `style` ונשווה את הערך שלו לאזרור JAVASCRIPT אליו נעביר את שם הקבוע שיצרנו

- בדוגמה השנייה נעביר ישירות אובייקט קונפיגורציות לתוך המאפיין `style` בתגית הפותחת של האלמנט פסקה של HTML

- התוצאה בדפסן

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



Styles from module

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

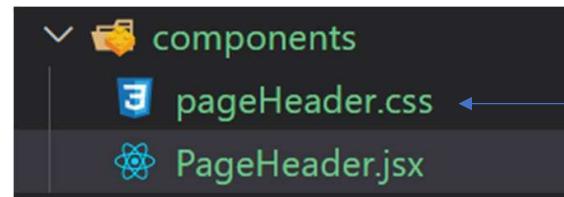


Styles from module

בדוגמה של להלן:

- ניצור קובץ חדש בשם pageHeader.css
- ניצור מחלקת עיצובה של css בקובץ שיצרנו
- ניבא את המודול לתוך הקובץ של הקומponent
- נשתמש במחלקת העיצובה שיצרנו

יש לשים לב ל syntax של המאפיין class בריקט שהוחלף ל - className



pageHeader.css

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

PageHeader.jsx

```
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 השלישי לעיצוב אלמנטים של HTML
בקומפוננטות של ריאקט היא באמצעות היבאת
ספריות עיצוב ושימוש במחלקות
העיצוב שלן



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 החלק המועד ל – HTML בקומפוננט ונציב בתוכו את הערך של המפתח שהילצנו מתוך אובייקט הprops

ChildComp.jsx

```
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;
```

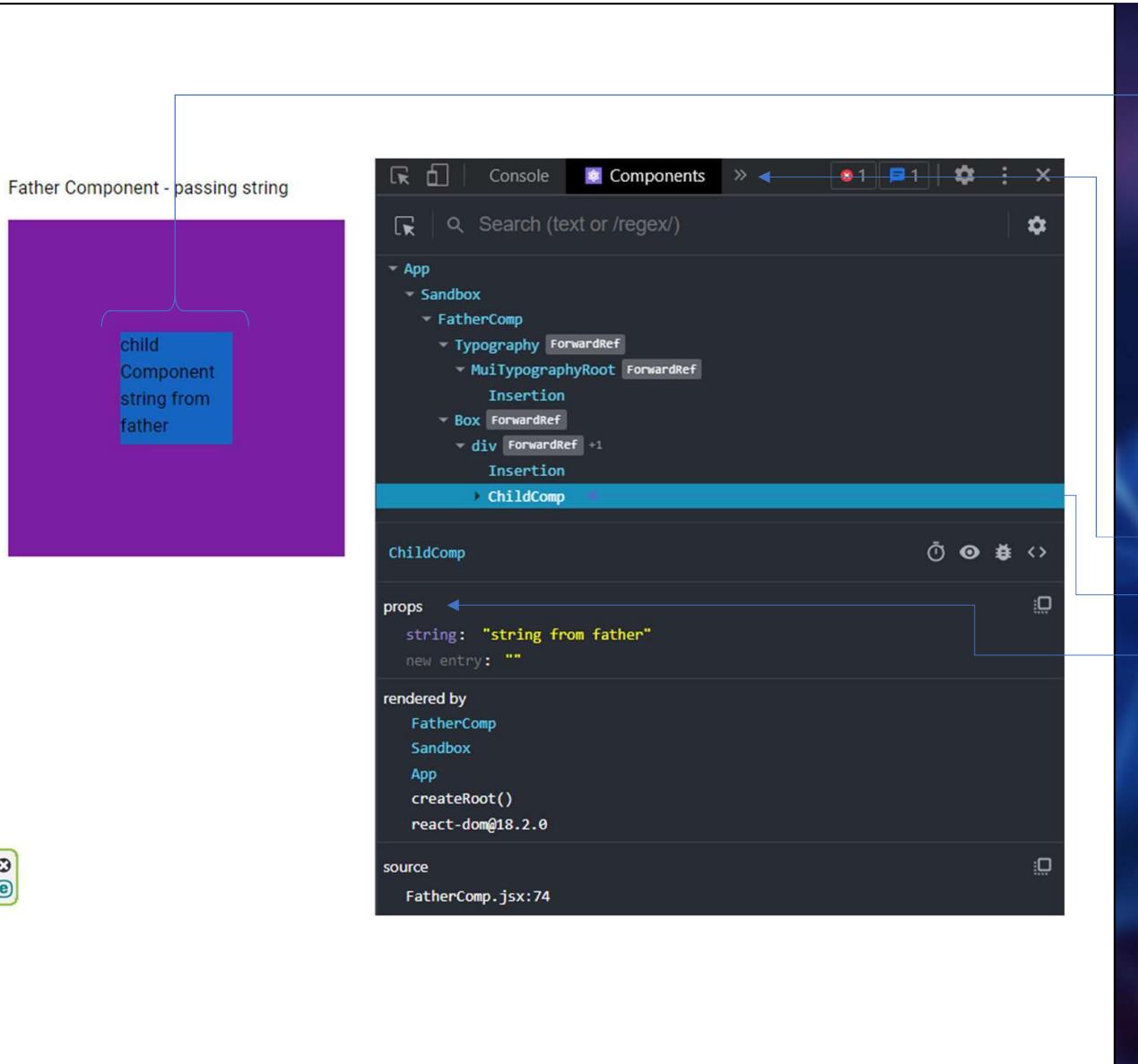
```
client > src > sandbox > props > FatherComp.jsx > ...
1 import { Box, Typography } from "@mui/material";
2 import React from "react";
3 import ChildComp from "./ChildComp";
4
5 const FatherComp = () => {
6   const string = "string from father";
7
8   return (
9     <>
10       <Typography variant="body1" 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 מחרוזתווים
- 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.

התוצאה בדף

- ניתן את הטקסט שהעבכנו מkomponent האב מוצג בתוך komponent הבן
- בנוסף בגלל שהורדנו את התוסף react tools dev kunnen אנו יכולם לgesht לשונית components
- לחוץ על komponent שמשמעותו אותו
- ולקבל בין היתר את המפתחות והערכיהם שמעברים 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 בתוך החלק המיועד ל – HTML בkomponent ונציב בתוכו את הערכים של המפתחות שהילצנו 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 };
```

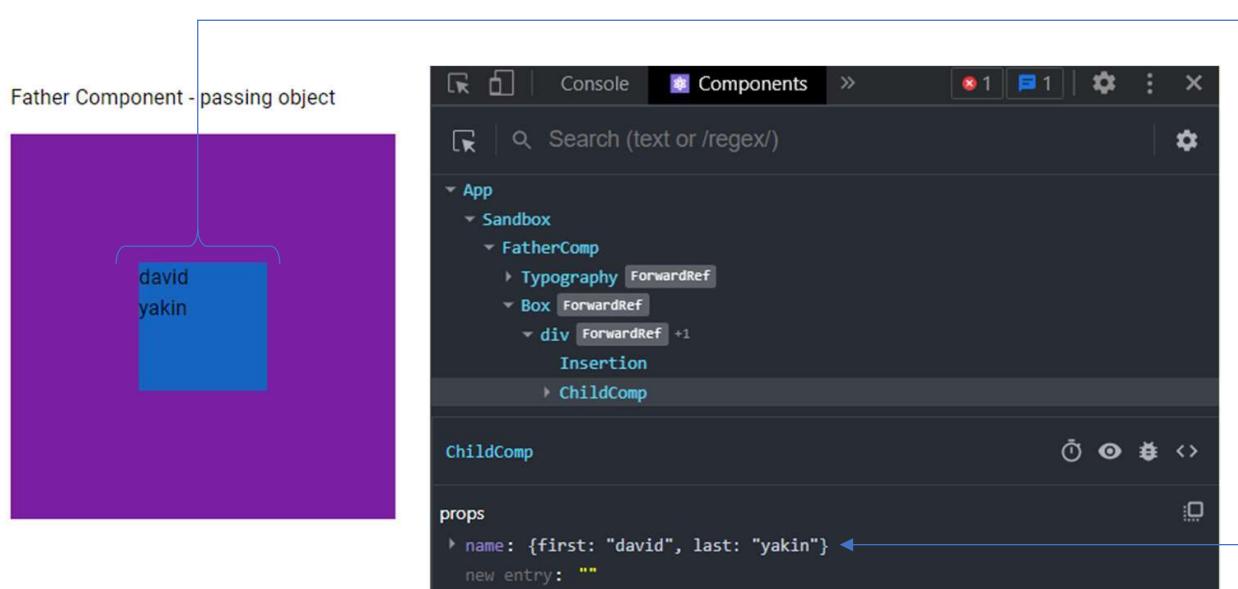
Father Component

- ניצור קבוע בשם name שערך יהיה אובייקט עם מפתחות וערכים
- נעשה השמה למפתח name בתוך 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 שהעבכנו מקומפוננט האב לקומפוננט הבן בתוך אובייקט מוצג בקומפוננט הבן
- וכי אובייקט ה – `props` מכיל עצ מפתח בשם `name` שהערך שלו זה האובייקט שיצרנו





Sending two keys

הדרך להעביר יותר מפתחות אחד לאובייקט
הפרופו



Child Component

- ב글 שkomponenT מסוג פונקציה מתנהגת כמו כל פונקציה ב – JAVASCRIPT נוכל לחלץ מאובייקט props מספר מפתחות first, last

- יפתח איזור של JAVASCRIPT בתוך החלק המועד 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
```

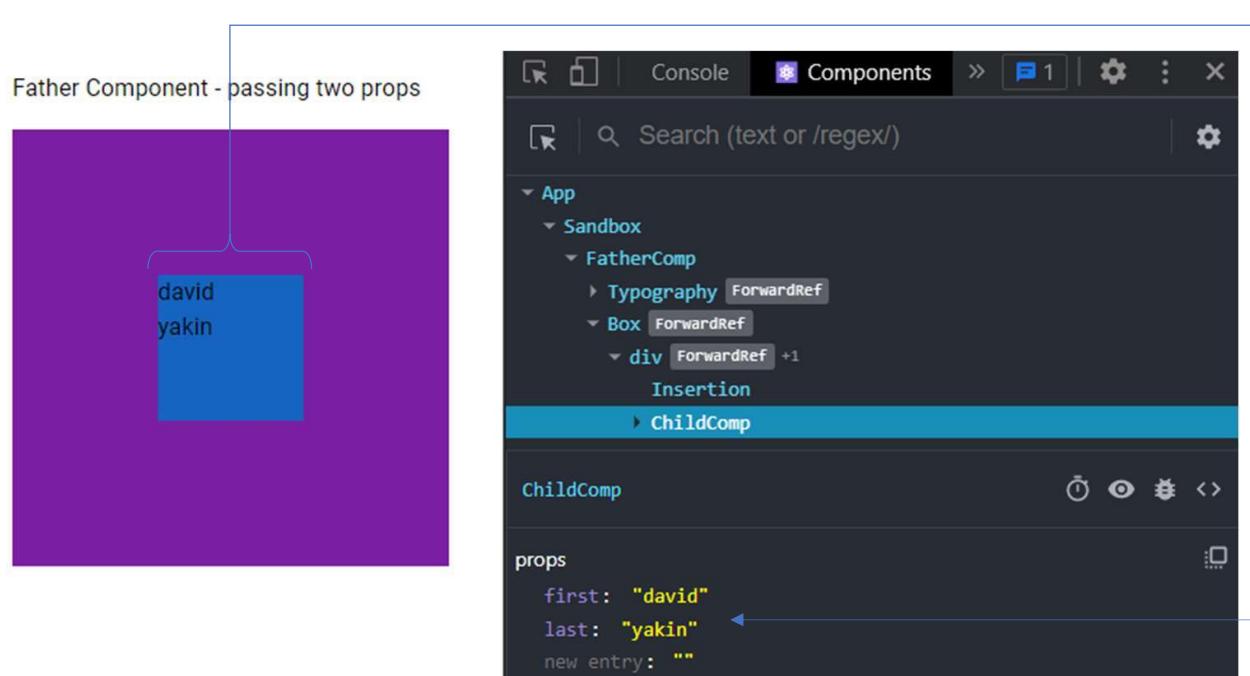
```
57 const FatherComp = () => {
58   const name = { first: "david", last: "yakin" };
59
60   return (
61     <>
62       <Typography variant="body1" m={2}>
63         {" "}
64         Father Component - passing two props
65       </Typography>
66       <Box
67         sx={{
68           m: 2,
69           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 שיצרנו לתוך מפתח משלה באובייקט הפַּרְוֹפּ

התוצאה בדף

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



Props מיפוי

```
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

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

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` בשבייל לביצ' לולאות על מרכיבים באזור המיועד ר-HTML

בדוגמה שלהן:

- יצרנו 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 נערךת האיטרציה
- בתוך הלולאה אני מדפיס את המערך ומציג לגולש כתוב שמקורו במערך
- כל אלמנט שאנו מכפילים באיטרציה צריך לקבל את המאפיין `key` שצריך להיות ייחודי.

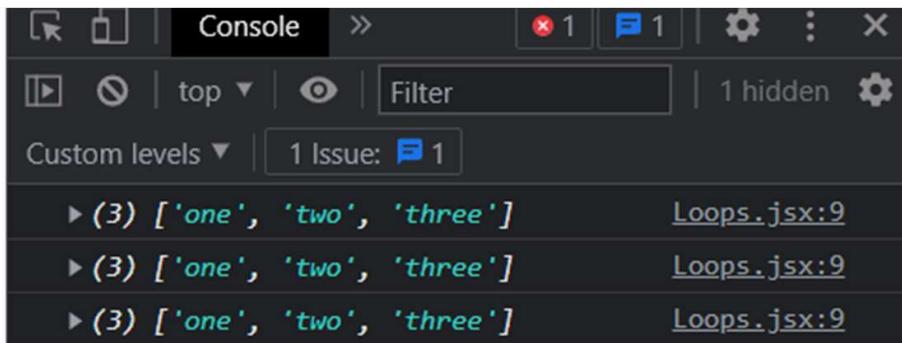
```
Loops.jsx X
client > src > sandbox > Loops.jsx > ...
1 import React from "react";
2 import { Box } from "@mui/material";
3
4 const Loops = () => {
5   const arrayOfString = ["one", "two", "three"];
6   return (
7     <Box m={2}>
8       {arrayOfString.map((item, index, array) => {
9         console.log(array);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

התוצאות בדף

React בחרה להשתמש בMETHOD 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



משימת Map



Business-cards-app

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

Conditional Rendering

תצוגות מיוחדות שונות כאשר יש מידע להציגה וכאשר אין

! יש לעבור על החלק של Layout במצגת של UIW בטרם ממשיכים במצגת הציג



Cards.jsx M X

```
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 ונרצת לעדכן בכר את הגולש.

- נשים לרגע את הקבוע cards שיצרנו בתוך הערה
- ניצור 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 למערך cards הקומponent יעצור ויחזיר תצוגה לגולש שתכלולאזור טקסט עם מחוזת תווים

Events

הדרך להאזין לאירועים ולהפעיל מטודות בעקבותיהם





JAVASCRIPT EVENTS

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

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



onClick event

דוגמה שללן:

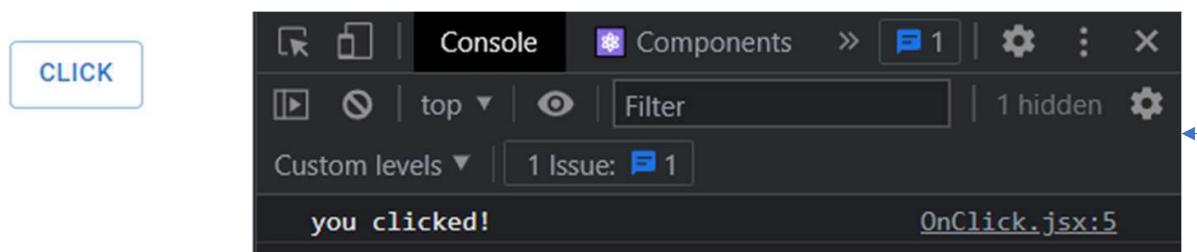
- יצרנו קומפונט בשם **OnClick**
- יצרנו קבוע בשם **handleClick** שמשווה ערך לפונקציה אוניברסלית כשתופעל תדף בקונסול מחרוזת תווים
- יצרנו כפטור שיאזין לאיירע **onClick** ויפעל את הפונקציה **handleClick** ברגע שהאיירע יקרה.

הטזאה בדף:

- כnellחץ על הכפטור תודפס בקונסול מחרוזת התווים מתוך 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 = () => 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;
```





Function invocation with parameters

הפעלת פונקציה עם פרמטרים



Function with parameters

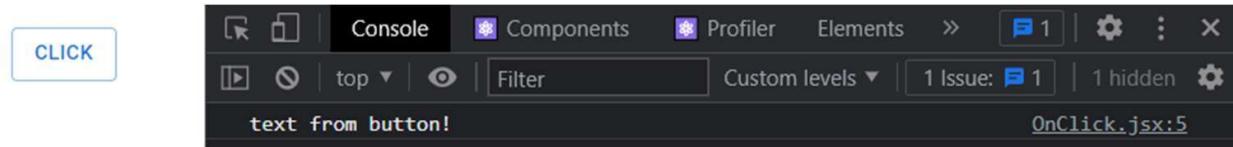
בדוגמה שללן:

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

כשנלחץ על הכפתור תודפס בקונסול מחרוזת התווים מתוך הfonkציה handleClick

אם עבור אירוע ישירות את הפעלת המטודה handleClick עם הפרמטר ללא הfonkציה האנוימית היא תופעל עם יצירת הקומפוננט ולא מתי 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;
```





**Catching the
event and passing
it to the function**

תפיסה האירוח והעברתו לפונקציית ה – call
back



Catching event

בדוגמה שלהן:

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

- אני מעביר בערך של האירוע onClick פונקציה call back אנונימית שמקבלת את האירוע ופעילה את הפונקציה handleClick עימיו

התוצאה בדף:

- כשנלחץ על הכפתור יודפס בקונסול האובייקט שהאזין לאירוע והפעיל את handleClick

The screenshot shows a code editor window for a file named `onClick.js`. 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 of medium. A `CLICK` button is visible in the browser's developer tools' Elements tab, which is currently active. The browser's developer tools also show the rendered HTML for the button, including its class names and attributes.

```
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={e => handleClick(e)} variant="outlined" sx={{ m: 2 }}>
9       Click
10    </Button>
11  );
12}
13
14 export default OnClick;
```

CLICK

OnClick.js:5

```
> <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-19skcmv-MuiButtonBase-root-MuiButton-root" tabindex="0" type="button">...</button> flex
```

הדרך השניה להעביר

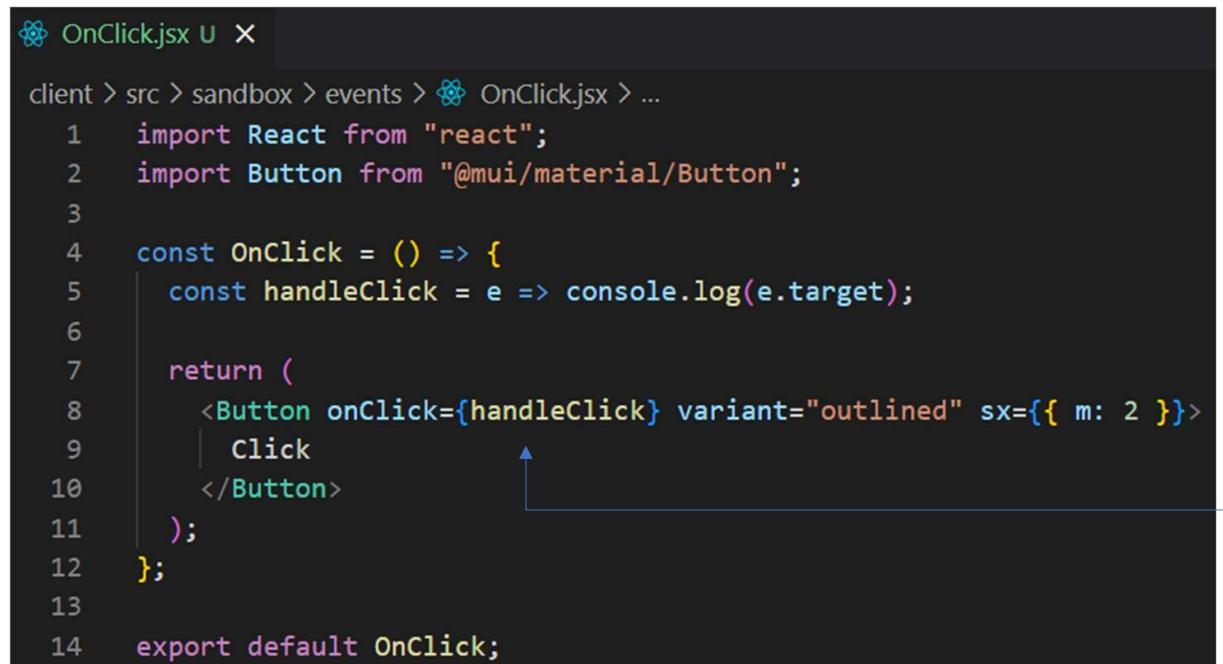
איירונן לפונקציה

בדוגמה שללן:

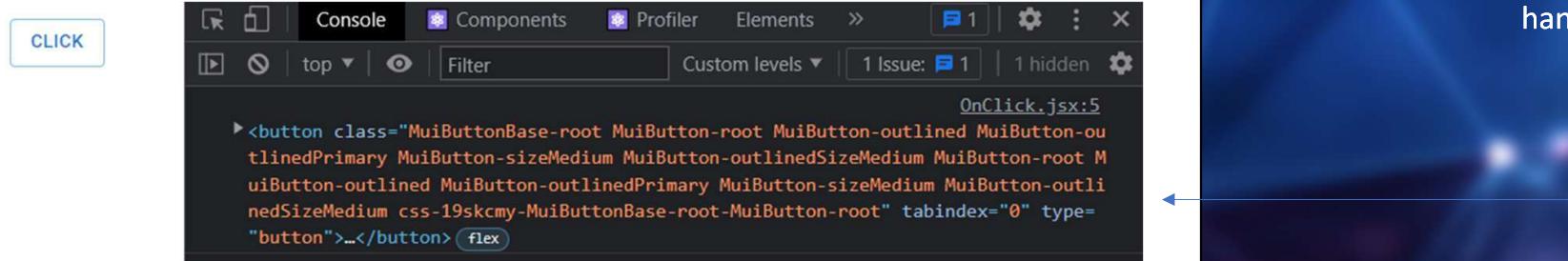
- הפעם אני כבירך לא מעביר לפונקציה handleClick פרמטר אלא מעביר אותה כפונקציית javascript call back. אולם javascript בכל זאת תעביר לפונקציה את האירוע ואוכל להדפיסו בקונסול

התוצאה בדף:

- כשנלחץ על הכפתור יודפס בקונסול האובייקט שהאזין לירע 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 ששווה ערך לפונקציה אונימית שמדפיסה בקונסול מחרוזת תווים
- אני עושה השמה למפתח בשם handleClick באובייקט הפרופס ומשווה את הערך שלו למטרודת handleClick שיצרתי לעיל.

בקומפוננט הבן

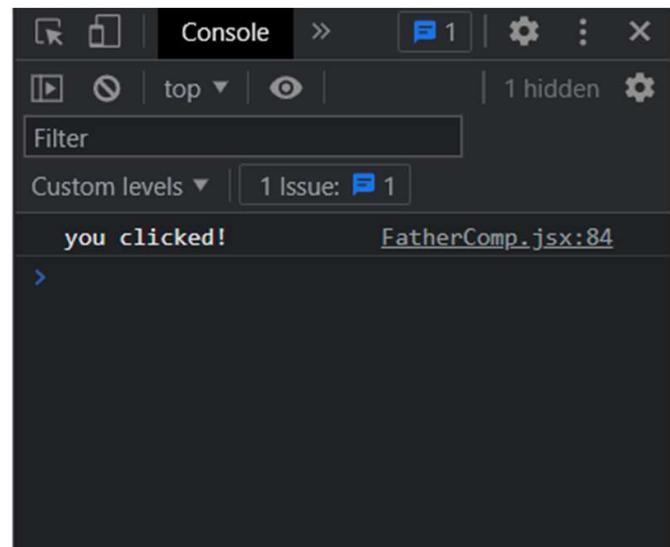
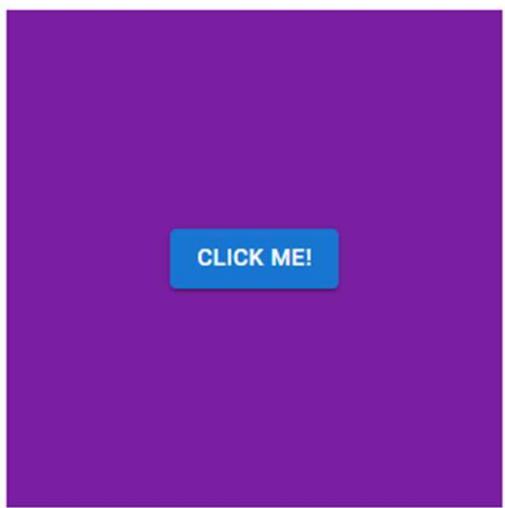
- אני מחלץ את המפתח מהATTRIBUTES מאובייקט הפרופס
- אני מאמין לאירוע onClick שtáfuil את handleClick מטרודת

```
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 הכפתור בקומponentה הבן
מופעלת המטודה handleClick שמדפיסה
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` מספר ותדפיס בקונסול את מחרוזת התווים ”no” והוסף את המספר שמופיע ב מפתח `_id`
- העבר באובייקט הפרויקט את מטלות `onDelete` ו-
- מוקומונט האב `xsj.Cards` דרך קומפוננט הבן `xsj.Card` ועד לקומפוננט שמתעסקת עם `Action Buttons`
- בלחיצה על איקון הלב הדפס את מחרוזת התווים ”no: liked card no:” והוסף את הערך שמופיע ב מפתח `_id`
- בלחיצה על איקון פח האשפה הדפס את מחרוזת התווים ” liked card no: ” והוסף את הערך שמופיע ב מפתח `_id`
- בלחיצה על איקון עריכת הCARTEIS הדפס את מחרוזת התווים ” edit card no. ” והוסף 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	ס.ן
	.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 מהעריכים שמצועים במערך בלבד	.oneOf(['News', 'Photos'])		.13
instance of a class	.instanceOf(new Class)		
ערך דיפולטיבי	. defaultProp		
যוצר סימן חדש	.symbol	Symbol()	.7
	.bigint		.8
	.node		.9
	.element		.10
	.elementType		.11
	.shape({ })		.16
מודוא שלא הועברו מפתחות שלא נמצאות בסוג הפרופ	.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 שיבאנו "prop-types"

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

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

! אני יכול לעשות השמה למפתחות שלא!

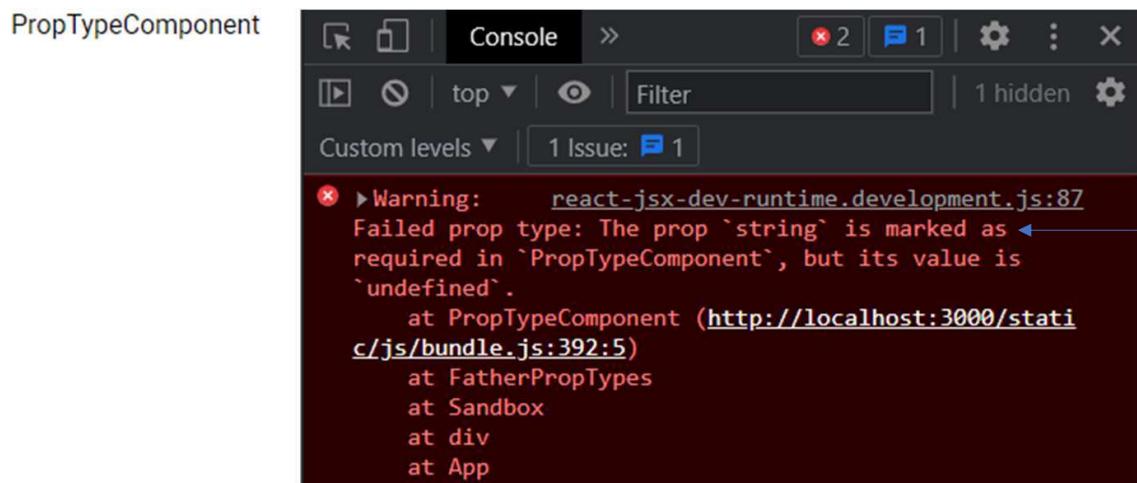
```
client > src > sandbox > propTypes > FatherPropTypes.jsx > FatherPropTypes.js
1 import React from "react";
2 import PropTypeComponent from "./PropTypeComponent";
3
4 const FatherPropTypes = () => {
5   return <PropTypeComponent />; ←
6 }
7
8 export default FatherPropTypes;
```

```
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 בדיקה באמצעות `propTypes` נזרקת לי על

כך שגיאה





Main Types

סוגי הערכים המרכזיים



Main Types

בקומפוננט האב

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

בקומפוננט הבן

- הקומפוננט PropTypeComponent מקבל בפרמטר את האובייקט props
- נחלץ את הערכים מתוך האובייקט props
- עשה השמה למפתח propTypes בתוך PropTypeComponent שיצרנו ומשווה את הערך שלו לאובייקט שיבדק את המפתחות שאינו מעוניין שיופיע 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 שהיא זקוקה להם באובייקט הprops

בקומפוננט הבן

- הקומפוננט מקבלת בפרמטר את האובייקט props
- נחלץ את הערכים מຕוך האובייקט props
- נעשה השמה למפתח propTypes בתוך קומפוננט PropTypeComponent לkiemponent שיצרנו ונשווה את הערך שלו לאובייקט שיבדק את המפתחות שאינו מעוניין שייהו באובייקט props ואת הערכים 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 ערכאים או לקבוע את הערךים באופן ליטרלי



oneOfType & oneOf

בקומפוננט האב

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

בקומפוננט הבן

- הקומפוננט מקבלת בפרמטר את האובייקט props
- נחלץ את הערכים מຕוך האובייקט props
- נעשה השמה למפתח propTypes בתוך PropTypeComponent לקומפוננט שיצרנו ומשווה את הערך שלו לאובייקט שיבדק את המפתחות שאנו מעוניין שייהיו באובייקט props ואת הערכים 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

אנו מקבלים שגיאה בגל שניסינו להעביר
במפתח סוֹך באובייקט הפרויקט מחרוזת
תווים שאינה אחת ממחוזות התווים
שהגדכנו

PropTypesComponent
PropTypesComponent
PropTypesComponent
PropTypesComponent

The screenshot shows the React DevTools Components tab. At the top, there are tabs for Console, Components, and a search/filter bar. Below the search bar, it says "Custom levels" and "1 Issue". A red box highlights the first issue: "Warning: Failed react-jsx-dev-runtime.development.js:87 prop type: Invalid prop `two` of value `c` supplied to `PropTypesComponent`, expected one of ["a", "b"].". The stack trace shows the component hierarchy: PropTypesComponent, FatherPropTypes, Sandbox, div, App.

```
undefined          PropTypesComponent.jsx:42
✖ Warning: Failed react-jsx-dev-runtime.development.js:87
prop type: Invalid prop `two` of value `c` supplied to
`PropTypesComponent`, expected one of ["a", "b"].
    at PropTypesComponent
    at FatherPropTypes
    at Sandbox
    at div
    at App
```



Exact & isRequired

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



Exact

בקומפוננט האב

- נעביר לה את המפתחות שהיא צריכה להם באובייקט הפרופס אולם נשים מפתח אחד יותר מיד' בתוך האובייקט שהוא מעבירים

בקומפוננט הבן

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

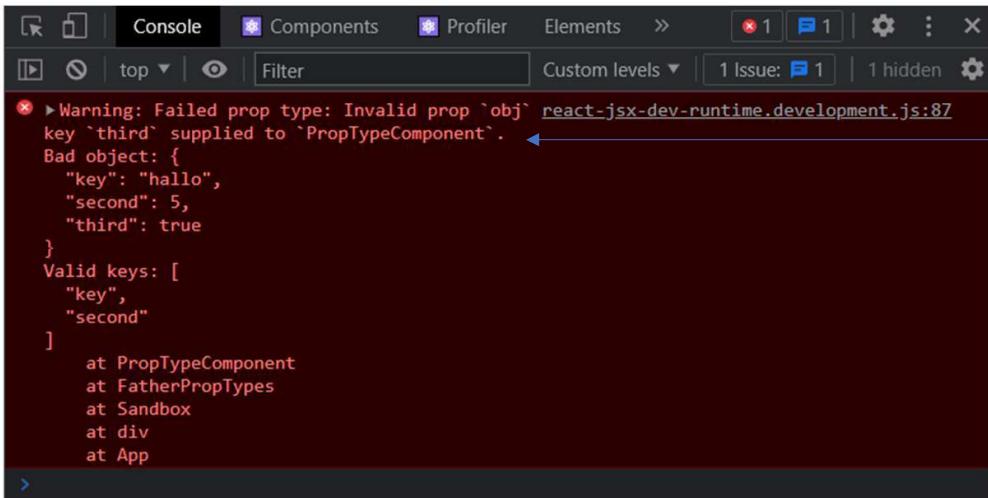
```
52 const FatherPropTypes = () => {
53   const obj = { key: "hallo", second: 5, third: true };
54   return <PropTypeComponent obj={obj} />;
55 };

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 שהערכנו מפתח שלישי של PropTypes
זורקתו לנו הודעת שגיאה על קר



The screenshot shows the Chrome DevTools console with the title "PropTypeComponent". It displays a warning message:

```
Warning: Failed prop type: Invalid prop `obj` supplied to `PropTypeComponent`.  
key `third` supplied to `PropTypeComponent`.  
Bad object:  
{  
  "key": "hallo",  
  "second": 5,  
  "third": true  
}  
Valid keys:  
[  
  "key",  
  "second"  
]  
at PropTypeComponent  
at FatherPropTypes  
at Sandbox  
at div  
at App
```

isRequired

בקומפוננט האב

- נציג את קומפוננט הבן בתוך קומפוננט האב אך לא נעביר לה את המפתח שהוא זמין לו באובייקט הprops

בקומפוננט הבן

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

התוצאה בדפדפן

- הודעת השגיאה של PropTypes בגלל שלא העברנו את המפתח שהגדכנו 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

בקומפוננטת האב

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

בקומפוננטת הבן

- נחלץ את המפתחות שאנו זקוק להם מאובייקט `PropTypes`
- ניצור קבוע בשם `imageType` שייהי שווה ערך למפתח `shape` מתוך אובייקט `PropTypes` שיקבל אובייקט קונFIGורציות עם המפתחות והערכים שאנו מעוניינים שיהיו ב `interface` של התמונה
- נגדיר 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 בניים בתוך קומפוננט האב
 - לראשונה נעביר מפתח בשם name עם ערך לאובייקט הפרויקט
 - בשנייה רק נציג את הקומפוננט מבלי להעביר לה מפתחות לאובייקט הפרויקט

בקומפוננט הבן

- הקומפוננט הפרויקט צריך לקבל את המפתח name באובייקט הפרויקט ולהחזיר אותו

- גודיר שערך המפתח name יכול להיות כל סוג של ערך בעקבות any אבל שייה ובה להעביר בו ערך כלשהו

- השתמש בdefaultProps כדי לקבועערכים דיפולטיבים למפתחות באובייקט הפרויקט כך שאם לא יעבירו לנו ערך במפתח name נקבע שהערך הדיפולטיבי שלו יהיה "david"

התוצאה בדף

- בגל שהעבכנו את המפתח name לקומפוננט בן הראשון אנו מקבלים את הערך שהעבכנו

- בgal שלא העבכנו את מפתח name בקומפוננט בן השני מוצג הערך הדיפולטיבי שקבענו

! לא מומלץ להשתמש ב – any אלא להגיד 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

בדיקות 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 להעביר מחרוזת תווים או אפילו קומפוננטות שלמות

קומפוננט הבן

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

- אני מחזיר מהקומפוננט את שני המפתחות שהילצתי כך שיוצגו לגולש

בעזרת PropTypes נודא ש:

- בפתח node הועבר לנו ערך שניית 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

קומפוננט האב

- בין התגיות הפתוחת 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

קומפוננט הבן

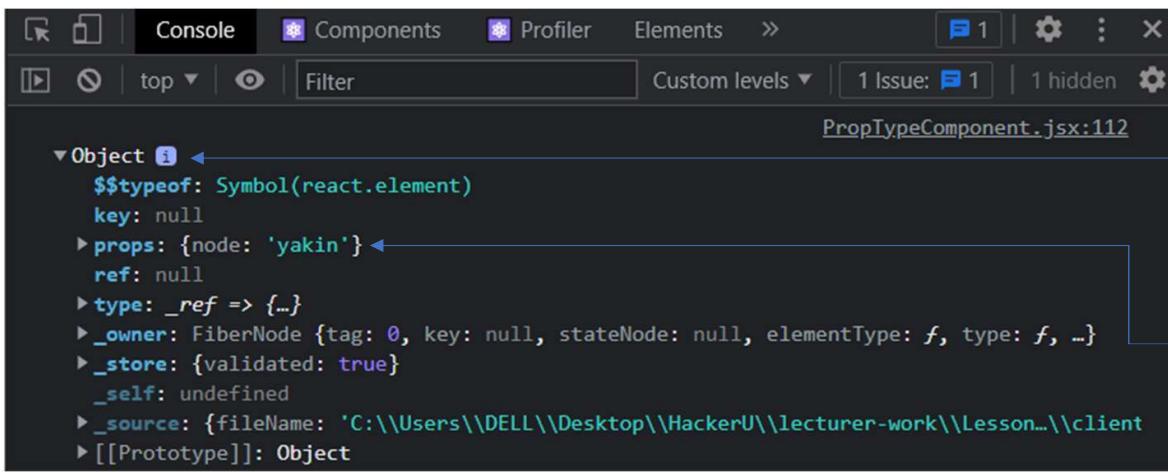
- אני מחלץ את המפתחות node children
- מדפיס בקונסול את children
- מחזיר מהקומפוננט לגולש children כי הערך של המפתח react על ידי צרי להיות אלמנט של element שימוש במפתח element של PropTypes

```
89 const FatherPropTypes = () => {
90   return (
91     <PropTypeComponent node="David">
92       <PropTypeComponent node="yakin" />
93     </PropTypeComponent>
94   );
95 };
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111 const PropTypeComponent = ({ node, children }) => {
112   console.dir(children);
113   return (
114     <>
115       {node} {children}
116     </>
117   );
118 };
119
120 PropTypeComponent.propTypes = {
121   node: PropTypes.node.isRequired,
122   children: PropTypes.element,
123 };
```

החוצאה בדף

- ניתן לראות שהאלמנט שהערתתי לקומפוננט ב - children הוא מסוג אובייקט props שהערך שלו הוא אובייקט עם המפתח node והערך שהערתתי לקומפוננט הבן שבתור קומפוננט הבן

David yakin



The screenshot shows the Chrome DevTools Elements tab with the title "PropTypeComponent.jsx:112". The main pane displays the structure of a React component node:

```
Object {  
  $$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 נעביר לקומפוננט הבן ב מפתח ה - children מספר אלמנטים

בקומפוננט הבן

- נחלץ את מפתח children

נדפס אותו

- נזכיר אותו מהקומפוננט

באמצעות PropTypes נבדק שאכן הועברו לקומפוננט יותר מאלמנט React אחד באמצעות השילוב של הפונקציה PropTypes.element שתתקבל arrayOf

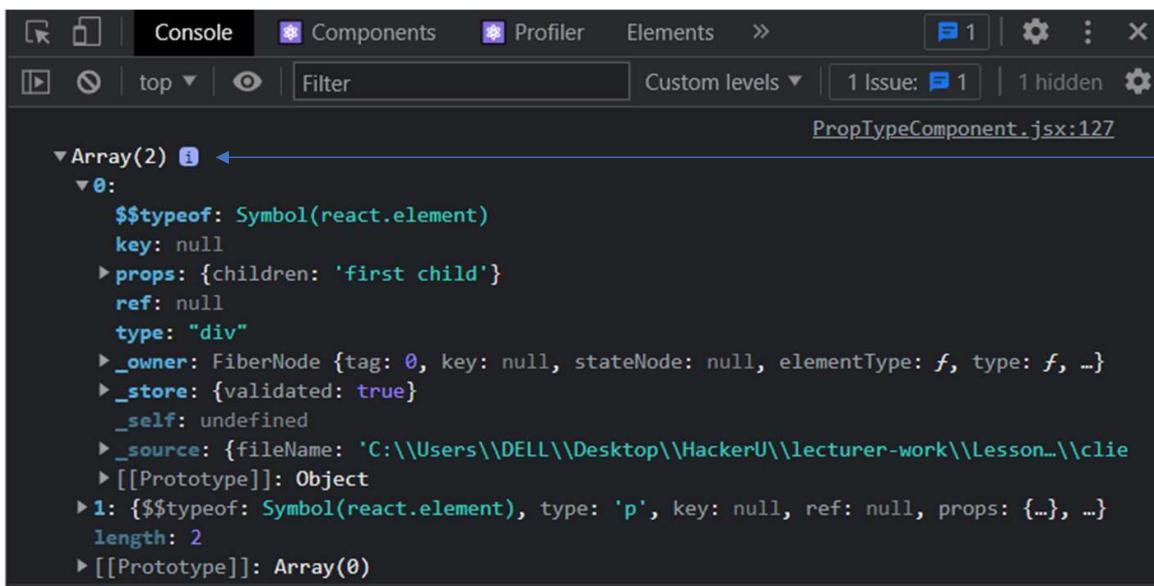
```
98  const FatherPropTypes = () => {
99    return (
100      <PropTypeComponent>
101        <div>first child</div>
102        <p>second child</p>
103      </PropTypeComponent>
104    );
105  };

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

התוצאה בדף

בגל שהעberman יותר מאלמנט אחד React הכניסה את האלמנטים לתוך מערך שכלי אלמנט הוא איבר במערך

first child
second child



```
Props: {children: "first child"}  
children: Array(2)  
 0: Object  
    $$typeof: Symbol(react.element)  
    key: null  
    props: {children: "first child"}  
    ref: null  
    type: "div"  
  1: Object  
    $$typeof: Symbol(react.element), type: 'p', key: null, ref: null, props: {...}, ...  
length: 2
```

משימת

חלק א' PropTypes



Business-cards-app

- צור את הנתיב `src/cards/models/types`
- צור שלושה קבצים בנתיב 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 addressType.js`

משימת

חלק ב' PropTypes



Business-cards-app

• cardType

- יבא את הקבועים addressType imageType שיצרת למודול

צור קבוע בשם `cardType` שיהיה שווה ערך לערך שיחזור ממוגנת
PropTypes.shape אליה תעביר את אובייקט הקונפיגורציות עם
הערכים הבאים:

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

- יצא את הקבוע שיצרת באמצעות `export default`

משימת חלק ג' PropTypes



Business-cards-app

- בקומפוננט card השתמש בPropTypes ובמודלים שיצרת בשקפים הקודמים על מנת לוודא כי כרטיס ביקור מסווג של cardType מועבר לקומפוננט Card
- וודא שככל הקומפוננטות שמרכיבות את קומפוננט 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 הנטיב :
src/component/PageHeader.jsx
- הקומponeנט קיבל את המפתחות title
subtitle באובייקט הפרויקט
- ניצור אלמנט h1 עם עיצוב של h2
- ניצור אלמנט h2 עם עיצוב של h5
- ניצור אלמנט hr בעזרת הקומponeנט Divider
- נוודא שהקומponeנט מקבלת את המפתחות
הדרושים לה באובייקט הפרויקט באמצעות
propTypes

CardsPage.jsx

CardsPage.jsx M X

```
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;
```

ניצור את הנתיב:

src/cards/pages/CardsPage.jsx

- נעביר את הכרטיסים לקומפוננט ה зат
- נעטוף את תוכן הדף במייל של MUI
- נציב את הקומפוננט שיצרנו PageHeader בתוך דף המועד לתצוגת הכרטיסים עם הכותרות המתאימות
- נציב את הקומפוננט Cards וنعביר אליו את המשתנה cards כמפתח באובייקט הפרופס

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 לטקו המסביר על האפליקציה
- ורואים את התמונה של הクリטיס

Static Folder

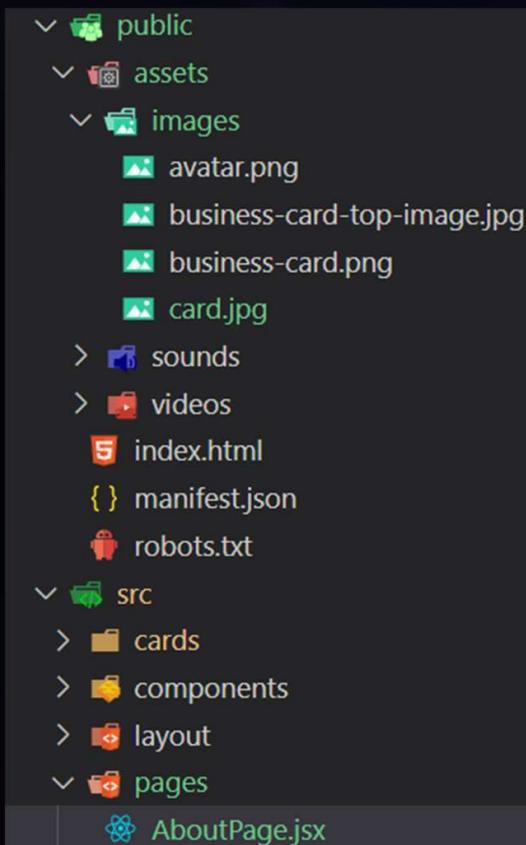
Using the Public Folder

<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 תיקיית **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 export default AboutPage;

```

AboutPage.jsx

ניצור דף אודות שישתמש בתמונה שנשמר
בתיקית public

- ניצור קומפוננט בשם AboutPage שתחזיר
- נשתמש בקומפוננט Container כדי לתת רווח
לתוכן בגדי מסך שונים
- אני מציב את הקומפוננט PageHeader עם
הcottroat המתאימות
- ניצור אזור רספונסיבי באמצעות הקומפוננט Grid

- נקבע כי החלק של המיל יתרפое את כל רווח המסן
בגודל מסך קטן ויתפое 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 ונמוך אליו
- מחוזצת התווים בערך של מאפיין src תחיל בסימן
סלאש / כדי ש react תחליל את החיפוש של בתקית
public ולאחר מכן נשלים את הנתיב עד לתמונה
המבוקשת

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! Quibusdam pariarur 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. Elus 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 pariarur neque porro, accusantium consequuntur, exercitationem quisquam? Itaque praesentium beatae consectetur, quisquam facilis qui laboriosam voluptate maxime cupiditate voluptas et nisi?



התוצאות בדף

- ניתן לראות כי בהתאם לתבנית שקבענו
- אנחנו רואים את כוורות הדף
 - את החלק המועד לטקס המסביר על האפליקציה
 - ורואים 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=dpw
s4bM&t=2EHDh9](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 שלහן ניצור counter בעזרת hook

- נחלץ את מетодת useState מספרית react

• ניצור קומפוננט בשם SetCounter

- נעשה array destructor לערך שיחזור אליו מהפעלת useState עם הפורמט 0 וナルץ ממנו את המפתחות:

- counter – שם המשתנה
- setCounter – מטודה שאחראית על שינוי ערכו של המשתנה

• נציב את הערך של המשתנה counter

• ניצור כפתור שבלחיצה עליו יפעיל פונקציה אונומית

• הֆונקציה האונומית תפעיל את מетодת setCounter שჩילצנו

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

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

• הפעם אנחנו מעבירים לפונקציה setCounter את הערך שאנו מעוניינים שייהי למשתנה counter מוביל להעיבר פונקציה אונומית כי אין אנחנו לא מ變ים את הערך על בסיס הערך הקודם אלא מוסףים את הערך במספר אף

```
SetCounter.jsx U ●  
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           >  
29             reset counter  
30           </button>  
31         </div>  
32       </div>  
33     );  
34   };
```

useState with Function

בדוגמה שלහן יצרנו 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 מетодת 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 בפרמטר 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 שנחלץ על הכפטור נפעיל את הפונקציה שיצרנו עם הערך המתאים

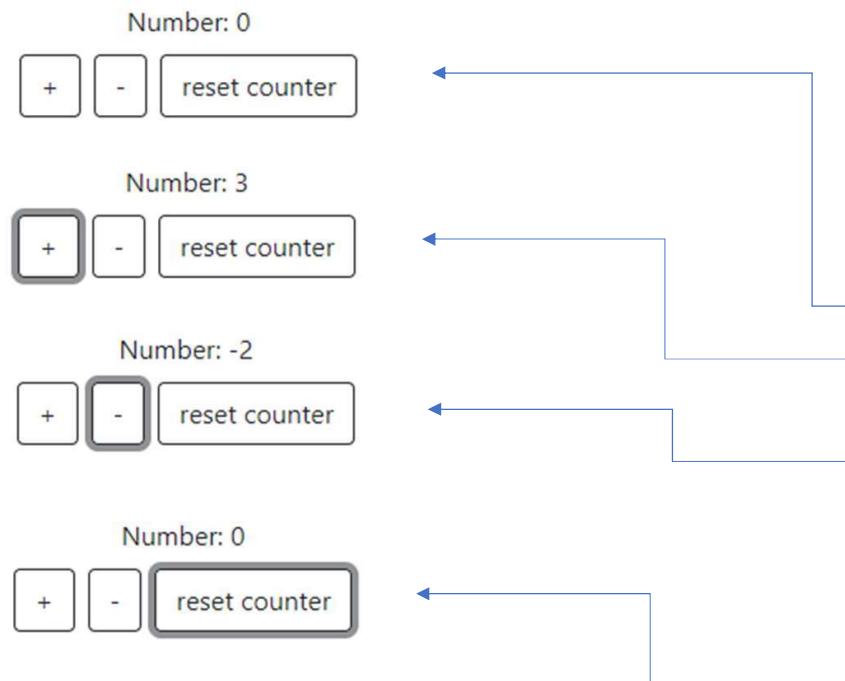
```
SetFunction.jsx

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 בדפסן

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

- בתחילת הוא מאפס
- כשלוחצים על cptור + הוא מעלה את המספר
- כשלוחצים על cptור - הוא מוריד את המספר
- כשלוחצים על cptור reset counter הוא מאפס את המספר



```
SetObject.jsx ✘  
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;
```

useState with objects

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

- נחלץ את מетодת useState מספרית react
- ניצור קומפוננט בשם SetObject
- ניצור קבוע בשם initialValue שערך יהיה אובייקט עם המפתחות first last
- נעשה array destructor למערך שיחזור אליו מהפעלת useState עם הparameter initialValue ונחלץ ממנו את המפתחות:
- name – שם המשתנה
- setName – מטודה שאחראית על שינוי ערכו של המשתנה
- נציב את ערכיו של המשתנה name כך שיוצגו לגולש
- ניצור אלמנטים מסוג submit שכשיכניםו אליהם ערך הינו תפעלנה פונקציה אונומית שתקבל את האירוע
- תפעיל את מטודת setName כאשר בפרמטר נעביר לה אובייקט אליו אנו מעתיקים את כל המפתחות מאובייקט name ומשנים את הערך של המפתח הרלוונטי בהתאם לנ נתונים שהוכנו לאלמנט input

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 נוצר טופס שיבקש מהמשתמש את השם הפרטיו 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 מחזוזות תווים
- email – מסוג מחזוזת תווים

- נעשה array destructor לערך שיחזור אליו מהפעלת useState עם הparameter 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 לגולש
- ניצור אלמנט מסוג input שכאשר יוכנסו אליו נתונים הוא יפעיל פונקציה אוניבריאט שתקבל את האירוע ותפעיל את מетодת setUser וגעיר לה אובייקט ש:

- געтир לתוכו את מפתחות האובייקט user
- נעשה השמה למפתח name ונשווה את ערכו לאובייקט שאליו געתיך את המפתחות של האובייקט user.name
- נעשה השמה למפתח first בתוך הערך של אובייקט name כך שנערך יהיה שווה לערך שיוכנס לאלמנט input (e.target.value)
- ניצור אלמנט מסוג input שיעתיך את המפתחות של user ויעשה שמה למפתח email

```
SetComplexObject.jsx X
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

הנתה התשתיית

- 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 מетодת useState עם הקבוע task setTask את שיצרנו ונחלץ ממנו את tasks setTasks עם ריק ונחלץ ממנו את createNewTask בשם פונקציה
- שתקבל אירוע
- תעצור את ההתקנות הדיפולטיבית של שליחת הטופס על ידי הכפתור
- שתפעיל את מетодת setTasks כشارוגמנט נטעין את מערך המשימות ונוסיף את המשימה
- נוצר את הפונקציה ונאפס את המשימה על ידי הפעלת מетодת setTask עם הקבוע של הערכים הראשונים

```
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

- ניצור טופו
- בחלקן העליון נציג לגולש את המשימה
- ניצור כפטור ש:

• הוא יהיה מנותר אם לא יהיה ערך ב מפתח todo
של אובייקט task

• בלחיצה על הכפטור הוא יפעיל את מетодת
createNewTask

- ניצור input

• שתזין לארוע onChange ותפעיל פונקציה
אנונימית שתקבל את האירוע

• תפעיל את מетодת setTask כאשר בארגומנט
נתיק את המפתחות מתוך אובייקט task וונעשה
השמה למפתח todo עם מה שיוצג בתוך
האלמנט

• הערך של האלמנט יהיה הערך של המפתח
task.todo

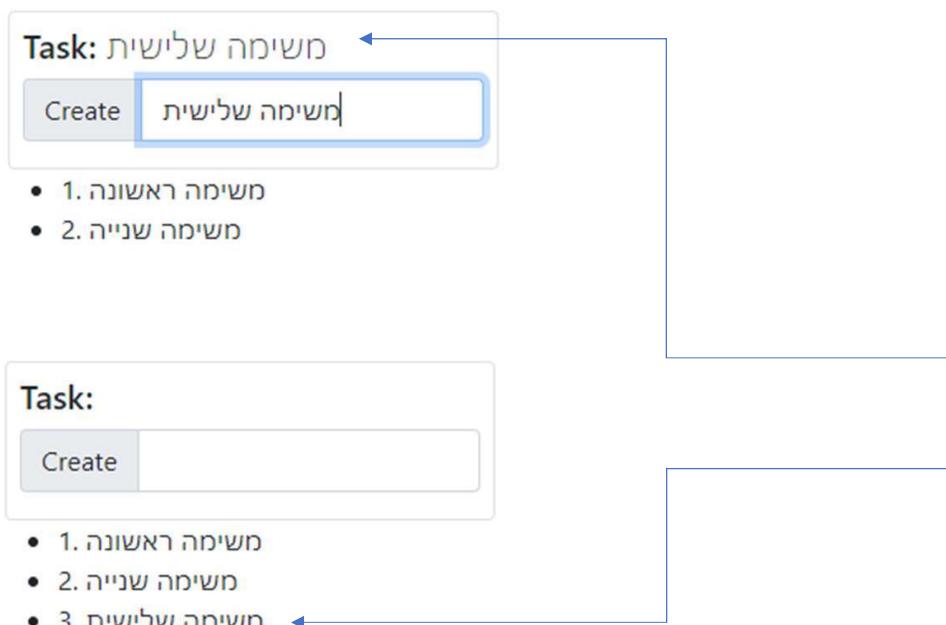
- ניצור רשימה ש:

• על כל איבר במערך tasks היא תיצור רשומה עם מספר
עם פרטי המשימה

```
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 };
```

התוצאות בדף

- ניתן לראות כי בהתאם למבנה שקבענו
- כשמלאים את הכתוב בשדה הטקסט מוצג בראש הטופס
 - ככלוחצים על הכפתור המשימה מוצגת כרשומה והשדה מתנתקה



useState משימת



Hooks-Sandbox

- צור קומפוננט בשם SetPost
 - יבא את useState ממודול
 - צור פונקציה בשם SetPost (קומפוננט)
 - צור קבוע בשם INITIAL_POST שערך יהיה אובייקט עם המאפיינות:
 - title – מסוג מחרוזת תווים
 - subtitle – מסוג מחרוזת תווים
 - author – מסוג מחרוזת תווים
 - createdAt – מחרוזת תווים
 - הפעיל את מетодת useState בעמ"ם כאשר בפעם הראשונה עם הארגומנט INITIAL_POST וחלץ מהערך שחוזר את post setPost
 - בפעם השנייה עם מערך ריק כארגומנט וחלץ מהערך שחוזר מהפעלת הפונקציה את posts setPosts
 - הצג גולש:
 - טופס שבתוכו
 - צור שלוש אלמנטים מסוג Button שהנתונים שיוצנו בתוכם יהיו מקושרים לערכים של שלושת המאפיינות הראשונות של אובייקט post החדש (כמו שהודגם בשקפים הקודמים)
 - צור כפתור שלחיצה עליו תכניס פווט חדש למערך הפואטים
 - טבלה שתציג רק אם יש פואטים במערך הפואטים

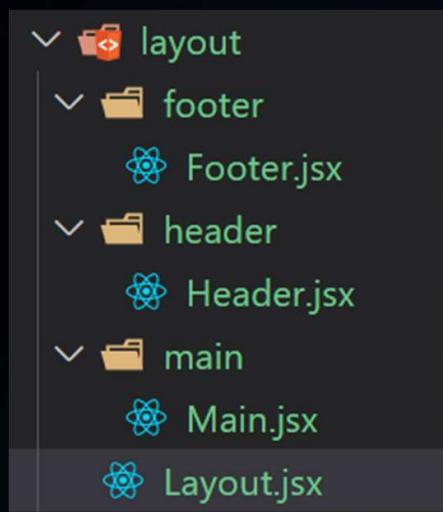
Layout

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` שיציג את התוכן שלה בתבנית הכללית



Footer.jsx



U

X

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 ניצור את הקומפוננט כך שתציג את התוכן שלו בתבנית הכללית

Main.jsx

קומponent זה תריה אחראית על תצוגת התוכן

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

```
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 ישתנה בהתאם לדף שהגולש נמצא בתוכו

- הקומפוננט קיבל בפתח של אובייקט הפורס את המילה השמורה children
- עטוף את האלמנטים בקומפוננט באמצעות React.Fragment

- נציג לגולש את הקומפוננטות:
 - Header – שתכלול תפריט ניווט בעtid
 - Main – אליו נעביר את התוכן של הדף שברצוננו להציג
 - Footer – תכלול תפריט ניווט בעtid לוגו וזכויות יוצרים

אני מזמין בעזרת ספריית propTypes שהקומפוננט אכן מקבל אלמנט של React.children שבאובייקט הפורס

כרגע התוכן של הדף הוא עדין סטטי עד שנלמד בהמשך המציג להשתמש בניירובים!

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

- נעתוף את הקומפוננט `CardsPage` כרך שיצא לנו תוכן הדף בקומפוננט `Layout` כרך שיצא לנו תוכן הדף `Header Footer` עם `CardsPage` בלבד

למעשה אנו מעבירים לקומפוננט `Layout` בפתח באובייקט ה – `props` את הקומפוננט `CardsPage` שהוא אלמנט של `React` בתוך הקומפוננט `Layout` מוצב הקומפוננט `main` שמקבל גם הוא את אלמנט ה `React` בפתח באובייקט ה – `props` ומציג אותו

התוצאה בדף

ניתן לראות כי בהתאם לבניית
שקבענו מוצג גולש

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

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



second

subtitle

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



third

subtitle

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



header

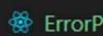
footer

ErrorPage.jsx

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

! יש לעبور למסך וUI לחלק של navigation בטרם ממשיכים עם מצגת זאת





ErrorPage.jsx



```
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 `src/pages/ErrorPage.jsx`

- ניצור את הקומפוננט `ErrorPage`
- הקומפוננט תציג לגולש כותרות מתאימות
- טקסט מתאים שישביר 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

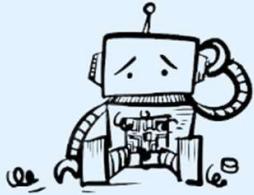
Search  

Error 404

Page not found

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

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



 About  Favorites  My Cards

התוצאה בדף

ניתן לראות כי בהתאם לתבנית שקבענו

- אנחנו רואים את כותרות הדף
- את החלק המועד לטקו המסביר על האפליקציה
- ורואים את התמונה של הלקוח

React Router Dom

ניתובים באפליקציה שהיא Single Page Application

<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

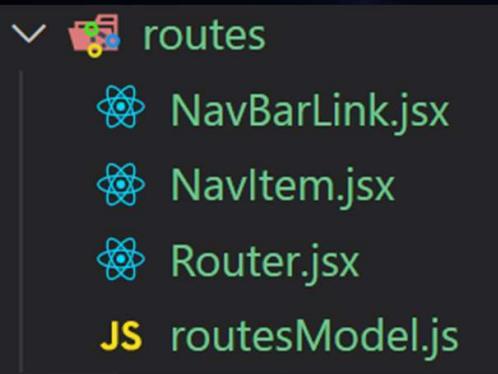
Easy to learn and apply



Installation

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

הכנות תשתית



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

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

routesModel.js

מודול זה ישתמש בשפה משותפת לכל
שמות הLINKים 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

The screenshot shows a code editor window with the file 'routeModel.js' open. The file path is shown as 'client > src > routes > routeModel.js > ...'. The code itself is as follows:

```
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` שהרואי על
ניתובים



Router.jsx

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;
```

קובץ זה ינהל את תצוגת הדפים לגולש

- נייבא את הקומפוננטות react-router-dom Route

- ניצור קומפוננט בשם Router

- נעתוף את קומפוננטות ה- Route – בקומפוננט Routes

- כל קומפוננט מסווג Route מקבל בשלב זה שני מאפיינים:

- path – כתובת URL

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

- הקומפוננט האחרון שנzieיב מקבל במאפיין path את הכתובת "*" וצר היא תירט את כל הכתובות שלא נתפסו בניתובים הקודמים ותציג Komponont ErrorPage

! יש חשיבות לסדר הקומפוננטות של Route



BrowserRouter

קומponeנט של react-router-dom שתנתב לדף
המתאים לפי הלוגיקה הרשומה בקומponeנט
Routes



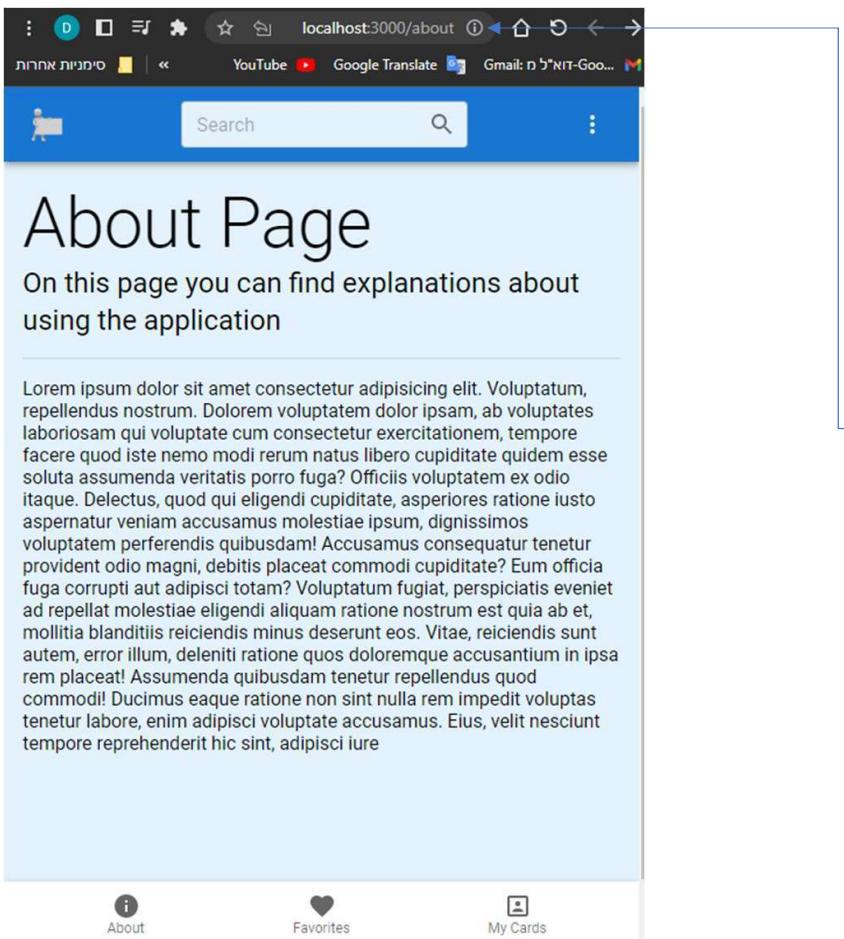
BrowserRouter

- ניבא את BrowserRouter מספרית react-router-dom
- געטוֹפַ אֶת הָאֱלִיקְזִיה שָׁלֵנוּ בְּקֻומְפּוֹנְט BrowserRouter

```
JS App.js M X
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;
```

התוצאות בדף

אם נ קיש בשורת הכתובת לאחר כתובות
ה DNS והפורט את כתובות ה URL של
[/about](#) נ עבור לדף אודוות





Link & NavLink

קומפוננט של `react-router-dom` שתיקבע את כתובות 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
- יצור קומponent בשם NavBarLink
- שתקבל באובייקט הפרופס את המפתחות:
 - URL – to
 - צבע – color – הצבע הכתוב של האלמנט שנעוטף
 - children – אלמנט תעוטף שהקומponent תעוטף
- הקומponent תחזיר את הקומponent Link שיבאנו כאשר מאפיין to יהיה שווהurre ל מפתח to מאובייקט הפרופס, וקבע כי המאפיין style יקבל את הצבע שנעביר לאובייקט הפרופס וקבע שלא יהיה קו תחתון לאלמנט שהוא עוטפיים
- נשתמש בספרית PropTypes כדי לוודא שהקומponent תקבל את כל מה שהיא צריכה
- קבע את הצבע הלבן כערך הדיפוליטיבי של צבע הטקסט

```
Logo.jsx U X  
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
- נוצר קומponeנט בשם Logo שתחzier

• נעצוף את הקומponeנט NavBarLink
בקומponeנט NAVBarLink שיצרנו ונעביר לו
בפרמטר את כתובות ה – URL המתאימה
מתוך אובייקט ROUTES

- נקבע עיצוב מיוחד ללוגו

NavItem.jsx

```
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;
```

קומפוננט המועדת לשימוש בתפריט ניוט

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

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

- געטוף את קומפוננט הcptor בקומפוננט NavBarLink שתקבל במאפיינים שלה את כתובות ה – URL אליה היא צריכה להעביר את הגולש ואת הצבע של הכתוב על הcptor
- נודא שהקומפוננט מקבלת את כל המאפיינים שהיא צריכה כדי שהלוגיקה שלה תפעל כראוי בעזרת PropType

```
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 }
```

LeftNavigation.jsx

שימוש בkomponent NavItem שיצרנו

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

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

התוצאות בדף

עכשו הlientים שלנו בתפריט הניווט
עובדים, לחיצה עליהם מעביר את הגולש
לדף המבוקש

משימת react-router-dom



Business-cards-app

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

URL	ROUTES	label	lienק / link	מונחים
/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 שערך יהיה הערך שייחסור
- שהערך שלו יהיה פונקציה שמקבלת בפרמטר to מסוג של מחזורת תווים
- ותפעיל את navigate עם מחזורת התווים שהועברה לפונקציה בפרמטר to
- בкомпонент שלחיצה עליה נרצה להעביר את הגולש לדף אחר נציג לאירוע onClick שיפעל פונקציה אוניברלית שתפעיל את פונקציית navigate 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={InfoIcon} />
23           onClick={() => navigateTo(ROUTES.ABOUT)}
24         />
25         <BottomNavigationAction
26           label="Favorites"
27           icon={FavoriteIcon} />
28           onClick={() => navigateTo(ROUTES.FAV_CARDS)}
29         />
30         <BottomNavigationAction
31           label="My Cards"
32           icon={PortraitIcon} />
33           onClick={() => navigateTo(ROUTES.MY_CARDS)} ←
34         />
35       </BottomNavigation>
36     </Paper>
37   );
38 }

39 export default Footer;
```



Navigate

קומפוננט של `react-router-dom` שתסייע 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` מתוך `react-router-dom`
- ניבא את `ROUTES`
- ניצור משתנה בשם `user` ונשווה את הערך שלו פעם ל – `null` ופעם ל – `true`
- ניצור התניה ואם הערך של `user` הוא לא פולסיבי נחזיר מהקומפוננט את הקומפונט `Navigate` של `react-router-dom` ובעזרת המאפיין `replace` נחליף את כתובת URL לכתובת הרשומה במאפיין `to`
- אם לא נכנס לתניה נחזיר מהקומפוננט את דף ההתחברות

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  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;
```

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

משימת
Navigate



Business-cards-app

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



useParams

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



```
CardDetailsPage.jsx ✘ ×  
client > src > cards > pages > CardDetailsPage.jsx > ...  
2 import { useParams } from "react-router-dom"; ←  
3 import Container from "@mui/material/Container";  
4 import PageHeader from "../../components/PageHeader";  
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;
```

CardDetailsPage.jsx

- ניצור את הנתיב src/cards/pages/CardDetailsPage.jsx
- ניבא את useParams מספריית react-router-dom
- נחלץ מהאובייקט שיחזור מהפעלת המethode useParams את מפתח id
- נציג לגולש את تعدות הזהות של הלקוח שאת פרטיו נציג בקומponent

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

- נוציא את המפתח CARD_INFO

Router.jsx

- נוצר Route חדש שיעביר אותנו לkomponent CardDetailsPage במידה ושורת הכתובות משתנה לכתחובת ROUTES.CARD_INFO אשר בפתח שורת הכתובות גם נצפה לקבל בשורה 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

בקומponent זה ניצור את הלוגיקהacr
שבלחיצה על כרטיס עסקי ספציפי וועבר
לדף עם הפרטים על הלקוח

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

- נאזרן ללחיצה על איזור זה בכרטיס
נפעיל את מетодת navigate כאשר
בפרמטר נעביר לה את כתובת URL
בנוסף את תעודת הזיהות 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   );
};
```

The screenshot shows a web application interface. At the top is a blue header bar with a user icon, a search bar containing the text 'Search', and a three-dot menu icon. Below the header, the main content area has a light blue background. The title 'Business Details' is displayed prominently. A sub-header below it reads 'Here you can find more details about the business'. A horizontal line separates this from the card details. Below the line, the text 'Details of card: 63765801e20ed868a69a62c4' is shown, followed by a left-pointing arrow. At the bottom of the main content area, there are three navigation icons: 'About' (info icon), 'Favorites' (heart icon), and 'My Cards' (card icon).

התווצה בדף

לחיצה על כרטיס הובילו אותנו לדף CardDetails.jsx.

ומוצג לנו תעודת זהות של הכרטיס בתוכה אותה הקומפוננט לkerja 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

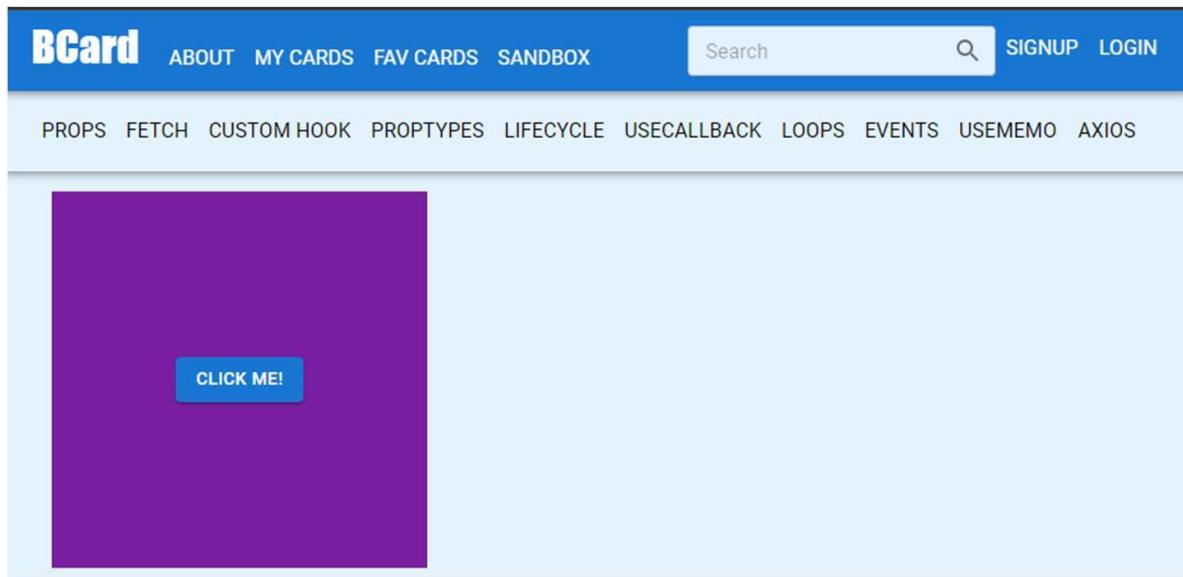
React-router-dom מותנת לנו דרך קלה ונוחה ליצור nested Route . ישן שלושה שלבים כאשר השלב הראשון הוא עטיפת Route בקומפוננט Route הראשית. בדוגמה שלהן:

- צרנו תגית סגירת לקומפוננט Route שמאפיין ה – path שלו שווה ל – “/sandbox” והקומפוננט 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

```
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;
```

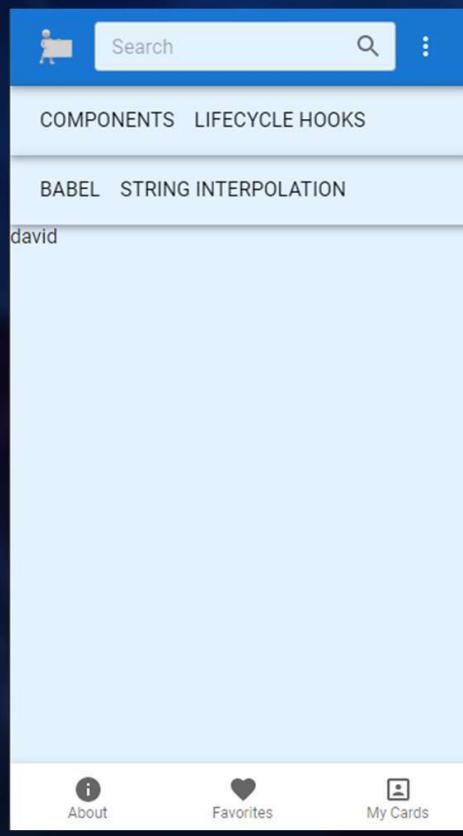
- בשלב שני ניבא את קומפוננט `Outlet` מתוכן `react-router-dom` וציב אותה במקום בו אנו רצים להציג את התוכן
- בשלב שלישי נוצר תפריט ניוט שינועט את הגולש לכתובות הרלוונטיות



התוצאה בדף

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

משימת Nested Routs



Business-cards-app

- צור טפריט ניווט בקומפוננט Sandbox קר שעל כל תיקייה בתוך תיקייה sandbox יש ליצור טפריט ניווט משלה עם התצוגה המתאימה לה כפי שモופיע בדוגמה מצד שמאל

Life Cycle Hooks

האזנה לאירועים של יצירת עדכון וחיסול קומפוננט והפעלת
לוגיקה מוגדרת בהתאם

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



Lifecycle Hooks

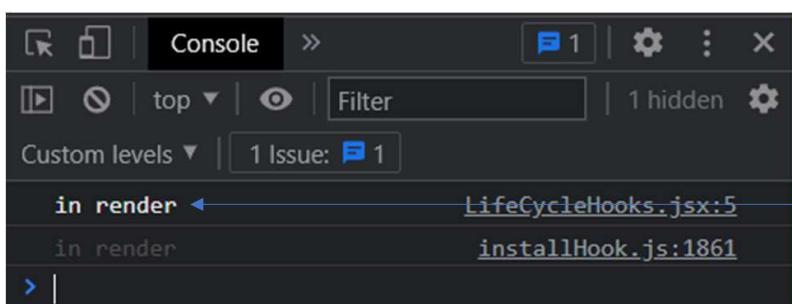


כשמשתמשים ב React Classes יש עוד נקודות במחזור החיים של הקומפוננט שניתן להאזין להם ולהפעיל מטודות !

אומרים שנקודות אלו יהיו בעtid גם ב – react hooks על כן בקישור הבא: !

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

```
LifeCycleHooks.jsx X  
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 כלומר עדכן התצוגה לגלש
- ניצור את הקומפוננט **LifeCycleHooks**
- הפונקציה תחזיר הדפסה של מחוזת תווים ואת הכתוב **LifeCycleHooks**
- התוצאה בדף
- ניתן לראות שהודפסה בקונסול מחוזת התווים שקבעו 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 setCount מהפעלת מטודה פונקציית callback אליה נעביר בארגומנט פונקציה שתופעל פעם אחת עם טיענת הקומפוננט. הפלוקציה תופיע בקונסול מחוזחת 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
 - מדפיס את ערכו של המשתנה count

התווצה בדף

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

```
Count: 1
Search : 
Custom levels ▾ | 1 Issue: 1
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
>
```



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 אשר קראו לפונקציה

- נשנה את התוכן של הקומפוננט LifeCycleHooks שיצרנו
- נפעיל את מетодת useState עם הספירה אפס כפרמטר ונהלץ מהמערך שיחזור את המפתחות count setCount

- ניצור אפקט בעזרת useEffect ש:
- בפרמטר הראשון – יקבל פונקציית call back אונימית שתפעיל את הקוד הבא:

- תדפיס את הזמן שמורכב מהשעה והמייל' שניות שבה קוראים לפונקציה
- תעלה את המשתנה count באחד בעזרת מетодת setCount
- בפרמטר השני – נעביר מערך ריק על מנת שפונקציה זאת תפעל רק פעם אחת

- הפונקציה תחזיר:
- הדפסה בקונסול של הזמן שבה 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 two instances of the React DevTools Components tab. Each instance displays a component tree with a 'Count' prop. In the top instance, 'Count' is 2, and in the bottom instance, it is 3. To the right of each screenshot, there is a list of log entries from the React DevTools backend. Arrows point from the text descriptions in the list to the specific log entries in the devtools interface.

Top Screenshot (Count: 2):

- 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

Bottom Screenshot (Count: 3):

- 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 שהkomponent NetUntet נטען בפעם הראשונה בשעה 20:51:15.992
- מעבר מס' מיל שניות היא מבצעת את הלוגיקה שבתוך useEffect והמשתנה count עולה באחד (במצב של development ההוק של strict mode עושה ריצה אחת לפני הריצה המרכזית ולכן רואים שהוא הulta את המספר בשניים ולא באחד כפי שהיינו מצפים)
- הkomponent NetUntet מוחדר בשעה 20:51:17.18 בעקבות השינויים במשתנה count
- כשלוחצים על כפתור שמעליה ספרה ניתן לראות שהkomponent NetUntet מוחדר אך useEffect לא מגיב לשינוי

useEffect with dependencies

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

- נוצר שני משתנים בעזרות useState

- count
- num

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

- עבריר למערך התלוויות של methodת useEffect רק את המשתנה num כך שהיא תעקוב 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;
```

התוצאות בדף

The figure consists of three vertically stacked screenshots of a browser's developer tools, specifically focusing on the React DevTools components tab and the console tab.

- Top Screenshot (Count: 0):** Shows the component tree with two components: "Count: 0" and "Num: 0". The "Count: 0" component has a plus and minus button. The console log shows the following entries:
 - in render 14:39:36
 - in render 14:39:36
 - in useEffect: 14:39:36.561
 - in useEffect: 14:39:36.562
 - LifeCycleHooks.jsx:59 ← react_devtools_backend.js:4066
 - LifeCycleHooks.jsx:55 ← LifeCycleHooks.jsx:55
- Middle Screenshot (Count: 1):** Shows the component tree with "Count: 1" and "Num: 0". The "Count: 1" component has a plus and minus button. The console log shows:
 - in render 14:39:36
 - in render 14:39:36
 - in useEffect: 14:39:36.561
 - in useEffect: 14:39:36.562
 - in render 14:40:19
 - in render 14:40:19
 - LifeCycleHooks.jsx:59 ← react_devtools_backend.js:4066
- Bottom Screenshot (Count: 1):** Shows the component tree with "Count: 1" and "Num: 1". The "Count: 1" component has a plus and minus button. The "Num: 1" component also has a plus and minus button. The console log shows:
 - in render 14:39:36
 - in render 14:39:36
 - in useEffect: 14:39:36.561
 - in useEffect: 14:39:36.562
 - in render 14:40:19
 - in render 14:40:19
 - in render 14:40:36
 - in render 14:40:36
 - in useEffect: 14:40:36.828
 - LifeCycleHooks.jsx:59 ← react_devtools_backend.js:4066
 - LifeCycleHooks.jsx:55 ← LifeCycleHooks.jsx:55

- ניתן לראות כי הקומponent נטענה בשעה 14:39:36
- קצת אחרי הפעלה מתודת `useEffect`
- ובגלל שלא היה שינוי באף משתנה דינמי שבאובייקט ה `state` הקומponent לא נטענה מחדש
- כאשר אנו לוחצים על הכפתור שמשפיע על משתנה `count` (שלא שמננו אותו במערך התלויות של `useEffect`) אנו רואים כי הקומponent נטענת מחדש אולם `useEffect` לא פועל
- אולם כאשר אנו משים את ערכו של המשתנה הזה שכן נמצא במערך התלויות גם הקומponent נטענת מחדש וגם `useEffect` מופעל.

useEffect as componentWillUnmount

ונכל לקרוא ל useEffect רגע לפני שהкомпонент מתחלס וונכל להפעיל את הלוגיקה שלו בנקודה הזאת

- ניצור את מетодת useEffect האנונימית בתוכה תדפיס בקונסול מחרוזת תווים עם השעה שבה useEffect הופעלה
- מה שיקרה לאחר המילה return יקרה רגע לפני שהкомпонент מתחלס וידפס ליקונסול 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

in LifeCycleHooks

```
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
```

1 Issue: 1

Error
404
Page not found

```
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
```

1 Issue: 1

ניתן לראות שכאשר אנו נכנסים לkomponent

- komponent נתענת
- יש ריצה מקדימה על useEffect כולל על מה שהפונקציה מחזירה
- לאחר מכן יש את ביצוע הקוד ב - useEffect

• רגע לפני שאנו עוברים לkomponent אחר מבוצע הקוד שמחזר מפונקציית useEffect

```

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 export default LifeCycleHooks;

```

useEffect as componentDidUpdate

אם לא נעביר למטרת useEffect פרמטר שני היא תפעל בכל פעם שתהיה טעינה חדשה של הדף בלי קשר למשתנה ספציפי שהשתנה

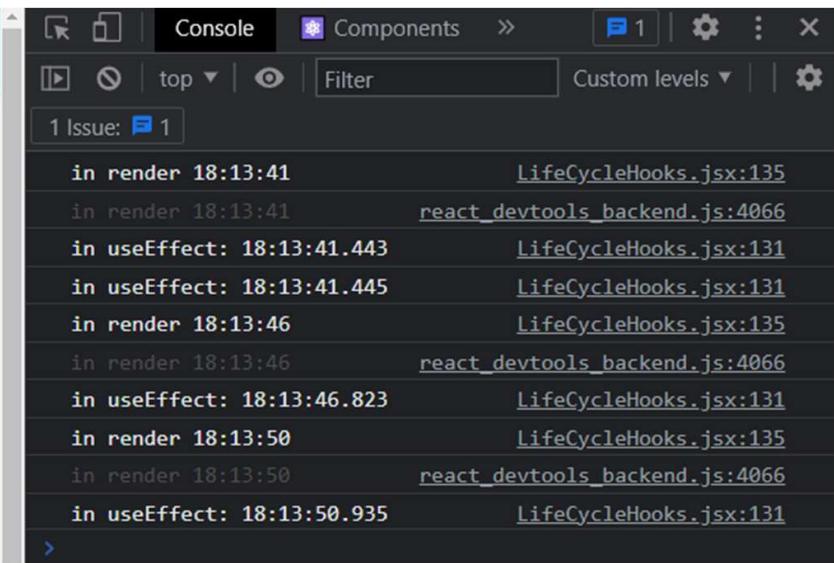
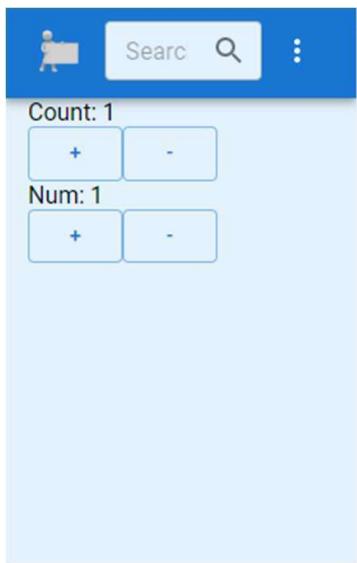
- נחזיר למשתנים הדינמיים count num炊 כרגע שינוי בהם יטען מחדש את הדף בעקבות מטרת useState

- הפעם לא נעביר פרמטר שני למטרת useEffect ונגדי רשותה היא תופעל היא תמיד בקונסול את השעה שהפעילו אותה

- הפקציה מחזירה כפורים ותצוגה של המשתנים הדינמיים

התווצה בדף

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



The screenshot shows the React DevTools interface. On the left, the Components tab displays a component tree with a yellow warning icon. On the right, the Console tab shows the following log entries:

```
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 - state של ה - counter

- ניבא את useState
- ניצור את הפונקציה useCounter שתתקבל בפרמטר initialCount מוסף של מספר עם ערך דיפולטיבי של אפס
- נפעיל את מטודת useState עם initialCount כפרמטר ונחלץ מהמערך שיחזר את המפתחות count setCount initialCount שתעלה את ערכו של count באחד
- ניצור את מטודת increment שתוריד את ערכו של count באחד
- ניצור את מטודת decrement שתוריד את ערכו של count באחד
- ניצור את מטודת reset שתאפשר את initialCount לערך של count
- נחזיר מהפונקציה מערך עם המשתנה count ושאר הפונקציות שיצרנו
- נזודא בעזרת סדריית propTypes שאנו מקבלים בפרמטר של הפונקציה מספר.

שימוש ב custom hook

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

- ניבא את `useCounter`
- נפעיל את `useCounter` ונחלץ ממנו את האיברים במערך שהוא מחייב
- נציג את ערכו של המשתנה `count`
- בלחיצה על הcptor נפעיל את Method `increment`
- בלחיצה על הcptor נפעיל את Method `decrement`
- בלחיצה על הcptor נפעיל את Method `reset`

```
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;
```

התווצה בדף

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

Count: 1

INCREMENT DECREMENT RESET

משימת custom hook



Business-cards-app

- צור custom hook בשם `useName`
- הhook מקבל פרמטר `initialName`
- הוא יחלץ מהhook `useState` את המשתנה `name` וmethod'ודה `setName`
- יצא מהmethod'ודה אובייקט עם המשתנה והmethod'ודה שיצרת
- השתמש בהook `useName` בкомпонент בשם `CustomName.jsx`
- יבא את `useName` חלץ ממנו את המפתחות
- השתמש באlement מסוג `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





Q 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

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

- ניצור קומponent בשם `ButtonComp`
 - הקומponent קיבל מפתחות לאובייקט `props` `handleClick` מסוג פונקציה ו- `children` מסוג מחוזת תווים
 - היא תדפיס בקונסול מחוזת תווים
 - היא תחזיר כפטור של `mui` שלחיצה עליו תפעיל את Method `onClick`
- נוודא בעזרת ספרית `propTypes` שהפונקציה `handleClick` מקבלים בפרמטר של הפונקציה מספר.

```
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;
```

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

UseCallBackComp

- נפעיל פעמיים את מетодת useState ונהלץ ממנה את המשתנים והmethodות
- ניצור קבוע בשם incrementAge שווה ערך לפונקציה אונומית שתפעיל את METHOD setAge שתעלה את המשתנה age באחד
- ניצור קבוע בשם incrementHeight שווה ערך לפונקציה אונומית שתפעיל את METHOD setHeight שתעלה את המשתנה height באחד
- נציג את המשתנים age height באחד
- 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. Both screenshots show a search bar at the top with a magnifying glass icon and a dropdown menu icon. Below the search bar, there is a light blue box containing the text "Age: 1" and "Height: 0". At the bottom of this box are two buttons: "AGE" on the left and "HEIGHT" on the right. To the right of the application is a browser's developer tools Console tab. In the first screenshot, the console shows the following log entries:

```
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
```

In the second screenshot, the application state has changed to "Age: 2" and "Height: 0". The console log entries have also changed:

```
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
לי בקונסול שני הcptors הופעלו
- אולם כאשר אני לוחץ על אחד מהם
שני הcptors נטענים מחדש
- הסיבה לכך היא שכל עם
שהkomponent נטענת מחדש, react
يُصرّ على كل المنشآت التي تم تغييرها
الكونponent מחדש لأنها متعلقة
وتصدر خطأ لمراقبة التغيير.
وهي تكون ملحوظة في زر.

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 לטען מחדש רק כאשר יש שינוי כזה שיוצר את יצירת הפקציה שקומפוננט קיבלה מחדש
- NEYBA AT meno MATORE SPERIYAT react
- הקומפוננט נשארת אותו דבר בלבד
- החלק שאנו מיצאים
- געטוף את ייצוא הקומפוננט בערך שיחזור מהפעלת מטודת meno שתגיד לריקט לבדוק מטודה שהקומפוננט קיבלה באובייקט ה - props השנתנה בצויה זאת צריך להעביר את הפקציה 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

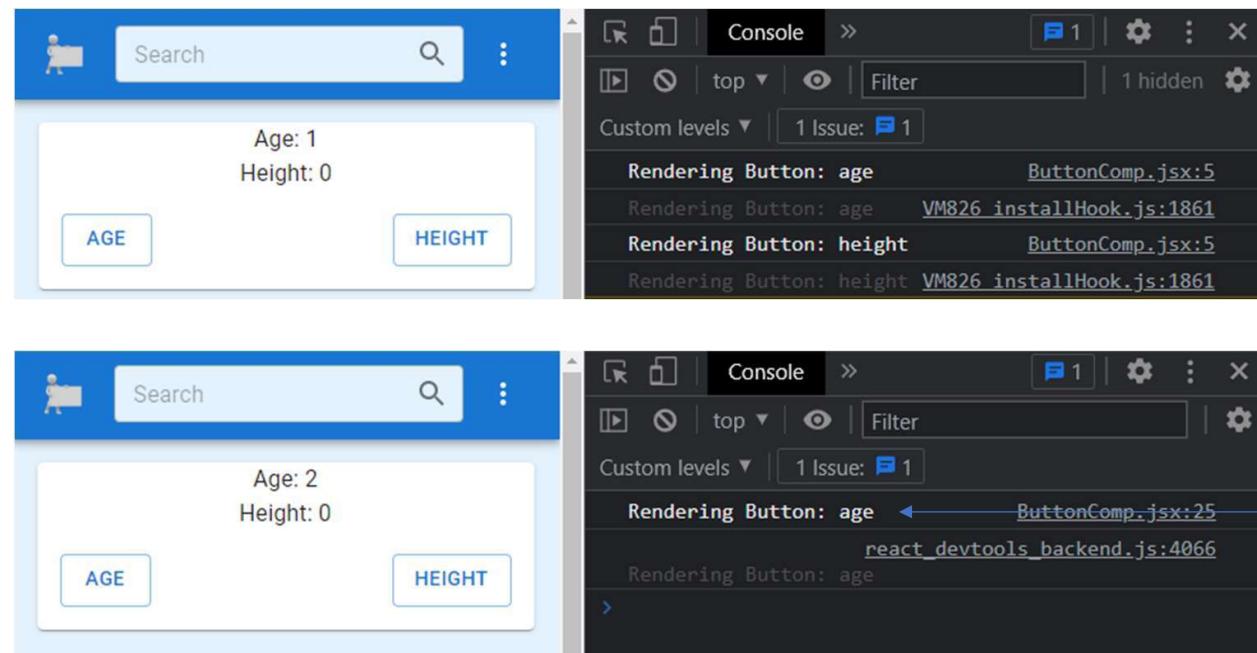
- געט את המетодות incrementAge ו incrementHeight במתודת useCallback שמקבלת שני פרמטרים:

- 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 יטען מחדש 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 נגדיר את הבעה

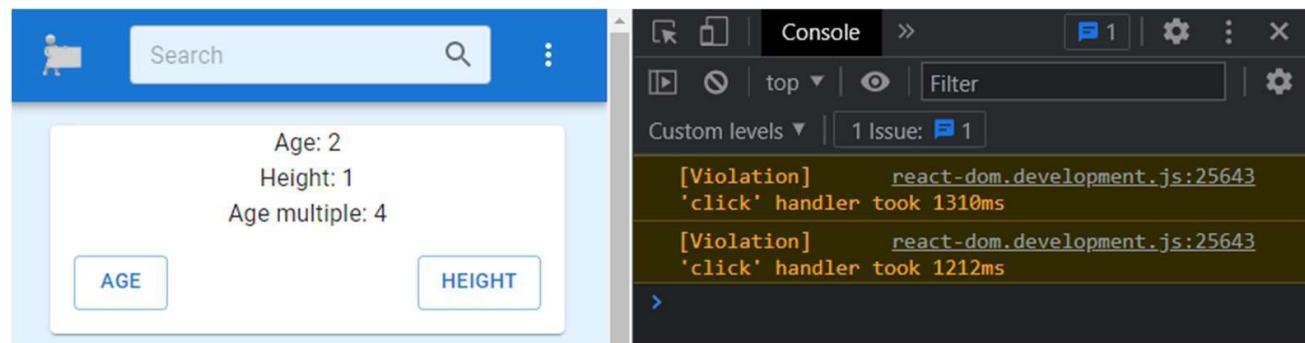
• בקומפוננט Memo

- נפעיל פעמים את מטודת useState ונהלץ ממנו את המשתנים והמתודות
- 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 שכךvr משך של לולאה מאוד ארוכה מחזירה את age כפול שתיים. מצב זה נועד כדי לדמות פונקציה "יקרה" שמעמיסה על משאבי מערכת.
- מציג את המשתנים age height
- מצב בטור ערך את הפעלת מטודת slowFunction ונוביר לה ארגומנט 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 );  
37  
38 export default UseMemo;
```

התווצה בדף

לחיצה על כל אחד מהכפתורים לוקחת לפחות 1212ms וזאת בגלל שבכל פעם שמשתנה דינامي משתנה, הקומפוננט נטענת מחדש והפונקציה slowFunction נוצרת מחדש ומופעלת



```

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;

```

UseMemo.jsx

הקוד נשאר זהה מלבד השינויים הבאים

- ניבא את `useMemo`
- נסנה את הערך של הקבוע `slowFunction` להפעלת מетодת `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` שערך הוא הערך שיחזור מMETHODת `useMemo`

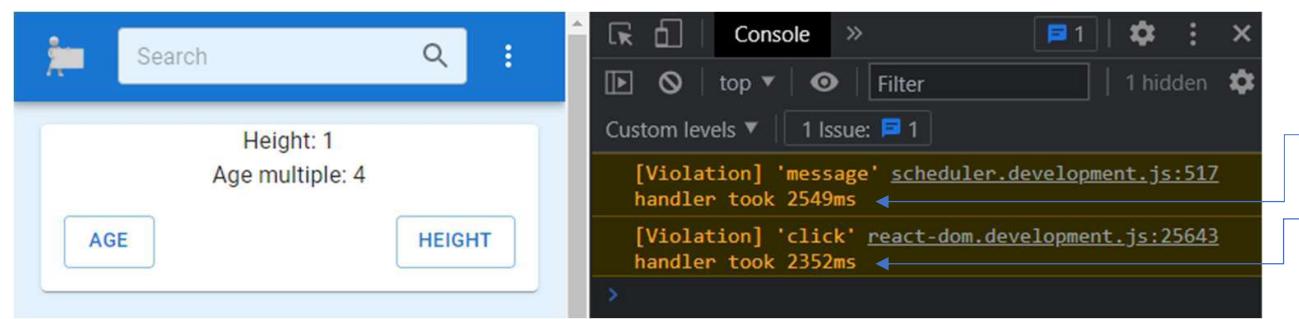
! לא ניתן להעביר לפונקציה שבתוך `useMemo` ערכים

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 ולובור על המציגת של התקנת db
mongodb





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

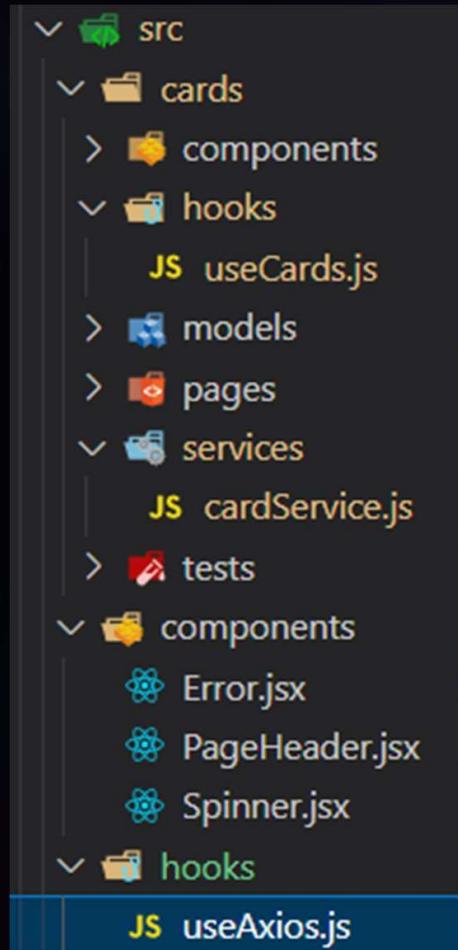
supports upload progress

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

```
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}; ←
```

- ניבא מופע מספריית axios לתוכה משתנה בשם axios
- גקשר בין המודול זהה 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 ונמצא אותו מפתח באובייקט exports שערכו היה פונקציה אסינכרונית
- שתחלץ את המשתנה data מתוך הערך שיחזור מהפעלת המטודה axios.get של axios אליה נעביר כארוגמנט את הכתובת אליה תשלח הבקשה
- נחזיר את הקבוע date
- אם תהיה שגיאה נתפוא אותה באמצעות try & catch ונחזיר בטעיות מנגןן עם הודעת השגיאהPromise.reject

Error.jsx

יצירת קומponent למקירה ויש שגיאות שמתקבלות מהשרת

- הקומponent מקבל בפרמטר errorMessage מסויים

טקסט

- תציג את הודעת השגיאה לגולש
- לצד תמונה של רובוט שבור מתוך תיקיית images



Error.jsx U X

client > src > components > Error.jsx > [?] 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 הפרופס

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

- הפקציה תחזיר קומפוננט שבתוכו האנימציה של UI M בשם CircularProgress 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 מtówר המודול שיצרנו
- נפעיל את מетодת useState שלוש פעמים ונחלץ ממנה את המשתנים cards error isPending setCards setError setPending

- נפעיל את מетодת useEffect שתפעל פעם אחת עם יצירת הקומponent ותפעיל את פונקציית ה – callback: true – לשינוי ערכו של המשתנה isPending ל – false

- תפעיל את מетодת then. שתפעיל פונקציה שתתקבל את data
- תנסה את ערכו של המשתנה isPending ל – false
- תעשה השמה למשתנה cards לערך data
- נשחרר אליה את מетодת catch. כך שאם תחזר שגיאה false – תנסה את ערכו של המשתנה isPending ל – false
- געשה השמה למשתנה error עם השגיאה שקיבלנו

נדפס לגולש:

- אם הערך של isPending הוא לא פולסיבי נציג את komponent Spinner שיצרנו
- אם הערך של error הוא לא פולסיבי נציג את komponent Error ונוביר לו errorMessage באובייקט הפרופס את השגיאה
- נבדוק ואם מערך הCARDSים ריק נציג לגולש את מחרוזת התווים הבא
- 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, setIsPending] = useState(false);
13
14   useEffect(() => {
15     setIsPending(true);
16     getCards()
17       .then(data => {
18         setIsPending(false);
19         setCards(data);
20       })
21       .catch(error => {
22         setIsPending(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
37       the parameters you entered</p>} ←
38       {cards && !!cards.length && <Cards cards={cards} />} ←
39     </Container>
40   );
41 }
42 export default CardsPage;
```



CardsFeedback.jsx

קומפוננט לחישוי מצב הלקוחות



CardsFeedback.jsx

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;
```

- ניבא את המטודות הkomponenot והמשתנים שדרושים לנו לצורך הלוגיקה של הקומוננט

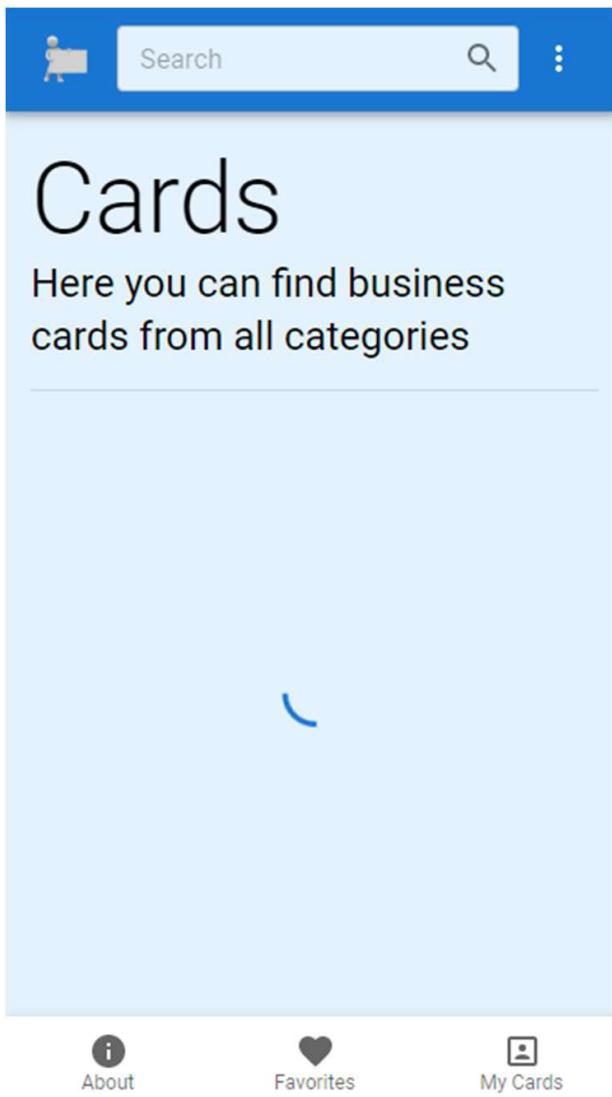
- נוצר את קומוננט CardsFeedback
 - הקומוננט קיבל את המפתחות הללו באובייקט הפרופס
 - נתנה אם המשתנה isLoading עם ערך Chivoi נציג את הקומוננט Spinner
 - נתנה אם המשתנה error עם ערך Chivoi נציג את קומוננט Error
 - נתנה אם אין או ר לערך crtisim נציג את crtisim
 - אם יש או ר לערך crtisim נציג את crtisim
 - אם הקומוננט לא עומדת באף התניה נחזיר null

- בעזרה ספירת PropTypes נודא שהקומוננט אכן מקבל בפרמטרים שלה את מה שהיא צריכה על מנת לבצע את הלוגיקה שלה

```
❶ CardsPage.jsx M X
client > src > cards > pages > 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  const onDeleteCard = () => {};
15
16  return (
17    <Container>
18      <PageHeader
19          title="Cards"
20          subtitle="Here you can find business cards from all categories"
21      />
22
23      <CardsFeedback <-->
24          isLoading={isLoading}
25          error={error}
26          cards={cards}
27          onDelete={onDeleteCard}
28      />
29      </Container>
30  );
31
32  > export default CardsPage;
```

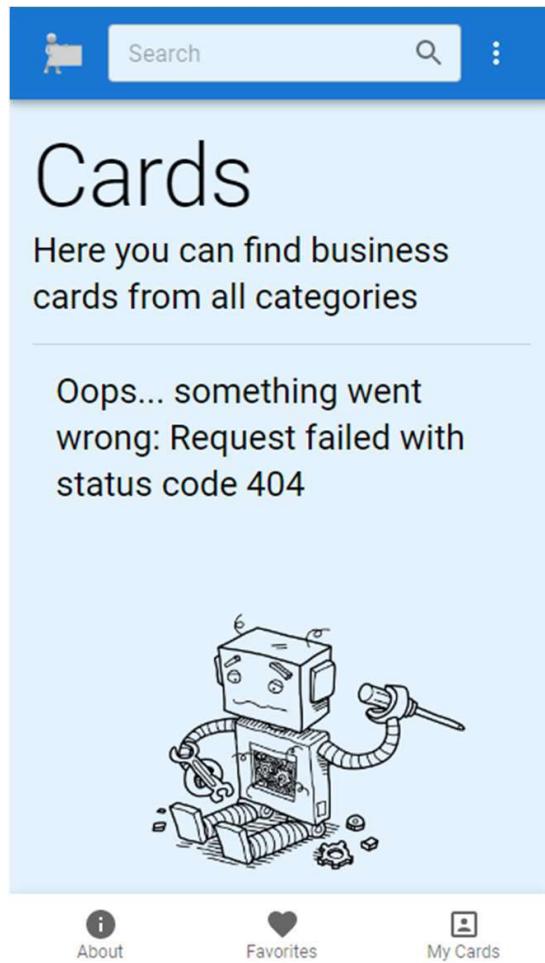
Cards.jsx

- נציב את הקומponent ונועביר לה את המשתנים שהיא מבקשת לקבל כמפתחות באובייקט ה props



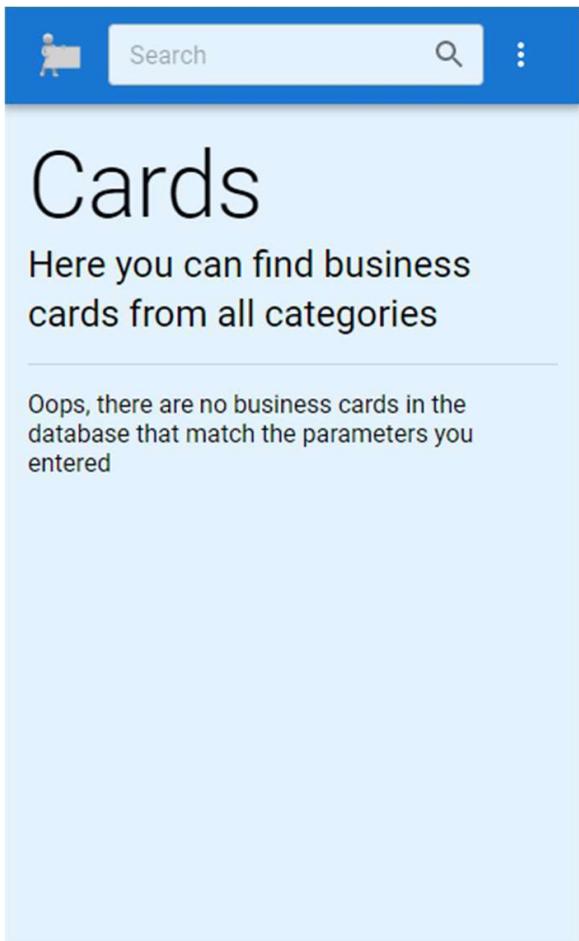
התווצה בדף

כשהקומפוננט מחייב למידע והקומפוננט spinner עובד



התווצה בדף

כשמתיקבת שגיאת שמקורה בשרת



התווצה בדף

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

התווצה בדף

כשהם ידוע ממאגר המידע מייד מתקבל
בהצלחה

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 تعدת זהות של כרטיס
- פועל את מטודת axios.get עם הנטייב בשרת שאחראי על הbattle כרטיס בודד וחלץ ממנו את מפתח data.
- החזר את data
- אם מתבלת שגיאה תפוס אותה באמצעות מנגןן try & catch

getMyCards

- פועל את מטודת axios.get עם הנטייב בשרת שאחראי על הbattle כל כרטיסי הביקור של המשתמש וחלץ ממנו את מפתח data.
- החזר את data
- אם מתבלת שגיאה תפוס אותה באמצעות מנגןן try & catch עם הודעת השגיאה

חלק ב'



• **creactCard**

- הfonקציה מקבל בפרמטר card מסוג של אובייקט המיצג כרטיס ביקור לעסק
- הפעל את מетодת axios.post כאשר בארגומנט 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 מתקבלת שגיאה תפוס אותה באמצעות מנגןן try & catch ו诙זרPromise.reject עם הודעה השגיאה

• **editCard**

- הfonקציה מקבל בפרמטר card מסוג של אובייקט המיצג כרטיס ביקור לעסק
- הפעל את מетодת axios.put כאשר בארגומנט הראשון הצב את כתובות ה – URL לעריכת כרטיס ביקור ובפרמטר השני העבר את card וחלץ ממנו את מפתח data.
- החזר את data
- אם מתקבלת שגיאה תפוס אותה באמצעות מנגןן try & catch ו诙זרPromise.reject עם הודעה השגיאה

חלק ג'



changeLikeStatus •

- הֆונקציה מקבל בפרמטר cardId מסוג של מחרוזת תווים המייצגת תעודת זהות של כרטיס ביקור
- הפעל את METHOD axios.patch והעביר בארגומנט את כתובתה ה – URL שאחראית על הוספה או הסרת ליק מכרטיס ביקור וחלץ ממנו את מפתח data.
- החזר את data מהפונקציה
- אם מתקבלת שגיאה תפוס אותה באמצעות מנגןן try & catch ו诙זרPromise.reject עם הודעה השגיאה

deleteCard •

- הֆונקציה מקבל בפרמטר cardId מסוג של מחרוזת תווים המייצגת תעודת זהות של כרטיס ביקור
- הפעל את METHOD axios.delete והעביר בארגומנט את כתובתה ה – URL שאחראית על מחיקת כרטיס ממAGER המידע וחלץ ממנו את מפתח data.
- החזר את data
- אם מתקבלת שגיאה תפוס אותה באמצעות מנגןן try & catch ו诙זרPromise.reject עם הודעה השגיאה



useCards

בגלל שדים רבים באפליקציה יהיו מעוניינים
לקבל את כרטיסי הביקור ולהשפע על תוכנם
ניתור custom hook לניהול הCARTEISIM



useCards.js

העברת הלוגיקה של ניהול הCARTEISIMS תבוצע בשלוש שלבים.
הראשון הוא יצירת custom hook בנתיב
src/cards/hooks/useCards

- ניבא את useState מ – react
- ניבא את METHOD getCards מהשירות שיצרנו
- נוצר את המETHOD useCards
- בעזרה METHOD setState ה – hook ינהל את מצב:

- cards – CARTEISI ה ביקור
- card – CARTEISI בIKOR ספציפי
- pending request – ניהול מצב isLoading
- error – ניהול שגיאות

- נוצר את הפונקציה האсинכורנית handleGetCards
- שתקבע את ערכו של המשתנה isLoading ל – false
- תפעיל את METHOD getCards ותשמר את הערך שיחזר בתוך קבוע בשם cards
- נעדכן את ערכו של המשתנה isLoading ל – חיובי
- נעדכן את המשתנה error ל – null
- נעדכן את המשתנה cards עם מערך הCARTEISIMS שהזר אליהם METHOD getCards
- במידה והייתה שגיאה היא תיתפס במנגנון try & catch ונדעך את המשתנים בהתאם

- לבסוף ניצא את המשתנים ואת METHOD 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 הפונקציות 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 שתקבל את המשתנים הבאים
 - הפונקציה תפעיל את method setLoading עם הערך loading שיתקבל במשתנה
 - תפעיל את method setCards עם הערך שיתקבל cards במשתנה
 - תפעיל את method setCard עם הערך שיתקבל card במשתנה (אם לא יתקבל ערך קבוע 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 הקומפונט ונחלץ ממנו את הערך שאמנו מעוניינים בהם
- נפעיל את מетодת `handleGetCards` בתוך `useEffect` כדי שתישלח בקשה להבאת הCARTEISIMS ממאגר המידע וכך כל הלוגיקה שכתבנו 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;
```

שימוש חלק א'



Business-cards-app

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

`handleGetCard`

- הפקציה תקבל בפרמטר `cardId` מסוג של מחוץ תווים שמייצגת תעודה זהה של כרטיס ביזור לעסוק
- היא תנסה את ערכו של המשתנה `isLoading` ל – `false`
- צור קבוע בשם `card` שערך יהיה הפעלת המethode האсинכרונית `getCard` והעביר לה בארגומנט את `cardId`
- הפעיל את מетодת `requestStatus` כך שתעדכן את המפתחות באובייקט ה – `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 שגיאה תפוס אותה באמצעות מנגן `try & catch` והפעיל את מетодת `requestStatus` כך שתעדכן את הגולש בשגיאה

שימוש בחلك ב'



handleGetMyCards •

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

handleCreateCard •

- הfonקציה 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 `createCard` והעביר לה כארוגמנט את `cardFromClient`
- הפעל את מטודת `requestStatus` כר שתעדכן את ה – state שהטעינה הסטיימה, אין הודעת שגיאה, אין כרטיסים ותעדכן את המשתנה שאחראי על כרטיס במידע שהתקבל בקבוע `card` שיצרת במידה ומתקבלת שגיאה תפוס אותה באמצעות מגנון & `try catch` והפעל את מטודת `requestStatus` כר שתעדכן את הגולש בשגיאה

שימוש בחلك ג'



handleUpdateCard •

- הfonקציה מקבל בפרמטר `cardFromClient` מסוג של אובייקט המציג כרטיס ביקור של עסק
- היא תנסה את ערכו של המשתנה `isLoading` ל – `false`
- צורך קבוע בשם `card` שערך יהיה הפעלת המטודה האסינכרונית `editCard` והעביר לה ארגומנט `cardFromClient`
- הפעל את מטודת `requestStatus` כך שתעדכן את ה – `state` שהטינה הסטיימה, אין הודעה שגיאה, אין כרטיסים ותעדכן את המשתנה שאחראי על כרטיס במידע שהתקבל בקבוע `card` שייצור
- במידה ומתקבלת שגיאה תפוא אותה באמצעות מנגןון & `try` `catch` והפעל את מטודת `requestStatus` כך שתעדכן את הגולש בשגיאה

handleDeleteCard •

- הfonקציה מקבל בפרמטר `cardId` מסוג של מחוזת תווים המציגת תעודה זהות של כרטיס ביקור לעסק
- היא תנסה את ערכו של המשתנה `isLoading` ל – `false`
- הפעל את המטודה האסינכרונית `deleteCard` והעביר לה ארגומנט את `cardId`
- במידה ומתקבלת שגיאה תפוא אותה באמצעות מנגןון & `try` `catch` והפעל את מטודת `requestStatus` כך שתעדכן את הגולש בשגיאה

שימוש בחلك ד'



• יצא את כל המethodות החדשות מהמודול

במידה ומתקבלת שגיאה תפוס אותה באמצעות מגנון `try & catch` והפעיל את מетодת `requestStatus` כך שתעדכן את ה `state` בהתאם בשגיאה

שהטינה הסתיימה, אין הודעת שגיאה, עדכן את המשתנה `cards` שהתקבל על הCARTEIS במידע שמתקבל מהקבוע שיצרת, ותעדכן את המשתנה שהתקבל על CARTEIS במידע שהתקבל בקבוע `card` שיצרת

צור קבוע בשם `cards` שערך יהיה הפעלת המטודה `getCards` האсинכרונית

צור קבוע בשם `card` שערך יהיה הפעלת המטודה `changeLikeStatus` האсинכרונית והעביר לה ארגומנט את `cardId`

הfonקציה תקבל בפרמטר `cardId` מסוג של מחרוזת תווים המציגת תעודה זהות של CARTEIS ביקור לעסן

היא תנסה את ערכו של המשתנה `isLoading` ל – `false`

• `handleLikeCard`



axios interceptors

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

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

