

React

A JavaScript Library for building user interfaces

Created By David Yakin 2023

תוכן ננין

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

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	users ApiService.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 המרכזיו אוטו מביאו
ריאקט הוא את ה - **Virtual DOM**

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

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

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



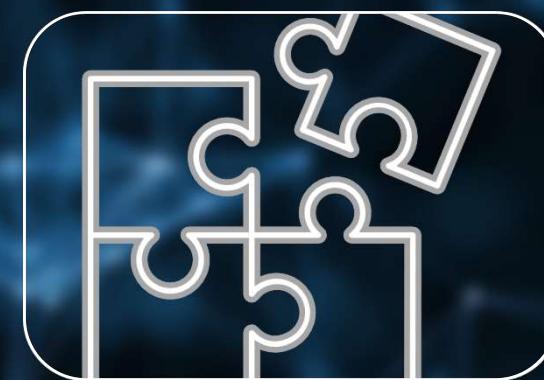
Virtual DOM implementations



State



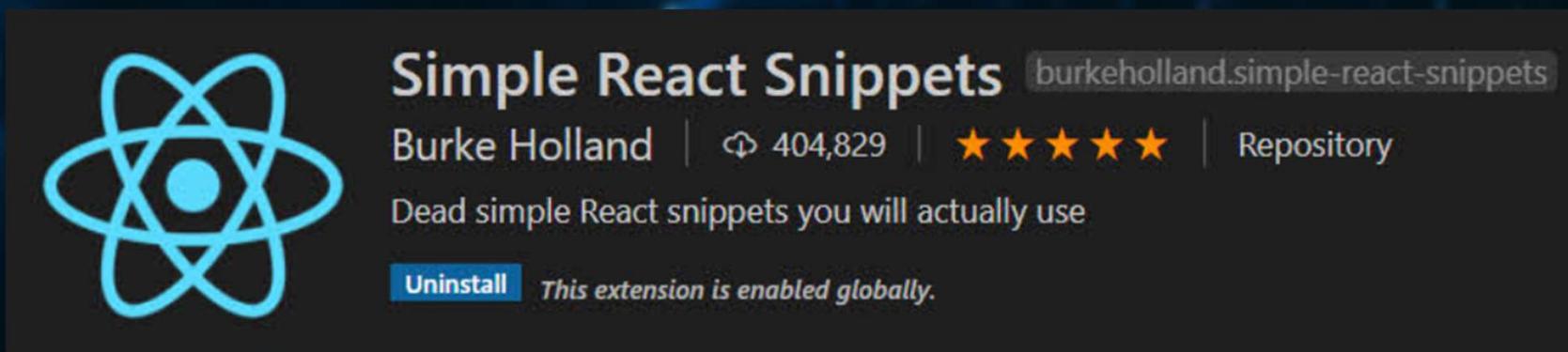
Life cycle
hooks



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

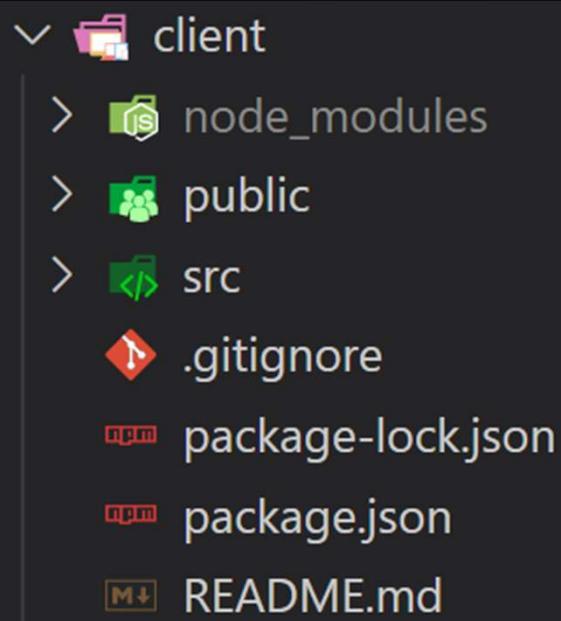
לאחר שהורדנו והתקנו את התוסף `create-react-app` נוכל להשתמש בו כדי לפתח פרויקט חדש ב - React

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

• נקיש בטרמינל את הפקודה `create-react-app client`

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

חשתית הקבצים שה-CLI יוצר

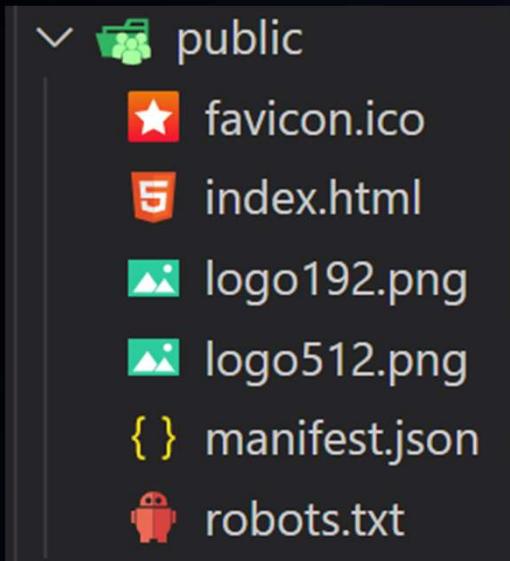


- **Node_modules** – תיקייה השומרת בתוכה את הספריות שהורדנו לפרויקט
- **public** – תיקייה השומרת את הקבצים הסטטיסטיים
- **src** - תיקייה בה רוב הקוד של הפרויקט יכתב
- **.gitignore** - קובץ קונפיגורציות של git ובתוכו ההוראות מאיילן תיקיות וקבצים להעתלם ולא להעלות ל- github
- **package.json** - קובץ קונפיגורציות 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

- קובץ התמונה שפרויקט משתמש בו באופן דיפולטיבי בפרויקט חדש
- קובץ ה-HTML המרכזי של הפרויקט
- הלוגו של ריאקט קטן
- הלוגו של ריאקט בגודל יותר
- קובץ קונפיגורציות לאפליקציה של מובייל או דסקטופ
- קובץ העוזר למנוע החיפוש google בסריקת האפליקציה

! לינק לסרטון המסביר על קובץ robots.txt
<https://www.youtube.com/watch?v=fzm-zYHjzJg>

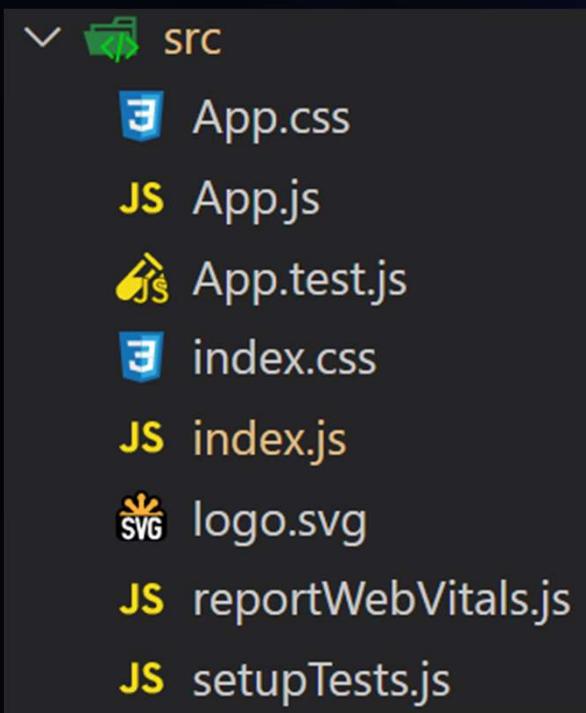


index.html

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

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

Src



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

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

JS App.js

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

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

App.js

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

index.js

קובץ הלוגיקה המרכזי של האפליקציה

- "יבוא מודולים:

- מופע מהספרייה React לkomponent
- קובץ העיצוב של הקומponent
- קומponent App
- קובץ בדיקות האופטימיזציה לkomponent

- יצרת קבוע בשם root שערך שווה ערך להפעלה מטודת ReactDOM.createRoot האלמנט HTML ב - DOM עם id=root (האלמנט זהה נמצא בתוך הקובץ index.html בתיקייה public)

- הפעלה מטודת root.render אשר תזריך לתוך האלמנט שתפסנו root את הקומוננטות (ידובר בהרחבה בהמשך)

- עדיפת האפליקציה בקומוננט של React שיכפה strictMode על הקוד שיוצג בתוכו (ידובר על כך בהרחבה בהמשך המציג)

- הצבת קומוננט App כר שתוצג לגולש 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    
10      <App />
11    
12  );
13
14  reportWebVitals();
```



React Server

בעזרת הפקודה בטרמינל `npm start` נפעיל את השירות הזמני של ריאקט וונכל לראות את התצוגה הראשונית של האפליקציה



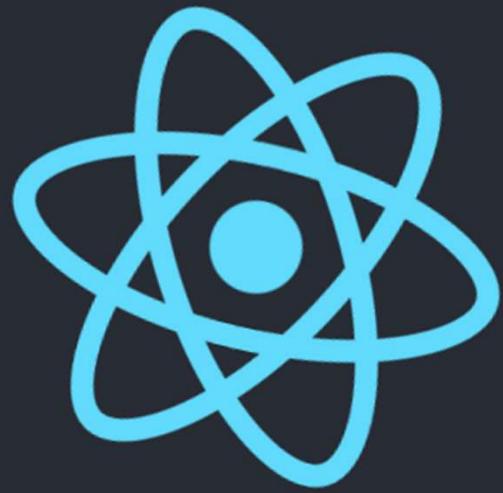
npm start

נפעיל את השרת הזמן של React
שגם יאפשר לשינויים בקוד ויתן לנו
תצוגת אמת של הקוד שלנו

```
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  
█
```

- נכנס לתיקייה הרלוונטייה בה נמצא פרויקט ה – React
- נקליד את הפקודה npm start
- ה – CLI יודיע לנו ובמידה ולא תהיה שגיאה בתהיליך compile נראה את הכתוב הבא
- ה – CLI יעדכן אותו כי אנו מאזינים לפורט הדיפולטיבי של React שהוא פורט 3000
- ה – CLI יעדכן אותו כי אנחנו בסביבת עבודה של development

! כמו כן – נפתח את הדף בכתובת
ובפורט הרשמי



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 שיעזר לנו בשלבי הפיתוח ב - React



React Developer Tools

Featured

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

<https://reactjs.org/blog/2015/09/02/noitallatsni#lmth.sloot-repoled-tcaer-wen/>



Components

לשוניית זאת מראה לנו נתונים נוספים על קומפוננט נבחרת

- בחלק זה של הדף נראה איפה עומדת הקומפוננט בהיררכיה של האפליקציה
- החלק השני מדבר על הקומפוננט עצמה ובו ניתן לראות נתונים כמו:
 - props – נתונים שהועברו לקומפוננט
 - state – משתנים שReact תגיב לשינויים בערכיהם
 - rendered by – הפקציה שאחראית על טעינה וטעינה מחדש ומחדש של הקומפוננט
 - source – קובץ המקור

! בגלל שהוא לא מנהלים state בקומפוננט זה
אני לא רואה את הנתונים הללו



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

[Learn React](#)

The screenshot shows the React DevTools Components tab. At the top, there's a search bar with placeholder text "Search (text or /regex/)" and a gear icon for settings. Below the search bar, the component tree starts with an "App" component. The first child of "App" is also an "App" component, which has a "props" field containing "new entry: ''". The second child of "App" is labeled "rendered by" and shows "createRoot()" and "react-dom@18.2.0". The final child of "App" is a "source" field pointing to "src/index.js:10". A blue bracket on the right side of the tree highlights the "props", "rendered by", and "source" fields. At the bottom left of the DevTools interface, there's a message "Edit `src/App.js` and save to reload." and a link "[Learn React](#)".

Profiler

The screenshot shows the React DevTools Profiler interface. At the top, there are two tabs: "Components" and "Profiler". The "Profiler" tab is active, showing a message: "No profiling data has been recorded. Click the record button ⚡ to start recording." Below this, another message says "Click [here](#) to learn more about profiling." In the main area, there's a component tree for a "Business Cards App". One component, "ToastCo...", is highlighted with a yellow border. A tooltip for this component displays the following data:

Rendered at:
3.2s for 14.2ms
3.9s for 10.6ms
4.6s for 13.2ms
5s for 32.1ms
7.2s for 2.5ms
8.8s for 11ms

Below the component tree, there are tabs for "Components", "Profiler", "Console", "Elements", and "Recorder". The "Profiler" tab is selected. The "Components" tab shows a list of components: BrowserRouter, Router, Navigation.Provider, Location.Provider, and App. The "Console" tab shows a log entry: "21 / 21". The "Elements" tab shows a list of elements: Header, NavBar, Navbar (ForwardRef), NavbarContext.Provider, Context.Provider, Container (ForwardRef), NavbarCollapse (ForwardRef), Anonymous (ForwardRef), Anonymous (ForwardRef), Transition, Conte..., and Nav (F...). The "Recorder" tab shows a histogram titled "HomePage".

בלשונית זאת נוכל לעשות בדיקות אופטימיזציה

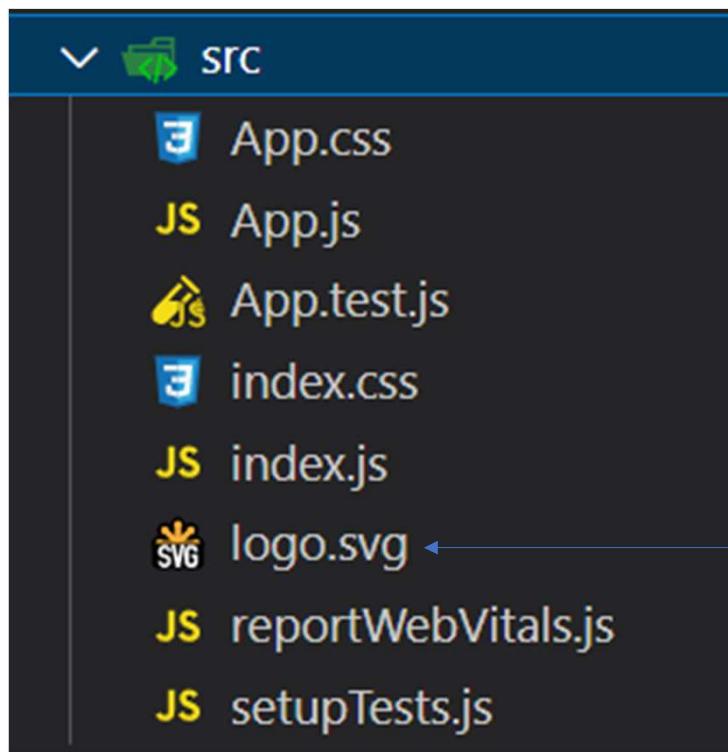
- לחיצה על כפתור `record` תחיל להאזין לאירועים שקרוים ב – DOM
- לחיצה נוספת תעוצר את ההקלטה ובמידה ויהיו נתונים להציג (כמו משך הזמן שלקח לכל אירוע לפועל) החלקים הללו יוצגו לנו.

- לחיצה על אחד מהאירועים תיתן לנו **פרטים עליון**

תחילת עבודה

ניקוי ראשוני של האפליקציה מתחנות וערכיהם דיפולטיים של
create-react-app





src

- מחיקת קובץ הלוגו של ריאקט

```
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;
```

App.js

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

index.css

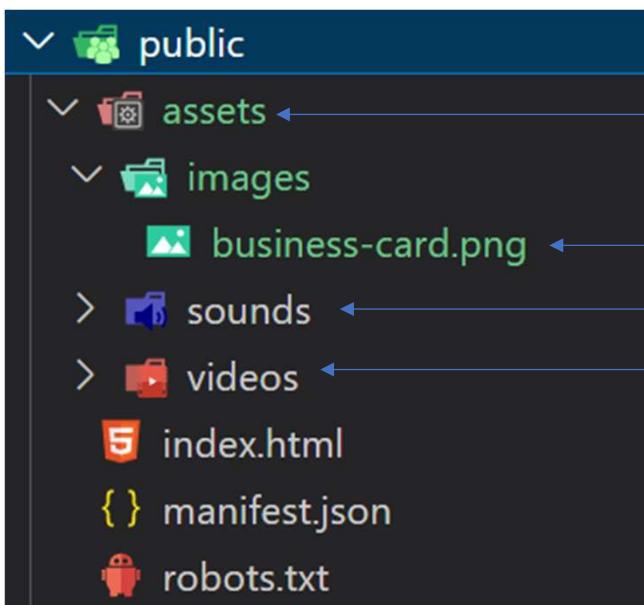
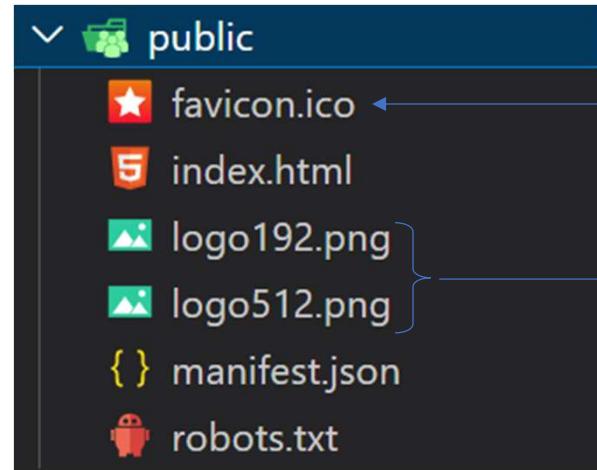
```
index.css X  
src > index.css > body  
1 body {  
2   margin: 0;  
3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',  
4   |   'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',  
5   |   sans-serif;  
6   -webkit-font-smoothing: antialiased;  
7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

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

- מחיקת תוכן הקובץ
- ייצירה של מחלקות עיצוביות משלנו
- מחיקת התוכן של הקובץ

App.css

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



public

- מחיקת קובץ favicon.ico של ריאקט
- מחיקת הלוגואים של react
- הוספה תיקייה בשם assets ובתוכה שלוש תיקיות:
 - Images •
 - נריד לאתר pixabay איקון מתאים לאפליקציה שלנו

Images •

• נריד לאתר pixabay איקון מתאים
לאפליקציה שלנו

sounds •

videos •

index.html

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

```
5 index.html M X

public > 5 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>
```

{ } 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 המקוצר של האפליקציה
- קביעת השם המלא של האפליקציה
- שינוי מיקום התמונות בשביל ה - icons

משימת app



Business-cards-app

- הורד את ה – CLI של create-react-app באופן גלובלי
- פתח פרויקט חדש בעזרת create-react-app בשם client
- הכן את הפרויקט לעבודה על ידי ניקוי הקבצים והתיקיות הלא 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 לתහיל המרת הקוד באמצעות
js.js ניתן לראות באתר שלהם
חתת הלשונית <https://babeljs.io/>
בתפריט הניווט של Try it out

- אריג המשחקים זהה בינוי משלווה חלקים:

- מסך ימני – מציג את הקוד לאחר תהיל הקומפליציה
- מסך אמצעי – משמש לכתיבה קוד javascript דקלרטיבי ועדרני
- תפירט צידי – ובו אפשרותויות שונות לתצוגת הקוד לאחר קומפליציה

The screenshot shows the Babel sandbox interface. On the left, there's a sidebar with settings like 'Evaluate', 'Line Wrap' (which is checked), 'Prettify', 'File Size', and 'Time Travel'. Below that are sections for 'Source Type' (set to 'Module') and 'TARGETS' (set to 'defaults, not ie 11, not ie_mob 11'). The main area has two panes: an input pane on the left containing the code `1 <div> hallo </div>` and an output pane on the right containing the transpiled code `1 "use strict"; 2 3 /*#__PURE__*/React.createElement(React.Fragment, null, "hallo");`. There are also tabs for 'Docs', 'Setup', 'Try it out' (which is active), 'Videos', and 'Blog' at the top.

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

! **לינק להרצאה המסביר איך js.js**
עובדת מאחורי הקלעים

Component

יחידת קוד עצמאית ואחת מאבני היסוד של ספרייה





Definition

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

Components structure



TEMPLATE
HTML

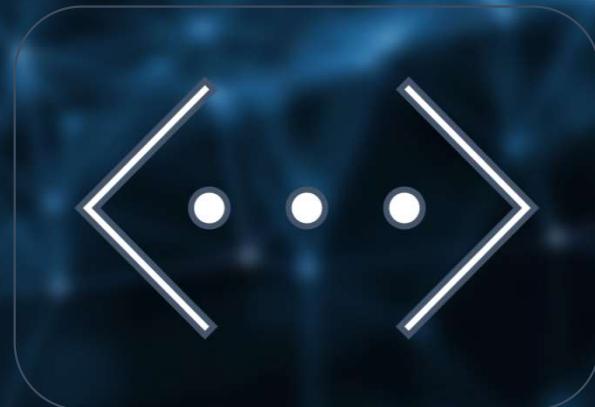


LOGIC
JAVASCRIPT



STYLES
CSS

Components Types

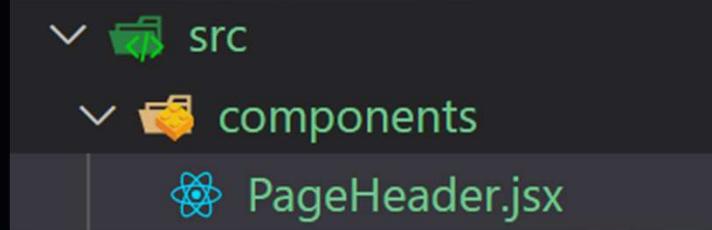


Function



Class

הכנות תשתית



- בתוכה תיקייה `src` נוצר תיקייה חדשה בשם **components**
- בתוכה נוצר קובץ בשם **PageHeader.jsx**.

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



Template

ויצור קומפוננט מסווג פונקציה שתחזיר
אלמנט של HTML אותו נציג לגולש



PageHeader

PageHeader.jsx

- נוצר קבוע בשם **PageHeader** שערך יהיה שווה לפונקציה אנונימית

- הפונקציה תחזיר אלמנט של **HTML** מסוג **H1** עם הכתיבה בתוכו

- לבסוף ניצא את הפונקציה **export default** מהמודול באמצעות **App.js**

App.js

- ניבא את הקומponent שיצרנו
- נציב אותה בתוך החלק של ה **HTML** אותה הפונקציה של הקומponent **App** ממחזירה.

! במצגת זאת נתמקד ב**קומponentות מסוג functional components** של **פונקציה** העטוף את כל האלמנטים שהפונקציה מחזירה באלמנט או של **React** או של **HTML**

PageHeader.jsx

```
src > components > PageHeader.jsx > ...
```

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

App.js

```
src > App.js > ...
```

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

התוצאה בדף

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

pageHeader works!



Compilation Error

במידה ותהיה שגיאה בקוד Babel תתריע
לי על כך במספר מקומות



איתור שגיאות

```
src
  └── components
    ├── PageHeader.jsx
    ├── App.css
    └── App.js
```

PageHeader.jsx 2, U

```
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;
```

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

בטרמינל של vscode

• בלשונית TERMINAL

- תופיע השגיאה Failed to compile
- פירוט השגיאה
- באיזה נתיב היא נמצאת

• בלשונית PROBLEMS

- יופיע באופן מכוון מקום השגיאה
- מהות השגיאה
- סוג השגיאה

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Local: http://localhost:3000

Failed to compile.

```
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 | }
```

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
cent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fr
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Filter (e.g. text, **/*.ts, !**/node_modules/**)

PageHeader.jsx src\components

2

```
✖ 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

• במסך התצוגה הראשי יופיעו פרטי
השגיאה

! במסך התצוגה הראשי ניתן לחוץ על
הסימן X ולחרור למסך האפליקציה אך
ומלץ לתקן את השגיאה בקוד במקום

```
Compiled with problems: X

ERROR in ./src/components/PageHeader.jsx

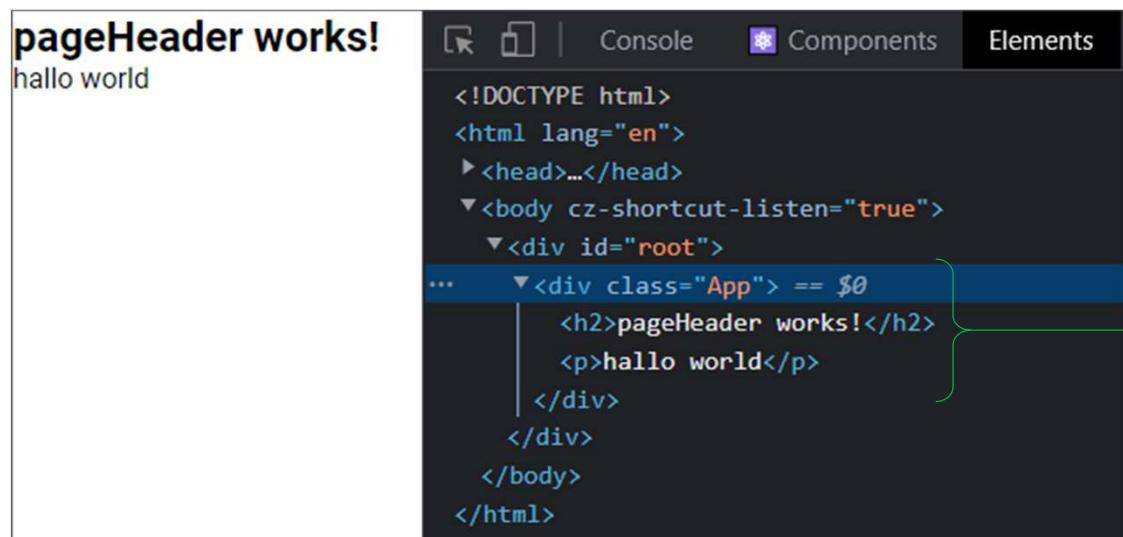
Module build failed (from ./node_modules/babel-loader/lib/index.js):
SyntaxError: C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\src\components\PageHeader.jsx: Adjacent JSX elements must be wrapped in an enclosing tag. Did you want a JSX fragment <>...</>? (4:2)

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

at instantiate (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules\@babel\parser\lib\index.js:67:32)
  at constructor (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules\@babel\parser\lib\index.js:364:12)
    at FlowParserMixin.raise (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules\@babel\parser\lib\index.js:7210:18)
      at FlowParserMixin.jsxParseElementAt (C:\Users\DELL\Desktop\HackerU\lecturer-work\Lessons\REACT\REACT-HOOKS\bcard-app\client\node_modules\@babel\parser\lib\index.js:7220:17)
        at FlowParserMixin.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 U X  
src > components > PageHeader.jsx > PageHeader  
1  const PageHeader = () => {  
2    return (  
3      <>   
4        <h2>pageHeader works!</h2>  
5        <p>hallo world</p>  
6      </>  
7    );  
8  };  
9  
10 export default PageHeader;
```



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

- אם אני לא מעוניין לעוטף את שני האלמנטים באלמנט עיצובי של HTML כמו div ריאקט pseudo element משלה שנקרא React.Fragment שבעזרתו אוכל לעוטף את האלמנטים מבלי האלמנט הזה לא יראה ב – DOM

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

הדרך המקוצרת של כתיבת האלמנט
היא <></>



Logic

כמו בכל פונקציה גם בקומפוננט ניתן ליצור לוגיקה מלבד החזרת אלמנט HTML ויצאת שתשפיע על האלמנט המוחדר



הוספה לוגיקת

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

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

PageHeader.jsx

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;
```

pageHeader works!

hallo world

Console

Console

>

1

⋮

x

Filter

Custom levels ▾ 1 Issue: 1

30 PageHeader.jsx:3

30 VM1241:236



String interpolation

יצירת אזור JAVASCRIPT באזור המוגדר
HTML בקומפוננט



String interpolation example

פתיחה אżור JAVASCRIPT באżor המיעוד ל-HTML מתבצעת על ידי פתיחה וסירה של סוגרים מסולסים

בדוגמה שללן:

- בתוך ה- scope של הקומפונט יוצרתי קבוע בשם text והשוויתי את הערך שלו למחרוזת תווים
- באżor המיעוד לשפת HTML פתחתי אżור של JAVASCRIPT בעזרת פתיחה סוגרים מסולסים ובתוכם הצבתי את שם הקבוע שיצרתי
- בעזרה string interpolation נוסף פתיחה אżור JAVASCRIPT גם בתוך אלמנט נוסף של HTML והפעם ביצעת חישוב כפי ששפת JAVASCRIPT יודעת לעשות
- התוצאה בדף:
 - כפי שניתן לראות הטקסט הוצב במקום שהגדרתי לו
 - החישוב בוצע במקום שהגדרתי לו

PageHeader.jsx

```
src > components > PageHeader.jsx > ...
1  const PageHeader = () => {
2    const text = "Hello 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!

Hello world

30



Styles

הוסף עיצוב לקומפוננט



Styles Types



INLINE



IMPORT STYLES FROM
MODULE



EXTERNAL LIBRARIES



Inline style

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



Inline style

בדוגמה שלהן:

- ניצור קבוע בשם `headLineStyle` ומשווה את הערך שלו לאובייקט JAVASCRIPT אשר המפתחות שלו הם המאפיינים העיצובים כמו בכל אובייקט JAVASCRIPT יופיעו לאחר הנקודותים
- ניצור בתגית HTML הפתוחת מאפיין `style` ומשווה את הערך שלו לאזר JAVASCRIPT אליו עבריר את שם הקבוע שיצרנו
- בדוגמה השנייה עבריר ישרות אובייקט קונפיגורציות לתוך המאפיין `style` בתगית הפתוחת של האלמנט פסקה של HTML
- התוצאה בדפסן

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

```
❖ PageHeader.jsx ●  
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



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 שלහן:

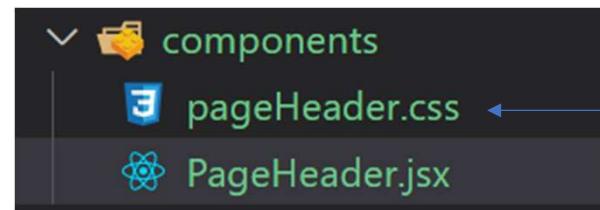
- ניצור קובץ חדש בשם pageHeader.css

- ניצור מחלקת עיצובית של css בקובץ שיצרנו

- ניבא את המודול לתוך הקובץ של הקומפוננט

- השתמש במחלקה העיצובית שיצרנו

יש לשים לב ל syntax של המאפיין className בריakkט 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 השלישית לעיצוב אלמנטים של HTML
בקומponentות של ריאקט היא באמצעות הBAT
ספריות עיצוב חיצונית ושימוש במחלקות
העיצוב שליהן



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 אב לkomponent בן





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 אזור של JAVASCRIPT בתוך החלק המועד ל – HTML בקומפוננט ונציב בתוכו את הערך של המפתח שחילצנו מתוך אובייקט הprops

ChildComp.jsx X

```
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;
```

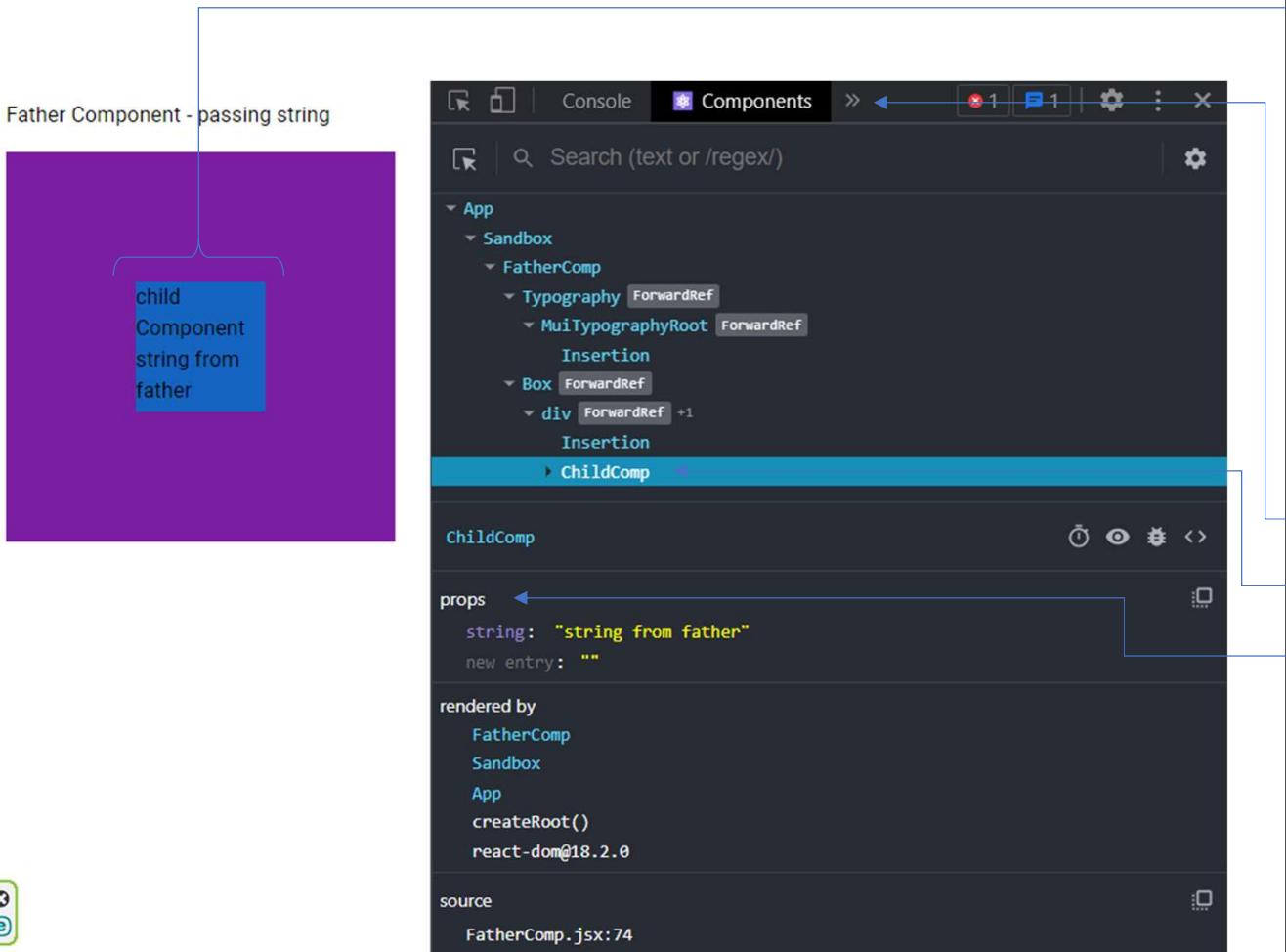
```
❖ FatherComp.jsx U X
client > src > sandbox > props > ❖ FatherComp.jsx > ...
1 import { Box, Typography } from "@mui/material";
2 import React from "react";
3 import ChildComp from "./ChildComp";
4
5 const FatherComp = () => {
6   const string = "string from father"; ←
7
8   return (
9     <>
10    <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
- נציג את קומפוננט הבן **ChildComp** בתוך החלק המועדף – HTML בקומפוננט האב וונעשה השמה למפתח **string** בטור אובייקט ה – **props** ונקבע את ערכו לקבוע **string** שיצרנו.

התווצה בדף

- ניתן את הטקסט שהעבירנו מkomponent האב מוצג בתוך komponent הבן
- בנוסף בגלל שהורדנו את התוסף react dev tools אנו יכולים לגשת לשונית components
- לחוץ על komponent שמשמעותו לנו
- ולקבל בין היתר את המפתחות והערכים שמועברים באובייקט props





Passing Object

העברת אובייקט מקומפוננט אב לקומפוננט בן
בקומפוננט מסווג פונקציה וחילוץ המפתחות
והערכים ממנה



Child Component

- ב글 שקומפוננט מסווג פונקציה מתנהגת כמו כל פונקציה ב – JAVASCRIPTani יכול לחלץ מאובייקט props את המפתחות הרלוונטיים ישיר בתוור הפרמטר של הפונקציה
- נחלץ את מפתחות first, last מתוך המפתח name שבאובייקט ה - props
- נפתח איזור של JAVASCRIPT בתווך החלק המיועד ל – HTML בקומפוננט 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 };
```

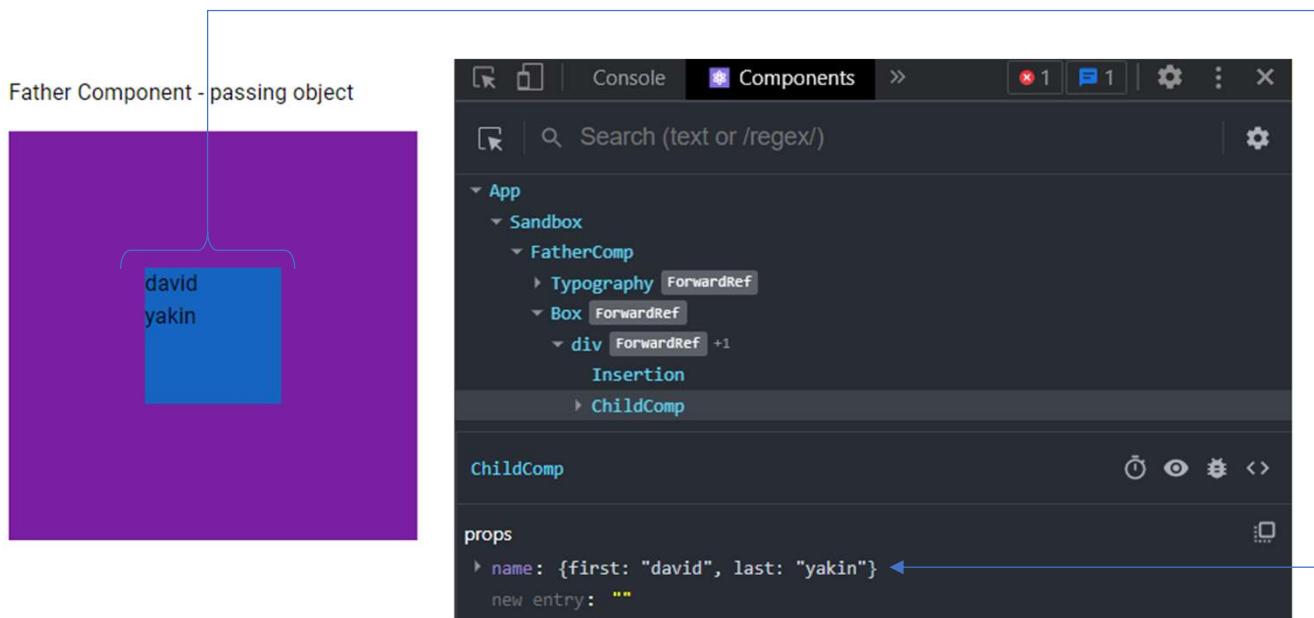
```
31 const FatherComp = () => {
32   const name = { first: "david", last: "yakin" }; ←
33
34   return (
35     <>
36       <Typography variant="body1" m={2}>
37         {" "}
38         Father Component - passing object
39       </Typography>
40       <Box
41         sx={{
42           m: 2,
43           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 };
```

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.

התווצהה בדף

- ניתן את הכתיבה שהעבכנו מkomponentה האב לkomponentה הבן בתוך האובייקט מוצג בkomponentה הבן
- וכי אובייקט ה – props מכיל עצט מפתח בשם name שהערך שלו זה האובייקט שיצרנו





Sending two keys

הדרך להעביר יותר מפתח אחד לאובייקט
הפרופו



Child Component

- בגלל שקומפוננט מסווג פונקציה מתנהגת כמו כל פונקציה ב – JAVASCRIPT נוכל להלץ מאובייקט props מספר מפתחות first, last

- נפתח אזור של JAVASCRIPT בתוך החלק המועד ל – HTML בקומפוננט ונציב בתוכו את הערךם של המפתחות שהילצנו מתחור 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 ];
```

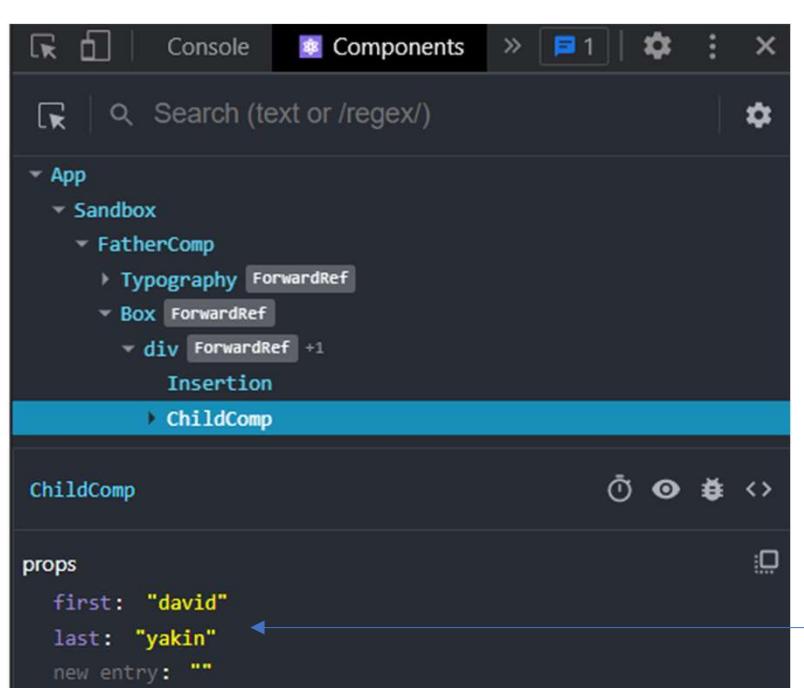
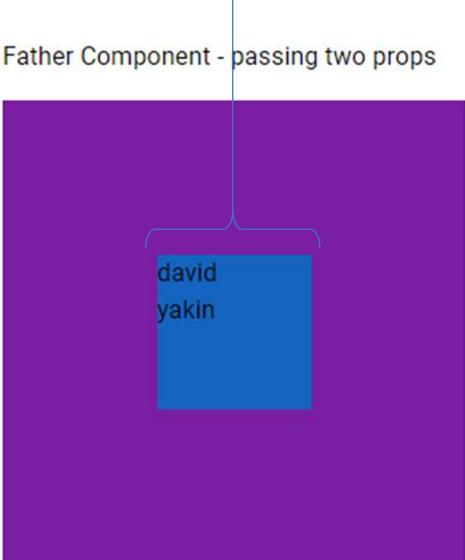
Father Component

- נוצר קבוע בשם name שערך יהיה אובייקט עם מפתחות וערכים
- הפעם נעביר כל מפתח מהאובייקט שיצרנו לתוך מפתח משלה באובייקט הפרופס

התווצהה בדף

- ניתן את הכתיבה שהעבכנו מkomponent האב לkomponent הבן בתוך האובייקט מוצג בkomponent הבן
- וכי אובייקט ה – props מכיל עצט מפתח בשם name שהערך שלו זה האובייקט שיצרנו

Father Component - passing two props



Props משימת

```
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

- בmarsh למשימת Card במצגת UI-Android צור בתיקייה בנתיב `src/cards/components/card` שלושה קבצים נוספים:
 - CardHead – קומponent זה יכול תמונה שאת הערכים שלו (url, alt) הקומponent קיבל אובייקט הפרופס
 - CardBody – קומponent זה יכול כותרת ראשית ומשנית לכרטיס, חוץ ושלוש שורות טקסט כפי שמופיע בדוגמה שבשקף הבא. את הערכים לשדות הטקסט עליו לקבל מפתח בשם `card` מתוך אובייקט ה- `props`
 - CardActionBar – אוצר זה בכרטיס יוכל אייקון של לב
- בקובץ Card צור קבוע בשם `card` שיכיל את המפתחות והערכים המופיעים בדוגמה משמאל
- הציב את שלושת הקומponentות שיצרת בתוך הקומponent `.Card`.
- העבר לקומponentות הבנים את המידע הדרוש להם באמצעות אובייקט הפרופס על מנת שיוכלו להציגו בגלוש

Props משימת חלק ב'



forth

subtitle

Phone: 050-0000000

Address: Shinkin 3 tel-aviv

Card Number: 4000000



Loops

הדרך לבצע לולאות ב React



Map Loop

בחרה להשתמש בMETHOD map
בשביל לבצע לולאות על מערכים באזור
המיועד רק ל-HTML
בדוגמה של להלן:

```
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;
```

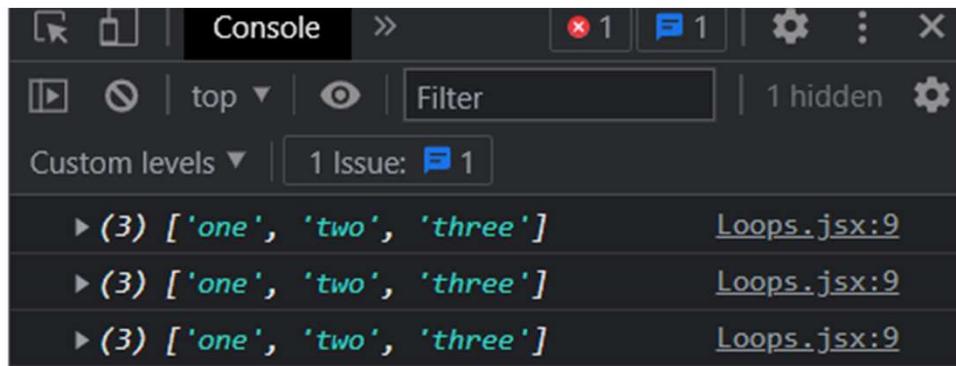
- ייצרנו קבוע בשם `arrayOfString` והשווינו את ערכו לערך של מחרוזות תווים
- בחלק המיועד לו-HTML
 - אנחנו מבצעים לולאה על הקבוע `arrayOfString` באמצעות METHOD `map` שמקבלת עד שלושה פרמטרים:
 - Item – האיבר במערך
 - Index – מספר האינדקס של האיבר במערך
 - array – המערך שעליו נערךת האיטרציה
 - בתוך הלולאה אני מדפיס את המערך
 - ומציג לגולש כתוב שמקורו במערך
- כל אלמנט שהוא מכפילים באיטרציה צריך לקבל את המאפיין `key` שצריך להיות ייחודי.

התווצהה בדף

בחרה להשתמש במתודת `map` בשביל לבצע לולאות על מערכים באזור המועד ו-HTML

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

item: one
item: two
item: three



משימת Map



Business-cards-app

- צור את הנתיב הבא:
`src/cards/components/Cards.jsx`
- Cards.jsx •
- צור מערך עם שלושה אובייקטים הלו צריכים להיות תואמים למפתחות של אובייקט הלקוח מתרגיל הקודם
- השתמש בMETHOD map כך שעל כל איבר במערך תציג גולש כרטיס בעזרת הקומponent `Card.js`.
- בדוק בדף כי אכן הלקוחות מוצגים גולש ! על הקומponent `Card` לקבל אובייקט ה – `props` כרטיס `card` במקום קבוע `card` שיצרנו בתרגיל הקודם.

Conditional Rendering

תצוגות מידע שונות כאשר יש מידע להציג וכאשר אין

! יש לעבור על החלק של Layout במצגת של WUI לפני 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 ונרצה 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(cards) הקומפוננט יעצור ויחזיר תוצאה לגולש שתכלולאזור טקסט עם מחרוזת תווים

Events

הדרך להאזין לאירועים ולהפעיל מטודות בעקבותיהם





JAVASCRIPT EVENTS

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

- **React** נתנת תמיכה לכל סוג האירועים ב-
JAVASCRIPT ובכל אחד מהם ניתן להפעיל
פונקציה או **מتدודה אחרת**



onClick event

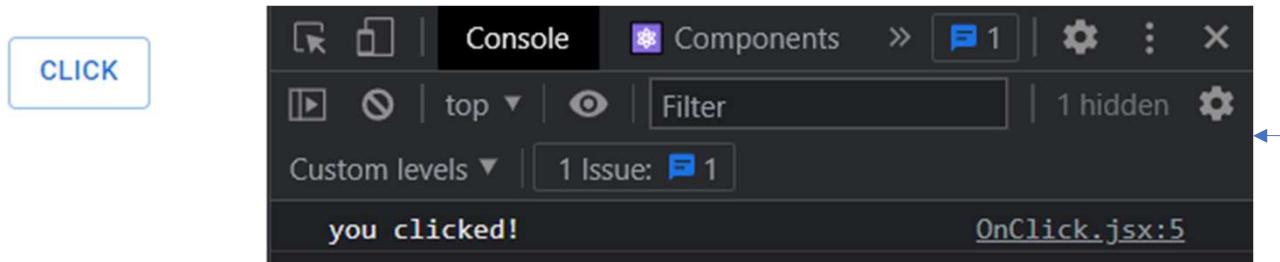
בדוגמה של הלן:

- יצרנו קומponent בשם **OnClick**
- יצרנו קבוע בשם **handleClick** שווה ערך **לפונקציה אוניברלית** כתופעל תדף בקונסול מחרוזת תווים
- יצרנו כפטור שיאזין לאיירוץ **onClick** ויפעל את הפקציה **handleClick** ברגע שהאיירוץ יקרה.

התוצאה בדף:

- כשלחץ על הכפטור **CLICK** תודפס בקונסול מחרוזת התווים מתוך הפקציה **handleClick**

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





Function invocation with parameters

הפעלת פונקציה עם פרמטרים



Function with parameters

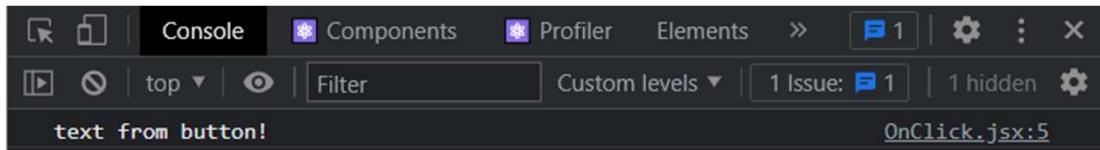
בדוגמה של להלן:

- הקבוע handleClick שווה לפונקציה אוניברלית שמקבלת פרמטר text מסוג מחרוזת תווים ומדפיסה אותו בקונסול
- אני מעביר בערך של האירוע onClick פונקציה call back אוניברלית שכשהאירוע יקרה היא תופעל ובתוך הפונקציה אני מפעיל את הפונקציה handleClick עם הערך שאני מעוניין שיודפס בקונסול
- התוצאה בדף:
 - כשנלחץ על הכפתור תודפס בקונסול מחרוזת התווים מתוך הפונקציה handleClick

! אםעביר לאירוע ישירות את הפעלת המטודה handleClick עם הפרמטר ללא הפונקציה האוניברלית היא תופעל עם יצירת הקומפוננט ולא מתן שהאירוע יקרה

OnClick.jsx

```
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 האירוע והעברתו לפונקציית ה – call
back



Catching event

בדוגמה שלהן:

- הקבוע handleClick שווה לפונקציה אנונימית שמקבלת פרמטר e מסוג אירוע ומדפיסו אותו בקונסול את האלמנט שתפס את האירוע
- אני מעביר בערך של האירוע onClick פונקציה call back אנונימית שמקבלת את האירוע ומפעילה את הפונקציה handleClick
- התוצאה בדף:**
 - כשנלחץ על הceptor יודפס בקונסול האובייקט שהאזין לאירוע והפעיל את הפונקציה handleClick

The screenshot shows a code editor window for a file named 'OnClick.jsx'. The code defines a functional component 'OnClick' that logs the target of a click event to the console. It uses the Material-UI 'Button' component with an 'outlined' variant and a size of 'medium'. A 'CLICK' button is visible in the browser's developer tools' element inspector, which is currently selected. The browser's developer tools show the rendered HTML for the button, including its class names and attributes.

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

CLICK

OnClick.jsx: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-19skcmy-MuiButtonBase-root-MuiButton-root" tabindex="0" type="button">...</button> flex
```

הדרך השני להעביר

אידוע לפונקציה

בדוגמה שללן:

- הפעם אני כביר ל��ול לא מעביר לפונקציה handleClick פרמטר אלא מעביר אותה כפונקציית javascript callback. אולם זו לא פונקציה את האירוע ואוכל להדפיסו בקונסול

התוצאה בדף:

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

>...</button> flex'. A blue arrow points from the 'CLICK' button in the browser to the 'onClick' prop in the code, and another blue arrow points from the browser's output to the 'handleClick' function definition in the code."/>

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

CLICK

Console Components Profiler Elements > 1 Issue: 1 hidden

onClick.jsx:5 <button class="MuiButtonBase-root MuiButton-root MuiButton-outlined Primary MuiButton-sizeMedium MuiButton-outlinedSizeMedium MuiButton-root MuiButton-outlined MuiButton-outlinedPrimary MuiButton-sizeMedium MuiButton-outlinedSizeMedium css-19skcmw-MuiButtonBase-root-MuiButton-root" tabindex="0" type="button">...</button> flex



Raising Events

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



הפעלה מטודה בקומפוננט

האב בקומפוננט הבן

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

בקומפוננט האב

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

בקומפוננט הבן

- אני מחלץ את המפתח handleClick מאובייקט הförופס
- אני מORIZן לאיורע onClick שתפעיל את מטודת handleClick

```

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

```

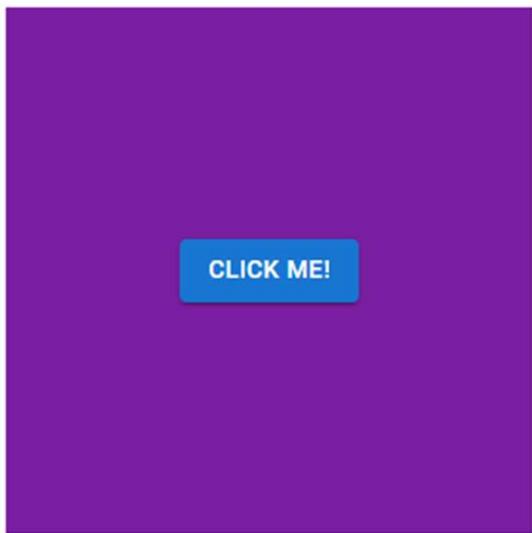
```

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

```

התוצאה בדף

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



```
Console  »  1 | 1 hidden  ⚙️
▶  top  ⚡  Filter
Custom levels  ▾  1 Issue:  ⚡  1
you clicked!  FatherComp.jsx:84
>
```

משימת Events



Business-cards-app

- צור בקומפוננט axj.Cards קבוע בשם handleCardDelete שיהיה שווה ערך לפונקציה אנוינימית שתתקבל בפרמטר `id` מספר ותדפיס בקונסול את מחרוזת התווים `no`. "you deleted card no." והוסף את המספר שמופיע ב מפתחה `_id`
- העבר באובייקט הпроפס את מטודות onDelete onLike מקומפוננט האב axj.Cards דרך קומפוננט הבן axj.Card.js ועד לקומפוננט שמתעסקת עם Action Buttons
- בלחיצה על אייקון הלב הדפס את מחרוזת התווים "you liked card no:" והוסף את הערך שמופיע ב מפתחה `_id`
- בלחיצה על אייקון פח האשפה הדפס את מחרוזת התווים "you liked card no:" והוסף את הערך שמופיע ב מפתחה `_id`
- בלחיצה על אייקון עריכת הלקוח הדפס את מחרוזת התווים "edit card no." והוסף את הערך שמופיע ב מפתחה `_id`

propTypes

Strong Typing in React

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





Q Definition

Runtime type checking for React props
and similar objects.



Installation

`npm i prop-types`



Types

הערות	PropTypes	ערך לבדיקה	או'
	.string	"string"	.1
	.number	5	.2
	.bool	true/false	.3
	.object	{ }	.4
מגדיר את סוג ערכי המפתחות באובייקט	.objectOf()		.5
	.array	[]	.6
מגדיר את סוג האיברים במערך	.arrayOf()		.7
	.func	()=>{ }	.6
אחד מהסוגים המפורטים בתוך המערך של הפונקציה	.oneOfType([])		
אחד מהערכים שמופיעים במערך בלבד	.oneOf(['News', 'Photos'])		.13
instance of a class	.instanceOf(new Class)		
ערך דיפולטיבי	.defaultProp		
יציר סימן חדש	.symbol	Symbol()	.7
	.bigint		.8
	.node		.9
	.element		.10
	.elementType		.11
	.shape({ })		.16
מנודא 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 בתחום קומפוננט האב אך לא נעביר לה את המפתחות שהיא זקוקה להם באובייקט הפרווף

בקומפוננט הבן

- ניצור מופע של מחלקת PropTypes מתווך הספרייה שייבאנו "prop-types"

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

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

! בגלל שפונקציה מאחוריה הקלעים היא אובייקט
אני יכול לעשות השמה למפתחות שלא

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

```
PropTypeComponent.jsx ✘ ×  
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;
```

התוצאה בדף

בגל שלא העברתי מקומפוננט האב לקומפוננט הבן את המפתח שעשית עלי בדיקה באמצעות `propTypes` נזרקת לי על כך שגיאה

PropTypeComponent

The screenshot shows a browser's developer tools console. The title bar says "PropTypeComponent". The main area of the console displays a warning message:

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



Main Types

סוגי הערכים המרכזיים



Main Types

בקומפוננט האב

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

בקומפוננט הבן

- הקומפוננט PropTypeComponent מקבל בפרמטר את האובייקט props
- נחלץ את הערכים מתוך האובייקט props
- נעשה השמה למפתח propTypes בתוך PropTypeComponent שיצרנו ומשווה את הערך שלו לאובייקט שיבדק את המפתחות שאין מעוניין שהיו 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 יכלול אובייקטים ומערכות
באמצעות PropTypes



arrayOf & objectOf

בקומפוננט האב

- נציב את קומפוננט הבן בתחום קומפוננט האב ונוביר לה את המפתחות שהיא זקוקה להם באובייקט הפרווף

בקומפוננט הבן

- הקומפוננט מקבלת בפרמטר את האובייקט props
- נחלץ את הערכים מתוך האובייקט props
- נעשה השמה למפתח propTypes בתחום קומפוננט PropTypeComponent שיצרנו ונשווה את הערך שלו לאובייקט שיבדוק את המפתחות שאנו 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 סוגי ערכים או לקבוע את הערכים באופן ליטרלי



oneOfType & oneOf

בקומפוננט האב

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

בקומפוננט הבן

- הקומפוננט מקבל בפרמטר את האובייקט props
- נחלץ את הערכים מתוך האובייקט props
- נעשה השמה למפתח propTypes בתוך PropTypeComponent לקומפוננט שיצרנו ומשווה את הערך שלו לאובייקט שיבדוק את המפתחות שאין מעוניין שהיו באובייקט props ואת הערכים שלהם

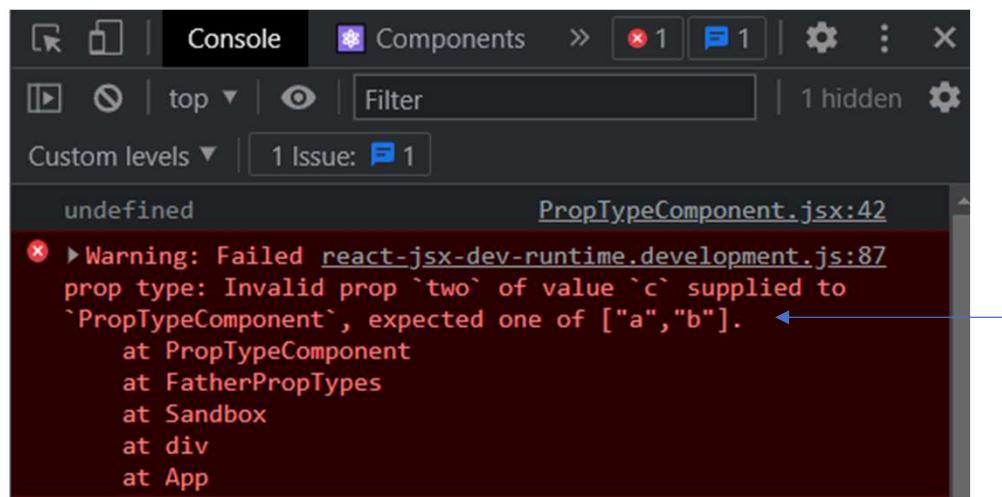
```
40 const FatherPropTypes = () => {
41   return (
42     <>
43       <PropTypeComponent one="string" />
44       <PropTypeComponent one={2} />
45       <PropTypeComponent two="a" />
46       <PropTypeComponent two="c" />
47     </>
48   );
49 };
```

```
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 };
```

התוצאה בדף

אנו מקבלים שגיאה בגל שניסינו להעביר
בפתחoso אובייקט הפרופס מחרוזת
תווים שאינה אחת ממחוזות התווים
שהגדכנו

```
PropTypeComponent  
PropTypeComponent  
PropTypeComponent  
PropTypeComponent
```





Exact & isRequired

יש ביכולתנו לבדוק אם באובייקט הפרופס יש
בדיוק את הערכים שאנו מבקשים או לחליפין
לחיבב העברת מפתח ספציפי



Exact

בקומפוננט האב

- נעביר לה את המפתחות שהוא **זקוקה** להם באובייקט הprops אולם נשים מפתח אחד יותר מיד' בתוך האובייקט **שאנו מעבירים**

בקומפוננט הבן

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

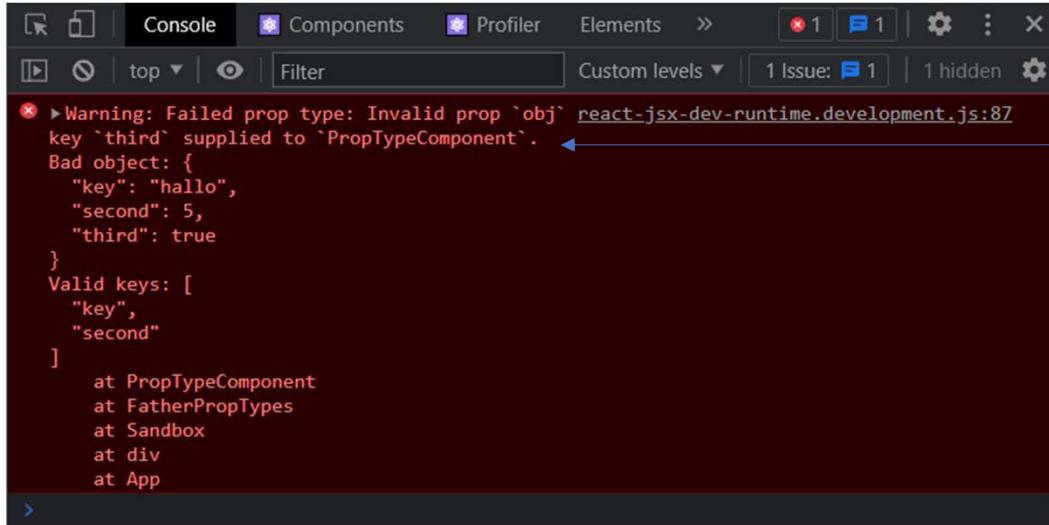
```
52 const FatherPropTypes = () => {
53   const obj = { key: "hallo", second: 5, third: true };
54   return <PropTypeComponent obj={obj} />;
55 };

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 };
```

התוצאה בדף

בגל שהערכנו מפתח שלישי של PropTypes
זורקת לנו הודעה שגיאה על כך



The screenshot shows the DevTools console with the following message:

```
PropTypeComponent
Console Components Profiler Elements > 1 Issue: 1 | 1 hidden
✖ Warning: Failed prop type: Invalid prop `obj` supplied to `PropTypeComponent`. react-jsx-dev-runtime.development.js:87
Bad object: {
  "key": "hallo",
  "second": 5,
  "third": true
}
Valid keys: [
  "key",
  "second"
]
at PropTypeComponent
at FatherPropTypes
at Sandbox
at div
at App
>
```

A blue arrow points from the text "זהו מבחן שפונקציית PropTypes מודפסת" to the word "PropTypes" in the warning message.

isRequired

בקומפוננט האב

- נציב את קומפוננט הבן בתוך קומפוננט האב אך לא נעביר לה את המפתח שהוא צפוקה לו באובייקט הפרופס

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

צירת interface של אובייקט



Shape

בעזרת `PropTypes.shape` אני יכול להגדיר `interface` של אובייקט בkomponentet האב

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

בkomponentet הבן

- נחלץ את המפתחות שאנו צריכים להם מאובייקט `PropTypes`
- ניצור קבוע בשם `imageType` שיהיה שווה ערך למפתח `shape` מתוך אובייקט `PropTypes` שיקבל אובייקט קונFIGורציות עם המפתחות והערכים שאנו מעוניינים שלו 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 בנים בתוך קומפוננט האב
 - לראשונה נעביר מפתח בשם name עם ערך לאובייקט הפרויקט
 - בשניה רק נציג את הקומפוננט מבלי להעביר לה מפתחות לאובייקט הפרויקט

בקומפוננט הבן

- הקומפוננט צריכה לקבל את המפתח name באובייקט הפרויקט ולהחזיר אותו
- נגיד רשם name המפתח יכול להיות כל סוג של ערך בעזרה עם אבל שהוא ובה להעביר בו ערך כלשהו
- השתמש בdefaultProps כדי לקבועערכים דיפולטיבים למפתחות באובייקט הפרויקט כך שאם לא יעבירו לנו ערך למפתח name נקבע שהערך הדיפולטיבי שלו יהיה "david"

התוצאה בדף

- בגלל שהערכנו את המפתח name לקומפוננט בן הראשון אנו מקבלים את הערך שהערכנו
 - בגלל שלא הערכנו את מפתח name בקומפוננט בן השני מוצג הערך הדיפולטיבי שקבענו !
- לא מומלץ להשתמש ב – `name` אלא להגדיר את סוג הערך המבוקש**

```

74 const FatherPropTypes = () => {
75   return (
76     <>
77       <PropTypeComponent name="shola" /> ←
78       <br />
79       <PropTypeComponent /> ←
80     </>
81   );
82 }
83
84
85
86
87 const PropTypeComponent = ({ name }) => { ←
88   return name;
89 };
90
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
וקומponentות ילדים



node & children

בקומפוננט האב

- עבור לפתח באובייקט הпроופס `node` מחרוזת תווים

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

בקומפוננט הבן

- הקומפוננט תחלץ מאובייקט הпроופס את מפתח `node` ומה שהעבכנו בין התגית הפתוחת לתגית הסגרת של קומפוננט יהיה בתוך מפתח `children` גם מילה שמורה בשם `children`

- אני מחזיר מהקומפוננט את שני המפתחות שהילצתי כך שיוצגו לגולש

- בעזרת `PropTypes` נודא ש:

- ב מפתח `node` הועבר לנו ערך שניינן להציגו לגולש
- שב מפתח `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

```

89  const FatherPropTypes = () => {
90    return (
91      <PropTypeComponent node="David">
92        <PropTypeComponent node="yakin" />
93      </PropTypeComponent>
94    );
95  };
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 };

```

PropTypes.element

בקומפוננט האב

- בין התגיות הפתוחות 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 מותן כתוב ב מפתח node

בקומפוננט הבן

- אני מחלץ את המפתחות node children
- מדפס בקונסול את children
- מחזיר מהקומפוננט לגולש children כי הערך של המפתח react על ידי צריך להיות אלמנט של PropTypes element של PropTypes

David yakin

The screenshot shows the Chrome DevTools Elements tab. At the top, there are tabs for Console, Components, Profiler, and Elements. The Elements tab is active. Below the tabs is a toolbar with icons for back, forward, search, and filter. The main area displays the structure of a React element. The element has the following properties:

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

The code editor at the bottom shows the file `PropTypeComponent.jsx:112`.

התוצאה בדף

- ניתן לראות שהאלמנט שהעברתי לקומפוננט ב- `children` הוא מסוג אובייקט
- שיש לו מפתח בשם `props` שהערך שלו הוא אובייקט עם המפתח `node` והערך שהעברתי לקומפוננט הבן שבתוך קומפוננט הבן

arrayOf(element)

בקומפוננט האב

- הפעם נعبر לkomponent הבן בפתח ה - children מספר אלמנטים

בקומפוננט הבן

- נחלץ את מפתח children

נדפסו אותו

- נחזיר אותו מהkomponent

באמצעות PropTypes נבדוק שacky העברו komponent 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 שהערכנו יותר מאלמנט אחד React הכניסה את האלמנטים לתוך מערך כscal אלמנט הוא איבר במערך

first child
second child

The screenshot shows the Chrome DevTools Elements tab. In the DOM tree, there is a node with the text "first child". Its "children" prop is expanded, showing an array of two elements. The first element at index 0 is a `div` with the text "first child". The second element at index 1 is a `p` with the text "second child". The `children` prop itself has a type of `Symbol(react.element)`. The `length` of the array is 2.

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

משימת חלק א' PropTypes



Business-cards-app

- צור את הנתיב `src/cards/models/types`
- צור שלושה קבצים בנתיב שיצרת:
 - `cardType.js` – פירוט על קובץ זה בעמוד הבא
 - `imageType.js` •
 - צור קבוע בשם `imageType` שיהיה שווה ערך לשיחזור ממטודת `PropTypes.shape` אליה תעביר את אובייקט הקונפיגורציות עם הערכים הבאים:
 - `url` – מסוג מחוץ תווים שדה חובה
 - `at1` – מסוג מחוץ תווים שדה חובה
- יצא את הקבוע שיצרת באמצעות `export default addressType.js` •
- צור קבוע בשם `addressType` שיהיה שווה ערך לשיחזור ממטודת `PropTypes.shape` אליה תעביר את אובייקט הקונפיגורציות 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` שייהה שווה ערך לערך שיחזור ממתודת `PropTypes.shape` אליה תעביר את אובייקט הkonfiguration עם הערךים הבאים:
 - `id` – מסוג מחוץ תווים שדה חובה
 - `title` – מסוג מחוץ תווים שדה חובה
 - `subtitle` - מסוג מחוץ תווים שדה חובה
 - `description` - מסוג מחוץ תווים שדה חובה
 - `address` – מסוג `addressType` שדה חובה
 - `image` – מסוג `imageType` שדה חובה
 - `bizNumber` – מסוג מספר שדה חובה
 - `phone` - מסוג מחוץ תווים שדה חובה
 - `likes` – מסוג מערך של מחוץ תווים שדה חובה
 - `web` - מסוג מחוץ תווים או `undefined` שדה חובה
 - `email` - מסוג מחוץ תווים שדה חובה
 - `user_id` - מסוג מחוץ תווים שדה חובה
 - `createdAt` - מסוג מחוץ תווים שדה חובה
- יצא את הקבוע שיצרת באמצעות `export default`

משימת PropTypes חלק ג'



Business-cards-app

- בקומפוננט card השימוש בPropTypes ובמודלים שיצרת בשקפים הקיימים על מנת לוודא כי כרטיס ביקור מסווג של cardType מועבר לקומפוננט Card
- וודא שכל הקומפוננטות שמרכיבות את קומפוננט Card מקבלות גם הן את הפרופס הדרושים להם כדי לעבוד כהלאה בעזרת המודלים שיצרת בשקפים הקיימים

Shared Components

יצירת קומפוננטות המשותפות למספר דפים ומספר פיצ'רים





PageHeader

יצירת קומפוננט של כותרות וכותרות משנה
לדף באתר



PageHeader.jsx

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;
```

ניצור את הנתיב :

src/component/PageHeader.jsx

- הkomponennt מקבל את המפתחות title subtitle באובייקט הפרופס
- ניצור אלמנט h1 עם עיצוב של h2
- ניצור אלמנט h2 עם עיצוב של h5
- ניצור אלמנט hr בעזרת הקומпонנט Divider
- נודא שהקומוננט מקבלת את המפתחות הדרושים לה באובייקט הפרופס באמצעות propTypes

```
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 = [ ... ];
8
9   return (
10     <Container sx={{ mt: 2 }}>
11       <PageHeader
12         title="Cards"
13         subtitle="On this page you can find all business cards from all
14           categories"
15       />
16
17       <Cards cards={cards} />
18     </Container>
19   );
20 };
21
22 export default CardsPage;
```

CardsPage.jsx

- ניצור את הנתיב:
src/cards/pages/CardsPage.jsx
- מעביר את הכרטיסים לkomponenet ה затה
- געטוֹף את תוכן הדף במייל של MUI
- נציב את הקומפוננט שיצרנו PageHeader בטור דף המיעוד לתצוגת הכרטיסים עם הכותרות המתאימות
- נציב את הקומפוננט Cards ומעביר אליו את המשתנה cards כמפתח באובייקט הפרויקט

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 המיועד לטקס המסביר על האפליקציה
- וראאים את התמונה של הלקוח

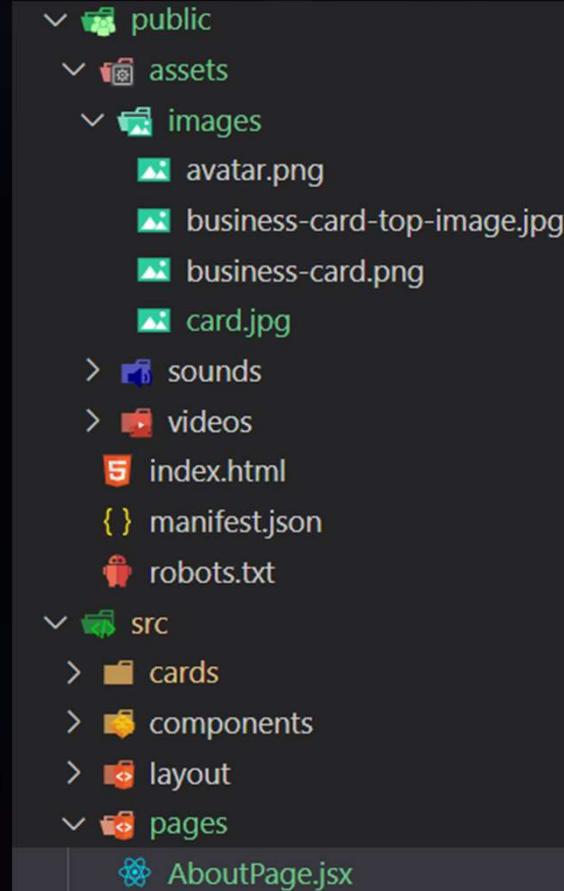
Static Folder

Using the Public Folder

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



הכנות לשתייה



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

AboutPage.jsx

יצור דף אודות שישתמש בתמונה שנשמר
בתיקיית public

- ייצור קומפוננט בשם AboutPage שת חריז |
- נשתמש בקומפוננט Container כדי לתת רווח לתוך בגדי מסך שונים
- אני מציב את הקומפוננט PageHeader עם הכותרות המתאימות

• ייצור אזור רספונסיבי באמצעות הקומפוננט Grid

- נקבע כי החלק של המיל יתפօס את כל רוחב המסך בגודל מסך קטן ויתפօס 8 עמודות מעל גודל מסך md ונמראץ את התוכן לאמצע באמצעות "alignSelf="center"
- נקבע את החלק של התמונה כך שיוצג לגולש רק מגודל מסך md ונמראץ אותו
- מחרוזת התווים בערך של מאפיין src תתחילה בסימן סלאש / כדי ש react תתחילה את החיפוש של בתיקיית public ולאחר מכן נשלים את הנטייה עד לתמונה המבוקשת

```
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         
27       </Grid>
28     </Grid>
29   </Container>
30 }
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48 export default AboutPage;
```

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 pariatur 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 preferendis tenetur? Debitis nihil facere quos? Debitis molestias quae voluptatum. Elus preferendis 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 quaerat deserunt totam praesentium eaque voluptatem pariatur neque porro, accusantium consequuntur, exercitationem quisquam? Itaque praesentium beatae consectetur, quisquam facilis qui laboriosam voluptate maxime cupiditate voluptas et nisi?



התוצאה בדף

ניתן לראות כי בהתאם לתבנית שקבענו

- אנחנו רואים את כוורות הדף
- את החלק המיועד לטקס המסביר על האפליקציה
- וראים את התמונה של הלקוח

Hooks

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

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



Introducing React Hooks

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



Rules

Only Call Hooks at the Top Level

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

Call Hooks from React function components

Call Hooks from custom Hooks

Call hooks from the lowest possible component in the component tree

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

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



useState

Hook שתפקידו הוא לנהל את האובייקט State



useState

בדוגמה שלහן ניצור counter בעזרת hook

- נחלץ את מетодת useState מספרית react

- ניצור קומפוננט בשם SetCounter

- נשא array destructor לערך שיחזור אליו מהפעלת המפתחות:

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

- מצב את הערך של המשתנה counter

- ניצור כפתור שבלחיצה עליו יפעיל פונקציה אונומית
- הפונקציה האונומית תפעיל את מетодת setCounter שჩילצנו

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

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

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

```
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 תמורה את ערכו של useState counter בעזרת counter

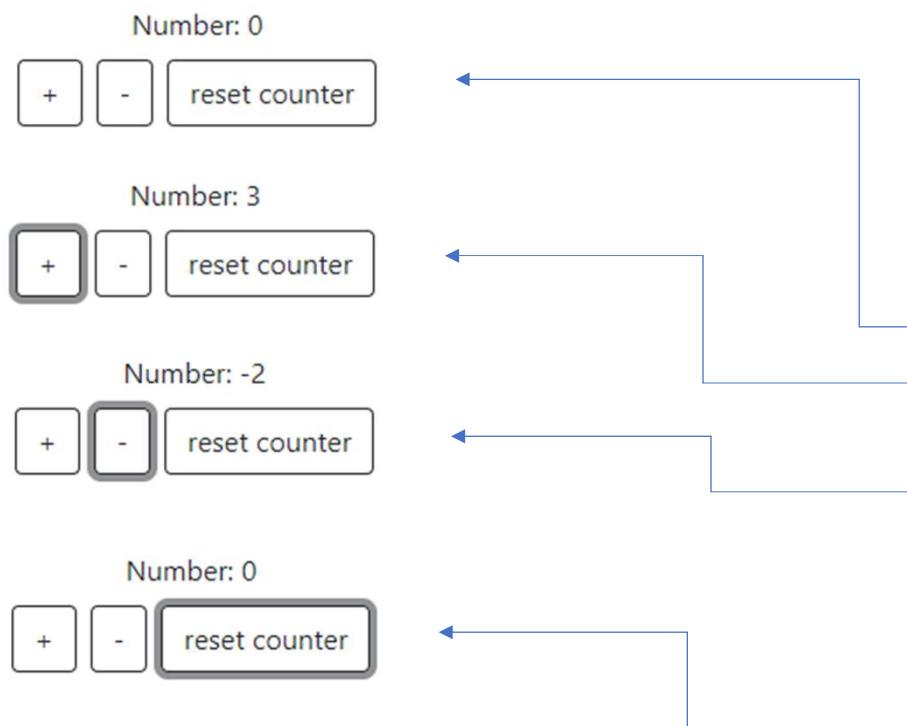
- ניצור קומפוננט בשם SetFunction
- נעשה array destructor לערך שיחזור אליו מהפעלת מטודת useState עם הפורמט `0` ונחלץ ממנו את המפתחות:
 - `count` – שם המשתנה
 - `setCount` – מטודה שאחראית על שינוי ערכו של המשתנה
- ניצור פונקציה בשם changeNum ש
 - קיבל בפורמטר `term`
 - תונה ואם הערך של `term` הוא מחזוזת התווים `"increment"` היא תבצע ותפעיל את מטודת `setCount` באחד ותעלה את המספר באחד
 - תונה ואם הערך של `term` הוא מחזוזת התווים `"decrement"` היא תבצע ותפעיל את מטודת `setCount` באחד וטוריד את המספר באחד
 - ואם היא מקבלת משהו אחר היא תאפס את הערך של `count`
- הפעם שנחלץ על הceptor 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  
37 export default SetFunction;
```

התווצה בדף

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

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



useState with objects

בדוגמה שלහלן 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 לערך שיחזור אליו מഫעלת מетодת useState עם הparameter initialValue ונחלץ ממנו את המפתחות:

- name – שם המשתנה
- setName – מטודה שאחראית על שינוי ערך של המשתנה

• נציב את ערכיו של המשתנה name כך שיוצגו לגולש

• ניצור אלמנטים מסוג Input שיכריכו אליהם ערך ה-

- תקבל את האירוע
- תפעיל את מטודת setName כאשר בפרמטר נעביר לה אובייקט אליו אנו מעתיקים את כל המפתחות מאובייקט name ואז משנים את הערך של המפתח הרלוונטי בהתאם לנתחים שהוכנו לאלמנט input

```
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  export default SetObject;
```

התוצאה בדף

ניתן לראות כי בהתאם לתבנית שקבענו מוצג לגולש

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

Your Name Is: David Yakin

David

Yakin

useState with complex objects

בדוגמה שלහן ניצור טופס שיבקש מהמשתמש את השם 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 שערךם הוא מחוזות תווים

• email – מסוג מחוזות תווים

• נעשה array destructor לערך שיחזר אלינו מהפעלת מטודת useState עם ה Parameter INITIAL_USER ומחילץ ממנו את המפתחות:

- user – שם המשתנה

• setUser – מטודה שאחראית על שינוי ערכו של המשתנה

• נציב את ערכיו של המשתנה user כך שיוצגו לגולש

• ניצור אלמנט מסוג `setInput` שכאשר יוכנסו אליו נתונים הוא יפעיל פונקציה אוניברסלית שתקבל את האירוע ותפעיל את מטודת setUser וונביר לה אובייקט ש:

- נתיק לתוכו את מפתחות האובייקט user

• נעשה השמה למפתח name ונשווה את ערכו לאובייקט שלו

ונעתק את המפתחות של האובייקט user.name

• נעשה השמה למפתח first בתוך הערך של אובייקט name כך שערך יהיה שווה לערך שיכנס לאלמנט input (e.target.value)

• ניצור אלמנט מסוג `input` שיתיק את המפתחות של user ויעשה שמה למפתח email

```
SetComplexObject.jsx U
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
51 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 התשתית

- ניבא את useState מספרית react
- ניצור קומפוננט בשם SetArray ומשווה את הערך שלו לאובייקט עם מפתחות וערכים ראשוניים
- נפעיל את מетодת useState עם הקבוע task setTask שיצרנו ונחלץ ממנו את מערך tasks setTasks
- נפעיל את מетодת useState עם מערך tasks setTasks ריק ונחלץ ממנו את פונקציה createNewTask בשם createNewTask
- שתקבל אירוע
- תבצע את ההתנהלות הדיפולטיבית של שליחת הטופס על ידי הceptors
- שתפעיל את מетодת setTasks כشارוגמנט נועתיק את מערך המשימות ונוסיף את המשימה
- נוצר את הפונקציה ונאפס את המשימה על ידי הפעלת מетодת setTask עם הקבוע של הערכים הראשוניים

SetArray.jsx M X

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

render

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

- ניצור טופו
- בחלקן העליון נציג לגולש את המשימה
- ניצור כפטור ש:
- הוא יהיה מנווט אם לא יהיה ערך ב מפתח todo של אובייקט task
- בלחיצה על הכפטור הוא יפעיל את מетодת createNewTask שיצרנו
- ניצור input
- שתזין לאירוע onChange ותפעיל פונקציה אונומית שתקבל את האירוע
- תפעיל את מетодת setTask כאשר בארגומנט נתיק את המפתחות מתוך אובייקט task ונעשה השמה למפתח todo עם מה שייכתב בתוך האלמנט
- הערך של האלמנט יהיה הערך של המפתח task.todo
- ניצור רשימה ש:
- על כל איבר במערך tasks היא תיצור רשומה עם מספר ועם פרטי המשימה

התוצאה בדפס

ניתן לראות כי בהתאם למבנה שקבענו

- כשמלאים את הכתוב בשדה הטקסט מוצג בראש הטופס
- כשלוחצים על הכפתור המשימה מוצגת כרשומה והשדה מתוקה

Task: משימה שלישית

Create

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

Task:

Create

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

שימוש useState



Hooks-Sandbox

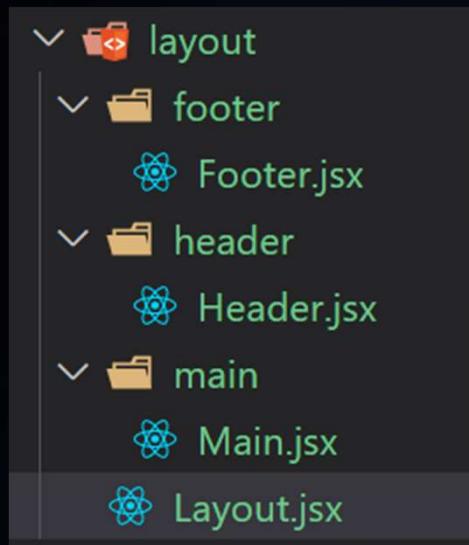
- צור קומפוננט בשם SetPost
 - ייבא את useState למודול
 - צור פונקציה בשם SetPost (קומפוננט)
 - צור קבוע בשם INITIAL_POST שערך יהיה אובייקט עם המפתחות:
 - title – מסוג מחרוזת תווים
 - subtitle – מסוג מחרוזת תווים
 - author – מסוג מחרוזת תווים
 - createdAt – מחרוזת תווים
 - הפעיל את מتدת useState פעמיים כאשר בפעם הראשונה עם הארגומנט INITIAL_POST וחלץ מהערך שבודק שמדובר במקרה post setPost את posts setPosts
 - בפעם השנייה עם מערך ריק כארגון וחלץ מהערך שבודק מהפעלת הפונקציה posts setPosts
 - הצג לגולש:
 - טופס שבתוכו
 - צור שלוש אלמנטים מסוג `Post` שהנתונים שיוצנו בתוכם יהיו מקושרים לערכים של שלושת המפתחות הראשונים של post post חדש (כמו שהוצג בשקפים הקודמים)
 - צור כפטור שלחיצה עליו תכניס פוסט חדש למערך הפוסטים
 - טבלה שתציג רק אם יש פוסטים במערך הפוסטים

Layout

פריסת האפליקציה



הכנות לשתייה



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

Header.jsx X

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 הקומponent כר שתציג
את התוכן שלו בתבנית הכללית

Footer.jsx

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 הקומponentן הראשון שתחזיג את התוכן שלו בתבנית הכללית

Main.jsx

קומפוננט זה תהיה אחראית על תצוגת התוכן

- הקומפוננט 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 ווי שאני קבע גובה מינימלי וצבע רקע
- בນקודה זאת התוכן יוצג
- אני מודא בעזרת ספריית propTypes שהקומפוננט אכן מקבלת אלמנט של בפתח children שבאובייקט הפרופס

```
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

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

- הקומפוננט מקבל בפתח של אובייקט הפוך את המילה השמורה children
- נעתוף את האלמנטים בקומפוננט באמצעות React.Fragment

- נציג לגולש את הקומפוננטות:
 - Header – שתכלול תפריט ניוט בעתיד
 - Main – אליה נעביר את התוכן של הדף שברצוננו להציג
 - Footer – תכלול תפריט ניוט בעתיד לוגו וזכויות יוצרים

- אני מודא בעזרת ספרית propTypes שהקומפוננט אכן מקבלת אלמנט של React בפתח children שבאובייקט הפוך

! כרגע התוכן של הדף הוא עדין סטטי עד שנלמד בהמשך המציג לשימוש בניתובים

```
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;
```

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

- נטעף את הקומפוננט **CardsPage** כרך שיוצג לנו תוכן הדף **Header Footer** עם **CardsPage** בלבד

! מעשה אנו מעבירים לkomponent **Layout** בפתח **children** באובייקט ה – **props** את הקומפוננט **React** **CardsPage** שהוא אלמנט של **main** בתוך הקומפוננט **Layout** מוצב הקומפוננט **main** שמקבל גם הוא את אלמנט ה **React** בפתח **children** באובייקט ה – **props** ומציג אותו

header

Cards

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



first

subtitle

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



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



התוצאה בדף

ניתן לראות כי בהתאם לתבנית
שקבענו מוצג לגולש

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

Footer

ErrorPage.jsx

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

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



client > src > pages > ErrorPage.jsx

```
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 שיסביר לגולש שהדף שהוא ביקש לא נמצא
- כפתור שמאחור יותר יהיה לינק חזרה לדף הבית
- תמונה להמחשה שהגולש הגיע לדף 404

BCard ABOUT MY CARDS FAV CARDS

Search 

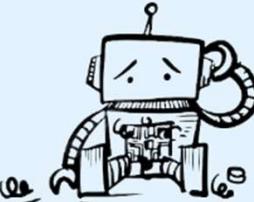


Error 404

Page not found

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

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



About  Favorites  My Cards 

התוצאה בדף

ניתן לראות כי בהתאם לתבנית שקבענו

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

React Router Dom

ניתובים באפליקציה שהוא Single Page Application

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





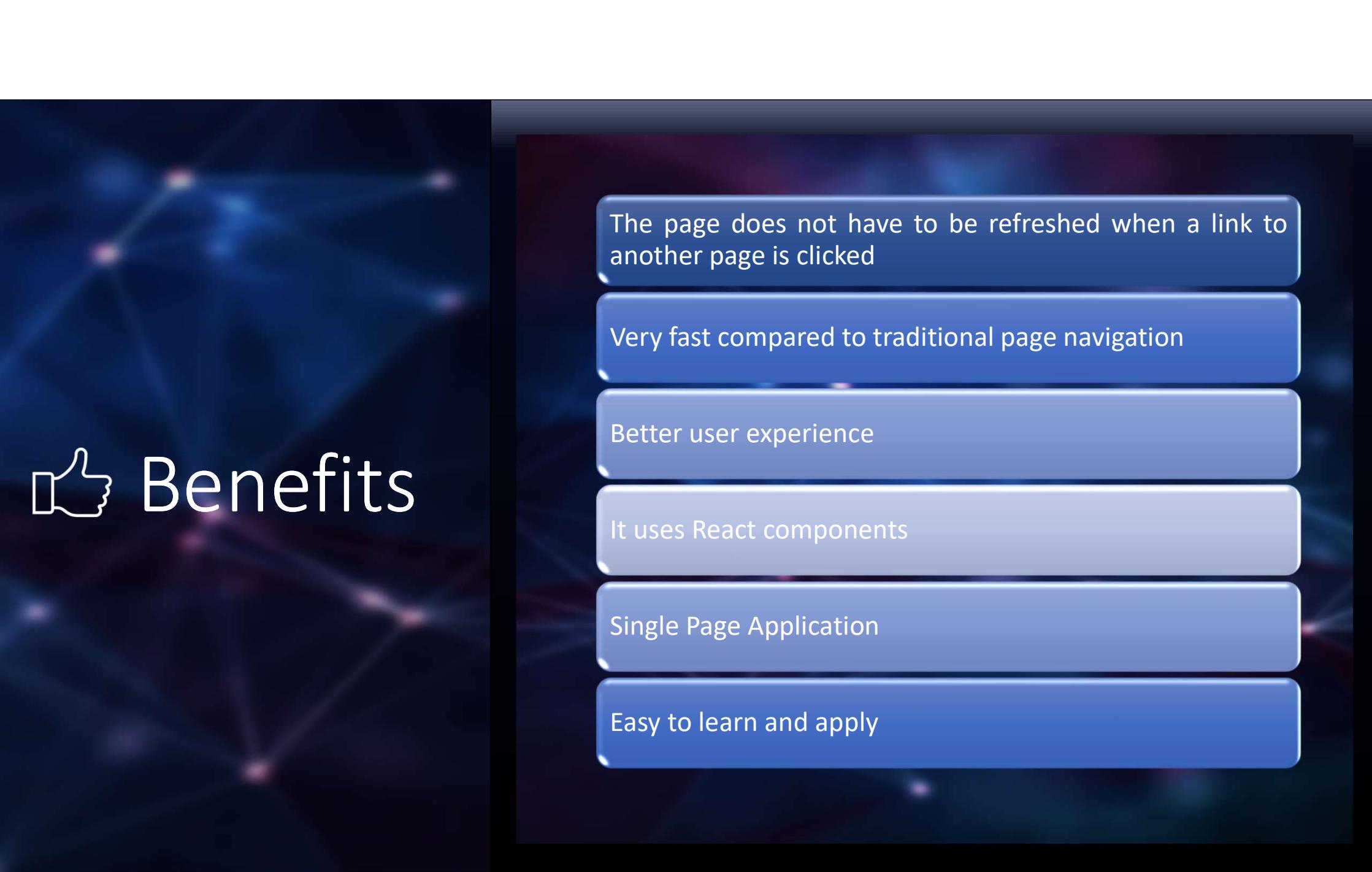
Q Definition

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.



thumb up 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

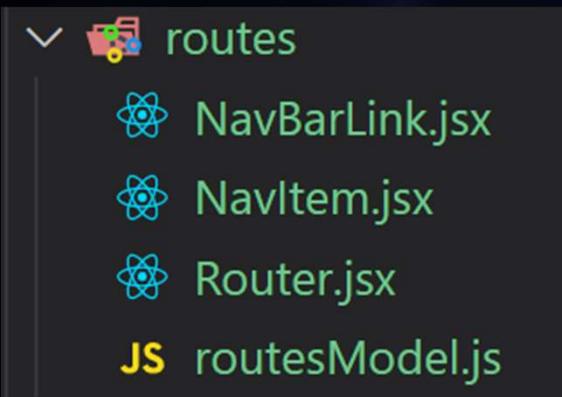
Easy to learn and apply



Installation

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

הכנות תשתית



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

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

routesModel.js

מודול זה ישמש כשפה משותפת לכל
שמות הLINKS וערוצים

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

- ניצא את הקבוע שיצרנו מהמודול

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



Routes

קומפוננט של `react-router-dom` שאחראי על
ניתובים



Router.jsx

קובץ זה ינהל את תצוגת הדפים לגולש

- ניבא את הkomponenot **Routes** **react-router-dom** **Route** **Route** מספריית

- ניצור קומפוננט בשם **Router**

- געטוף את komponenot ה – **Route** **Routes** בkomponent **Routes**

- כל komponent מסווג **Route** מקבל בשלב זה שני מאפיינים:

- path – כתובות URL
- element – komponent שנרצה להציג במידה ומגיעים לכתובת ה – URL של komponent

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

יש חשיבות לסדר komponenot של **Route** !

Router.jsx

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



BrowserRouter

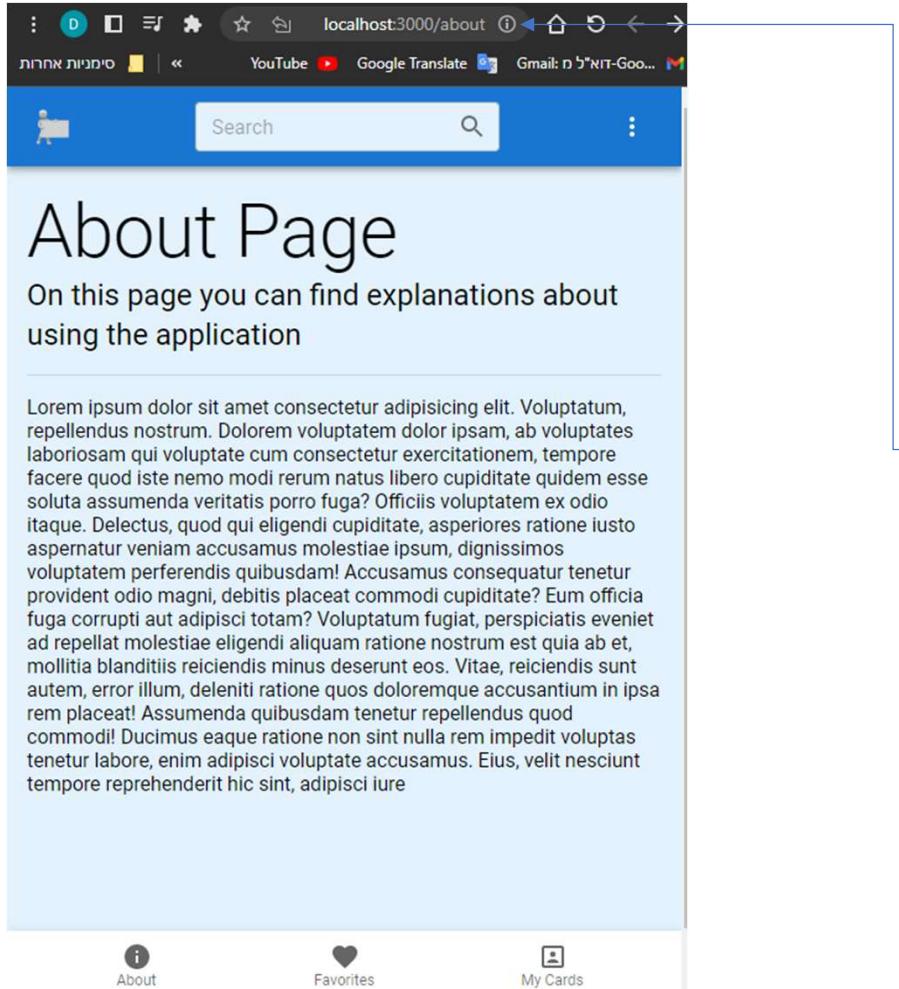
קומפוננט של `react-router-dom` שתנתב לדף המתאים לפי הלוגיקה הרשומה בקומפוננט `Routes`



```
JS App.js 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;
```

BrowserRouter

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



התוצאה בדף

אם נקיש בשורת הכתובות לאחר כתובות
ה DNS והפורט את כתובות ה URL של
/about נعبر לדף אודות



Link & NavLink

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



NavBarLink.jsx

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

קומפוננט שيعוטף אלמנט כרך שלחיצה עליו
תעביר לכתובת ה – URL המוגדרת

- ניבא את Link מספרית-dom
- נוצר קומפוננט בשם NavBarLink
- שתקבל באובייקט הפרופס את המפתחות:
 - URL – to
 - color – הצבע הכתוב של האלמנט שנעטוף
 - children – אלמנט React שהקומפוננט תעוטף
- הקומפוננט תחזיר את הקומפוננט Link
шибאנו כאשר מאפיין to יהיה שווה
ערך למפתח to מאובייקט הפרופס,
ונקבע כי המאפיין style קיבל את
הצבע שנעביר לאובייקט הפרופס
ונקבע שלא יהיה קו תחתון לאלמנט
שאנו עוטפים
- השתמש במספרית PropTypes כדי
לודא שהקומפוננט קיבל את כל מה
שהיא צריכה
- נקבע את הצבע הלבן כערך
הדיופולטיבי 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

קומponent שתשתמש בקומponent
NavBarLink שיצרנו

- ניבא את NavBarLink
- ניבא את ROUTES מתוך המודול
שיצרנו לURL
- ניצור קומponent בשם Logo שתחזיר
 - געטוף את הקומponent Typography בקומponent NavBarLink שיצרנו ונעביר לו בפורמטר את כתובות ה – URL המתאימה מתוך אובייקט ROUTES
- נקבע עיצוב מיוחד ללוגו

NavItem.jsx

```
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;
```

קומponent המועד לשימוש בתפריט
נירוט

- ניבא את הקומponent NavBarLink
- נוצר את הפונקציה NavItem
- הפונקציה תחלץ מאובייקט הProps
את המפתחות:
 - label – מסוג מחוץ תווים
 - to – מחוץ תווים של URL
 - color – מחוץ תווים
- עטוף את קומponent הפתור
בקומponent NavBarLink שתקבל
במאפיינים שלה את כתובת ה – URL
אליה היא צריכה להעביר את הגולש
ו את הצבע של הכתוב על הפתור
- נודא שהקומponent מקבלת את כל
המאפיינים שהיא צריכה כדי
שהלוגיקה שלה תפעל נכון בעזרת
ספריית PropType

LeftNavigation.jsx

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 בקומponent 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



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



forth
subtitle

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

 About

 Favorites

 My Cards

התוצאה בדף

עכשו הLINKים שלנו בתפריט הניווט
עובדים, לחיצה עליהם תעביר את הגולש
לדף המבוקש

משימת react-router-dom



Business-cards-app

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

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



useNavigate

של Hook react-router-dom שעזר לנו בניתו



useNavigate

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

- ניבא את ROUTES למודול react-router-dom מ - useNavigate
- ניצור קבוע בשם navigate שערך יהיה הערך שייחסור מהפעלת מטודת useNavigate
- ניצור קבוע בשם navigateTo שם הערך יהיה שמו של מהערך של פונקציה שמקבלת כפרמטר to מסווג של מחוזת תווים
- ותפעיל את navigate עם מחוזת התווים שהועברה לפונקציה בפרמטר to
- בקומפוננט שלחיצה אליה נרצה להעביר את הגולש לדף אחר נאזרן לאיירע onClick שיפעל פונקציה אונומית שתפעיל את פונקציית navigateTo שיצרנו עם כתובות ה – URL שתנתב 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
40 export default Footer;
```



Navigate

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



SignupPage.jsx

SignupPage.jsx x

```
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 מותך-react-router-dom
- ניבא את ROUTES
- ניצור משתנה בשם user ונסווה את הערך שלו פעם אחת – 이후 פעם אחת true
- ניצור התנינה ואם הערך של user הוא לא פולטיבי נחזיר מהקומפוננט את הקומפוננט Navigate של-react-router-dom ובעזרת המאפיין replace נחליף את כתובת URL לכתובת הרשומה במאפיין to 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;
```

Router.jsx

- ניבא את הקומponent SignupPage
- ניצור Route חדש שיעביר אותנו לkomponent SignupPage במידה ושורת הכתובות משתנה לכיתובות 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 הcarteisim



useParams

האובייקט `params` משורט 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

- נוצר את הנתיב src/cards/pages/CardDetailsPage.jsx
- ניבא את useParams מספריית react-router-dom
- נחלץ מהאובייקט שיחזור מהפעלת המטודה useParams את מפתח id
- נציג לגולש את תעודת הזיהות של הלקוח שאת פרטיו נציג בקומponentה

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

- נוסיף את המפתח CARD_INFO

Router.jsx

- נוצר Route חדש שיעביר אותנו לkomponent CardDetailsPage במידה ושורת הכתובות משתנה לכתובת ROUTES.CARD_INFO אשר בפתח ROUTESSIGNUP ונצפה 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;
```

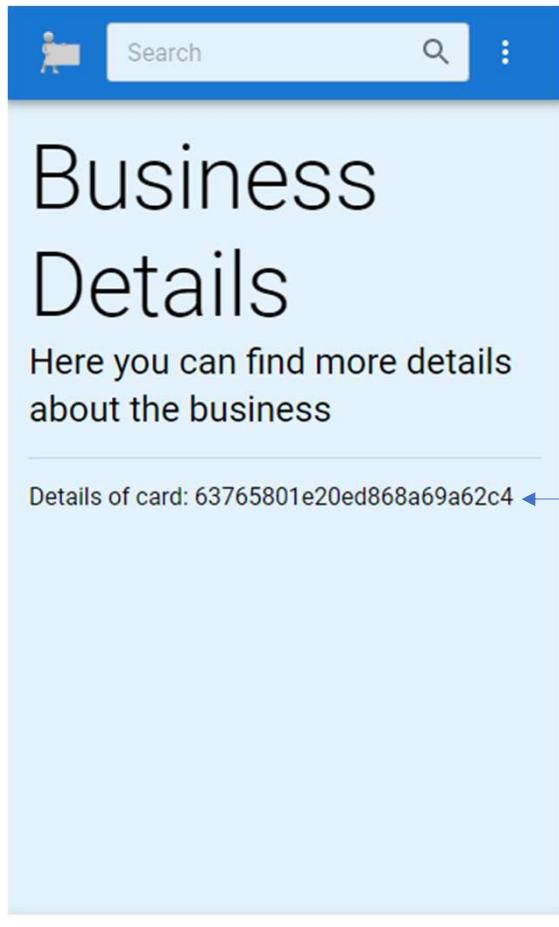
Card.jsx

בקומפוננט זה נוצר את הלוגיקה כך
שבלחיצה על כרטיס עסוק ספציפי נועבר
לדף עם הפרטים על הלקוח

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

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

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



התווצה בדף

לחיצה על כרטיס הובילו אותנו לדף CardDetails.js
שמציג את קומפוננט `szs`.
ומוצג לנו תעודת זהות של הלקוח
בתוכה אותה הקומפוננט לkerja
מאובייקט ה `params` באמצעות המטודה
`useParams`



Nested Routes

הדרך לייצר תפריט ניווט 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

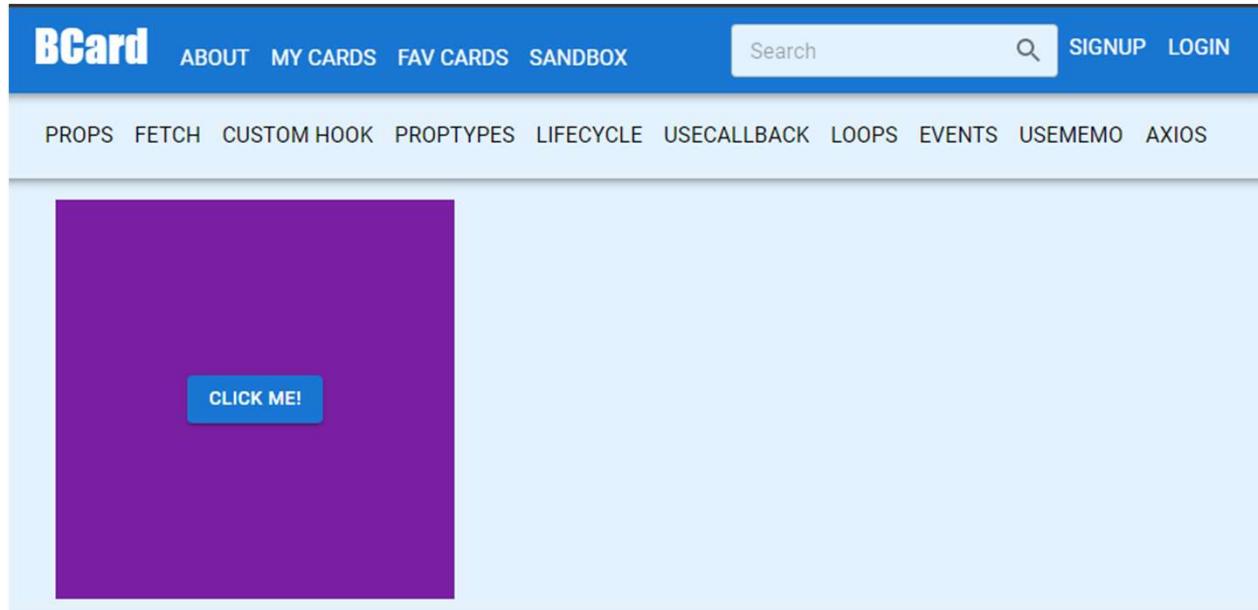
נותנת לנו דרך קלה ונוחה ליצור nested Route . ישנן שלושה שלבים כאשר השלב הראשון הוא עטיפת הקומפוננטות Route בקומפוננט Route הראשית. בדוגמה שלהן:

- יצרנו תגית סגורה 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 הרצטיבית שלהם (ללא סלאש בתחילת כתובות ה - url)

Sandbox.jsx

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

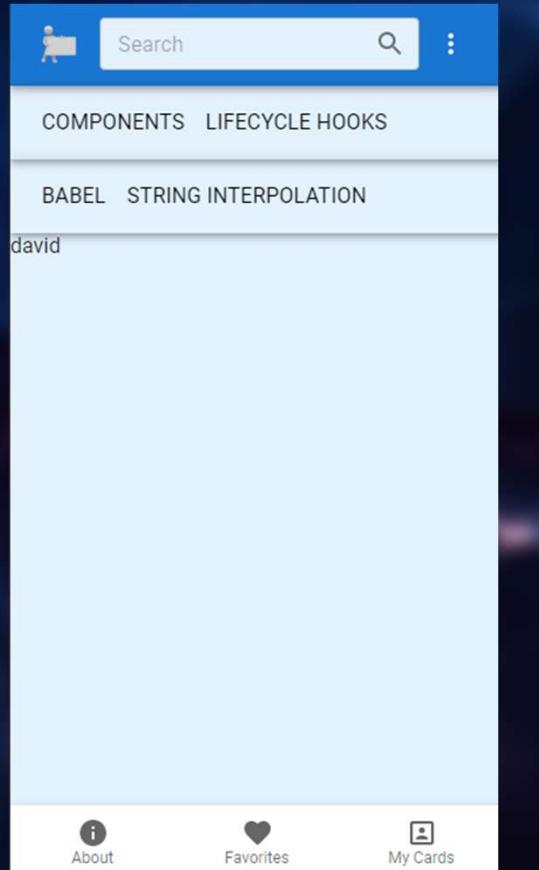
```
client > src > sandbox > Sandbox.jsx > ...
1 import AppBar from "@mui/material/AppBar";
2 import Toolbar from "@mui/material/Toolbar";
3 import NavItem from "../../routes/NavItem";
4 import { Outlet } from "react-router-dom"; ←
5 import Container from "@mui/material/Container";
6
7 const Sandbox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



התוצאה בדף

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

משימת Nested Routs



Business-cards-app

- צור תפריט ניוט בкомпонент Sandbox כך של כל תיקייה בתוך תיקייה sandbox יש ליצור תפריט ניוט משלה עם התצוגה המתאימה לה כפי שownfu בדוגמה מצד שמאל