

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 עוקב 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>



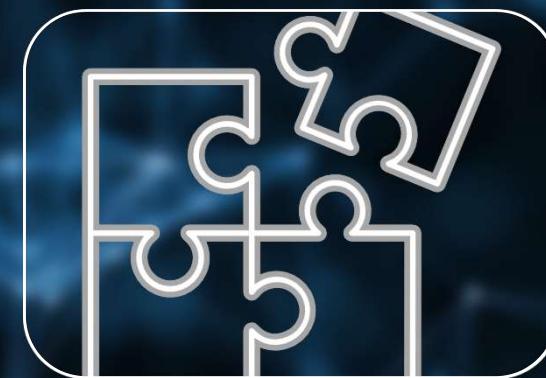
Virtual DOM implementations



State



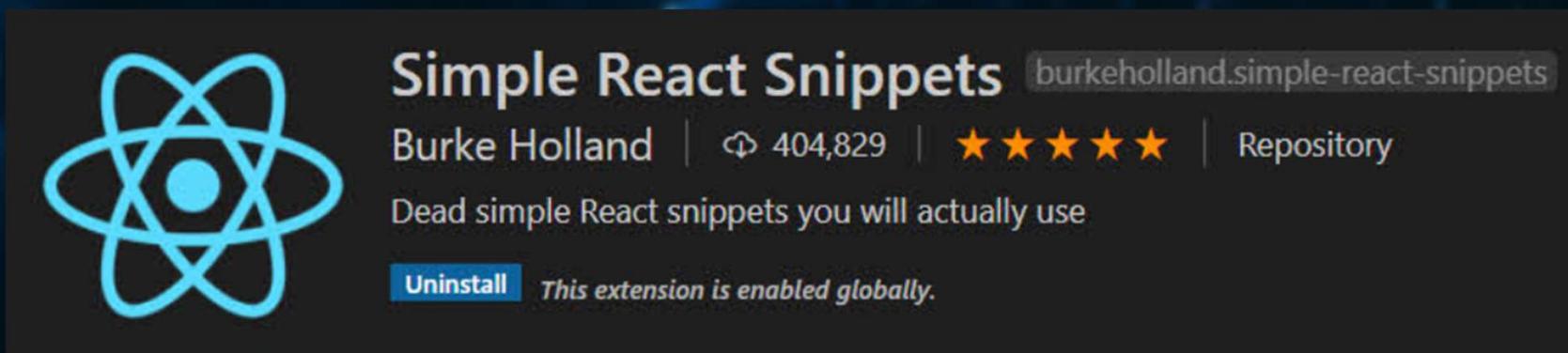
Life cycle
hooks



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

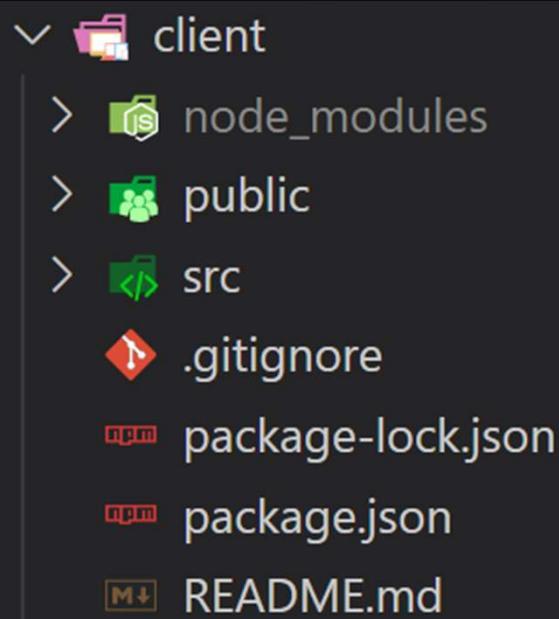
לאחר שהורדנו והתקנו את התוסף
create-react-app נוכל להשתמש בו
כדי לפתח פרויקט חדש ב - React

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

- נקיש בטרמינל את הפקודה
create-react-app client

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

חשתית הקבצים שה-CLI יוצר

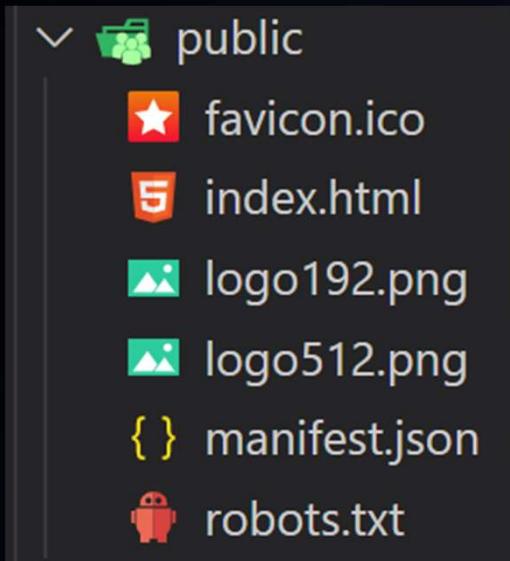


- **Node_modules** – תיקייה השומרת בתוכה את הספריות שהורדנו לפרויקט
- **public** – תיקייה השומרת את הקבצים הסטטיסטיים
- **src** - תיקייה בה רוב הקוד של הפרויקט יכתב
- **.gitignore** - קובץ קונפיגורציות של git ובתוכו ההוראות מאיילן תיקיות וקבצים להעתלם ולא להעלות ל- github
- **package.json** - קובץ קונפיגורציות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

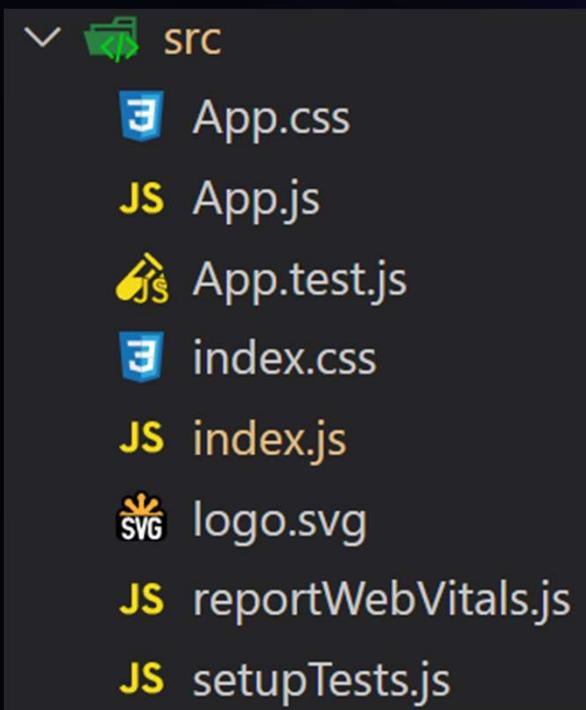


index.html

- 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 ונוכל לראות את התצוגה הראשונית של האפליקציה



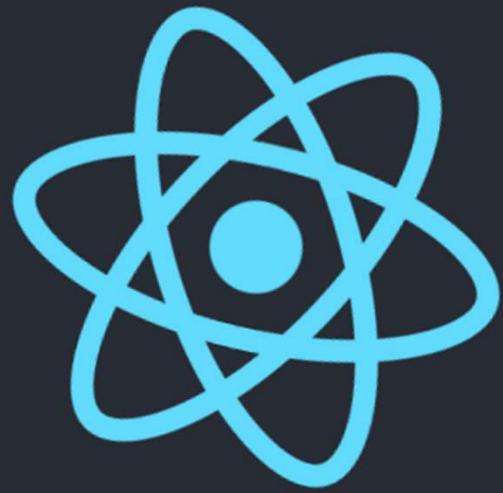
npm start

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 i | Developer Tools | 3,000,000+ users

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

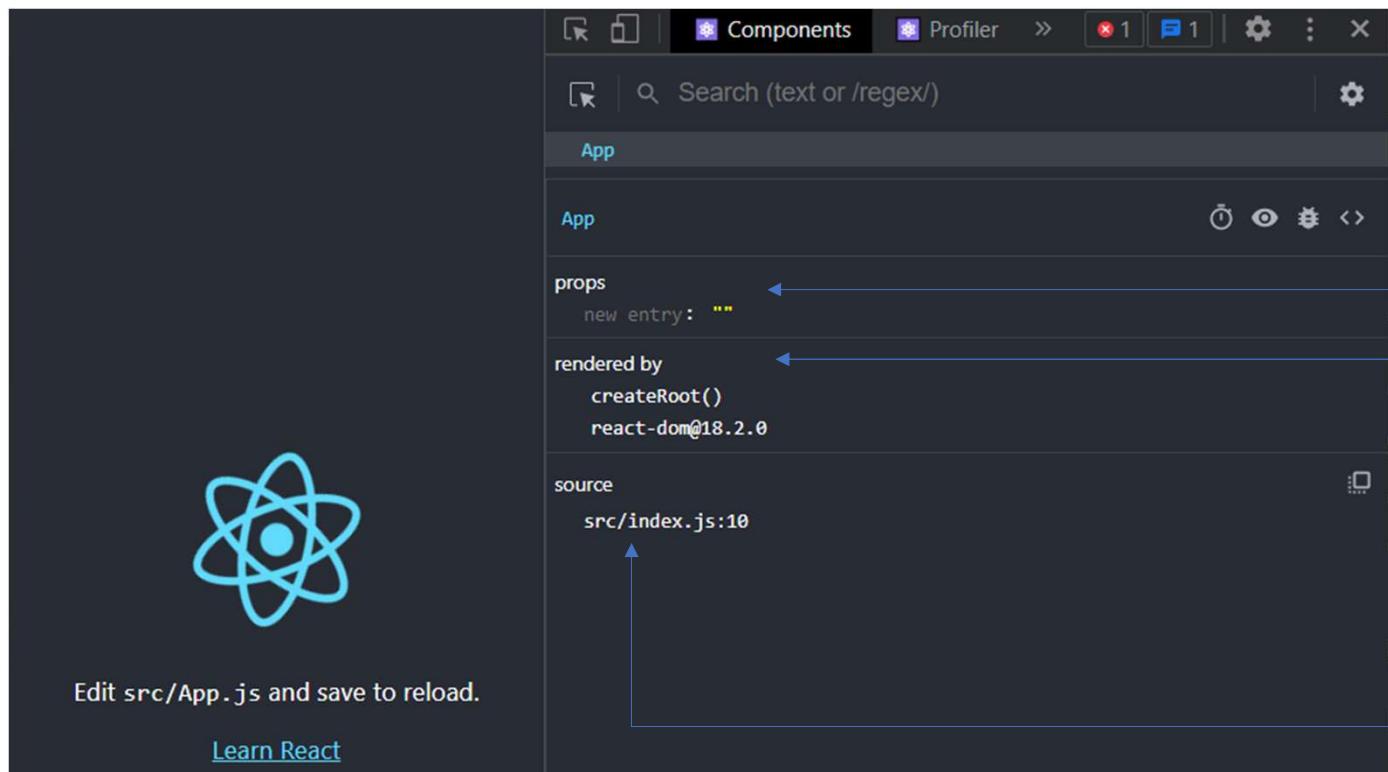


Components

לשונית 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 ש React תגיב לשינויים בערכיהם
 - rendered by – הפונקציה שאחראית על טעינה וטעינה מחדש של הקומפוננט
 - source – קובץ המקור

! בغالל שהוא לא מנהלים state בקומפוננט זהה אני לא רואה את הנתונים הללו



Profiler

The screenshot shows the React DevTools Profiler interface. At the top, there's a header with tabs for Components and Profiler, and a toolbar with various icons. A message says "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 on the left and a timeline or details panel on the right. The timeline panel shows a single entry: "Profiling is in progress...". The details panel shows a list of components with their rendering times:

Component	Rendered at
BrowserRouter	3.2s for 14.2ms
Router	3.9s for 10.6ms
Navigation.Provider	4.6s for 13.2ms
Location.Provider	5s for 32.1ms
App	7.2s for 2.5ms
Header	8.8s for 11ms
NavBar	
Navbar (ForwardRef)	
NavbarContext.Provider	
Context.Provider	
Container (ForwardRef)	
NavbarCollapse (ForwardRef)	
Anonymous (ForwardRef)	
Anonymous (ForwardRef)	
Transition	
Content	
Nav (F...	
ToastContainer	Main Routes
Toast	

בלשונית זאת נוכל לעשות בדיקות אופטימיזציה

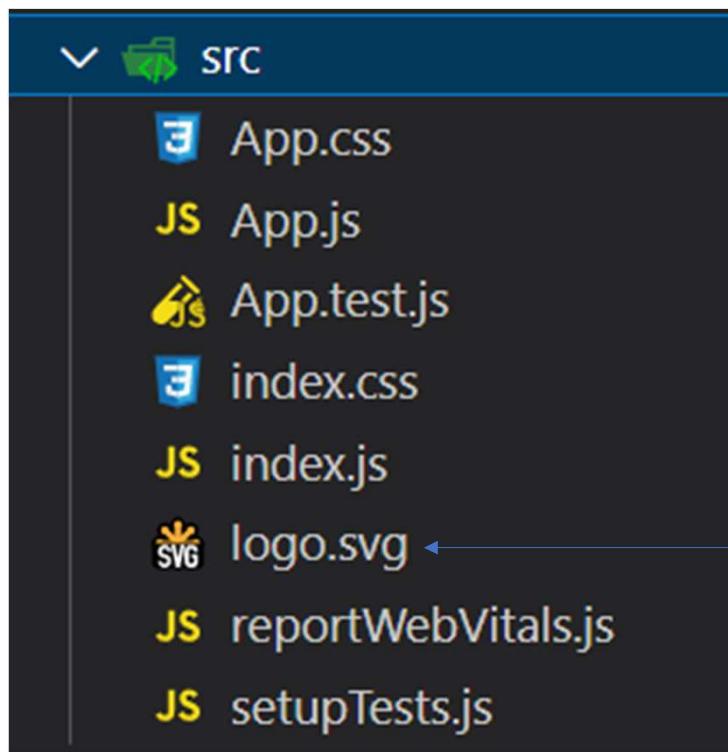
- לחיצה על כפתור **record** תתחיל להאזין לאירועים שוקרים ב – DOM
- לחיצה נוספת תעצור את ההקלטה ובמידה ויהיו נתונים להציג (כמו משך הזמן שלקח לכל אירוע לפועל) החלקים הללו יוצגו לנו.

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

תחילת עבודה

ניקוי ראשוני של האפליקציה מתחנות וערכיהם דיפולטיביים 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

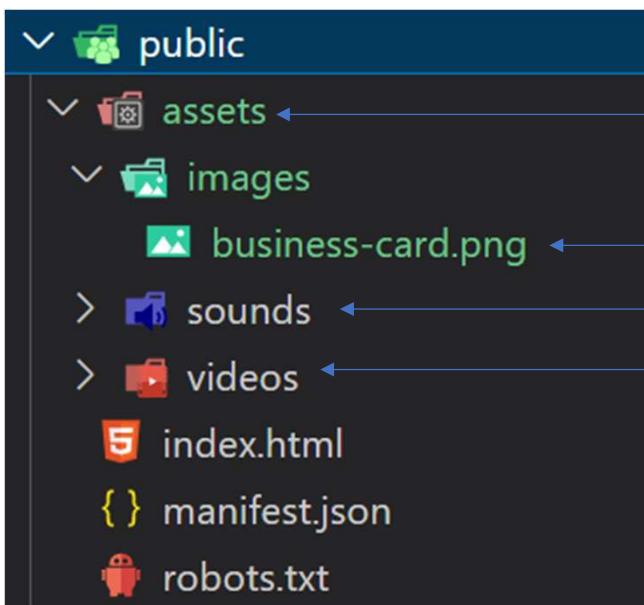
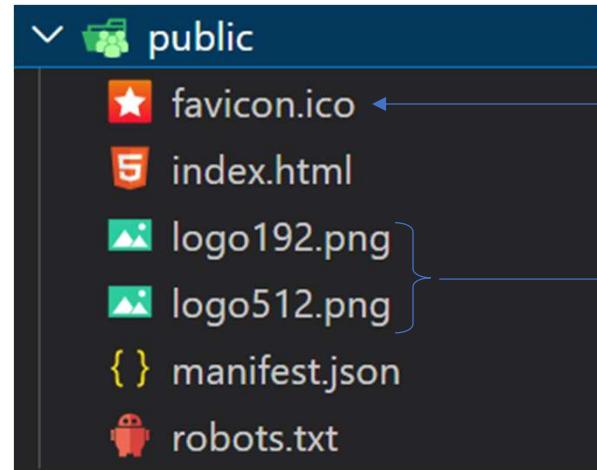
```
index.css X  
src > index.css > body  
1 body {  
2   margin: 0;  
3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',  
4   |   'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',  
5   |   sans-serif;  
6   -webkit-font-smoothing: antialiased;  
7   -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 הקומpileציה
- מסך אמצעי – משמש לכתיבה קוד javascript דקלרטיבי ועדכני
- תפירט צידי – ובו אפשרותויות שונות לתצוגת הקוד לאחר קומPILEציה

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 is a 'Source Type' dropdown set to 'Module'. Under 'TARGETS', it lists 'defaults, not ie 11, not ie_mob' and '11'. The main area has two code blocks. The top one contains the handwritten code: '1 <div>', '2 hallo', and '3 </div>'. The bottom one contains the generated code: '1 "use strict";', '2', and '3 /*#__PURE__*/React.createElement(React.Fragment,' followed by a long string of characters. Arrows from the text on the right point to both the handwritten and generated code blocks.

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

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

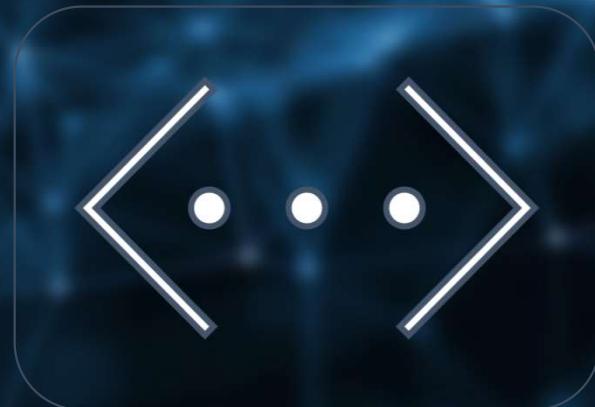


LOGIC
JAVASCRIPT



STYLES
CSS

Components Types

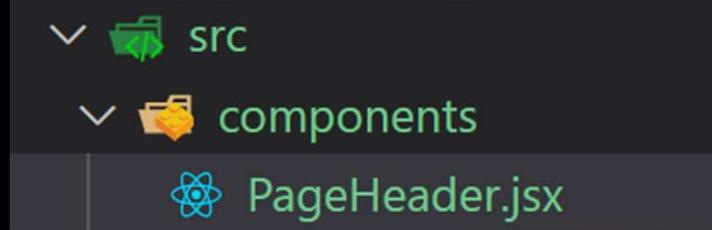


Function



Class

הכנות תשתית



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

! קומפוננט תמיד תחיל באות גדולה
! סימנת `jsx / tsx` אוילו 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 הכתובת 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 מחזירה.

! במצגת זאת נתמקד בקומponentות מסוג **functional components** של **React** או של HTML
חווב לעתוף את כל האלמנטים שהפונקציה מחזירה באלמנט או של

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

The screenshot shows a file tree on the left and a code editor on the right. Red arrows point from the following elements in the file tree to specific parts of the code editor:

- A red box surrounds the "components" folder in the file tree, pointing to the "PageHeader.jsx" file in the code editor.
- A red arrow points from the "PageHeader.jsx" file in the file tree to the "PageHeader.jsx" file in the code editor.
- A red arrow points from the "PageHeader.jsx" file in the code editor to the line of code containing the error: <h2>pageHeader works!</h2>.

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

PageHeader.jsx

```
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

The screenshot shows the VS Code terminal window. At the top, it says "Local: http://localhost:3000". Below that, it says "Failed to compile." followed by a red arrow pointing to the error message. The error message is a "SyntaxError" from Babel loader, stating that adjacent JSX elements must be wrapped in an enclosing tag. It shows a code snippet with lines 2 through 7. Line 2 starts with "return (", line 3 has "

pageHeader works!

", line 4 has "

hallo world

", line 5 has a closing brace ")", and line 6 has a closing brace "};". A red arrow points from the error message to the opening brace on line 5. At the bottom, it says "ERROR in ./src/components/PageHeader.jsx" followed by a red arrow pointing to the file name. Then it says "Module build failed (from ./node_modules/babel-loader/lib/index.js):" followed by another red arrow pointing to the module name. Finally, it repeats the "SyntaxError" message from the top.

The screenshot shows the VS Code problems panel. At the top, it says "PROBLEMS 2". Below that is a toolbar with icons for filter, close, and other options. A "Filter (e.g. text, **/*ts, !**/node_modules/**)" input field is present. The main area lists two errors for "PageHeader.jsx" in "src\components":

- ① JSX expressions must have one parent element. ts(2657) [Ln 3, Col 3]
- ② Parsing error: Adjacent JSX elements must be wrapped in an en... eslint [Ln 4, Col 2]

Red arrows point from the error descriptions in the terminal screenshot to the corresponding entries in the problems panel.

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\lib\index.js:3364:19)
      at FlowParserMixin.jsxParseElementAt (C:\Users\DELL\lib\index.js:7210:18)
        at FlowParserMixin.jsxParseElement (C:\Users\DELL\lib\index.js:7220:17)
          at FlowParserMixin.parseExprAtom (C:\Users\DELL\lib\index.js:7233:19)
            at FlowParserMixin.parseExprSubscripts (C:\Users\DELL\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;
```

The screenshot shows a browser's developer tools with the 'Elements' tab selected. The page content is displayed above the tool, showing the text "pageHeader works!" and "hallo world". Below this, the DOM tree is shown:

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

A green bracket on the right side of the code block points to the curly brace at the end of the first div element in the DOM tree, indicating where the error occurred.

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

- אם אני לא מונין לעוטוף את שני האלמנטים באלמנט עיצובי של HTML כמו div ריאקט pseudo element משלה שנקרא React.Fragment לעוטוף את האלמנטיםอลם האלמנט זהה 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 <></>

React.Fragment



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

top

1 hidden

Filter

Custom levels

1 Issue: 1

30

PageHeader.jsx:3

30

VM1241:236

>



String interpolation

יצירת אזור JAVASCRIPT באזורי 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 לאזרע 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:

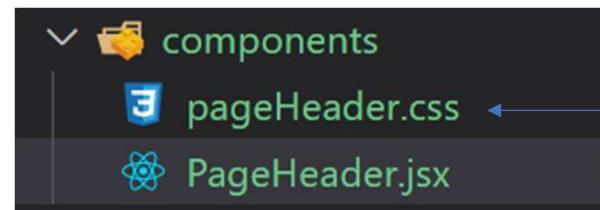
- ניצור קובץ חדש בשם pageHeader.css

- 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;
```

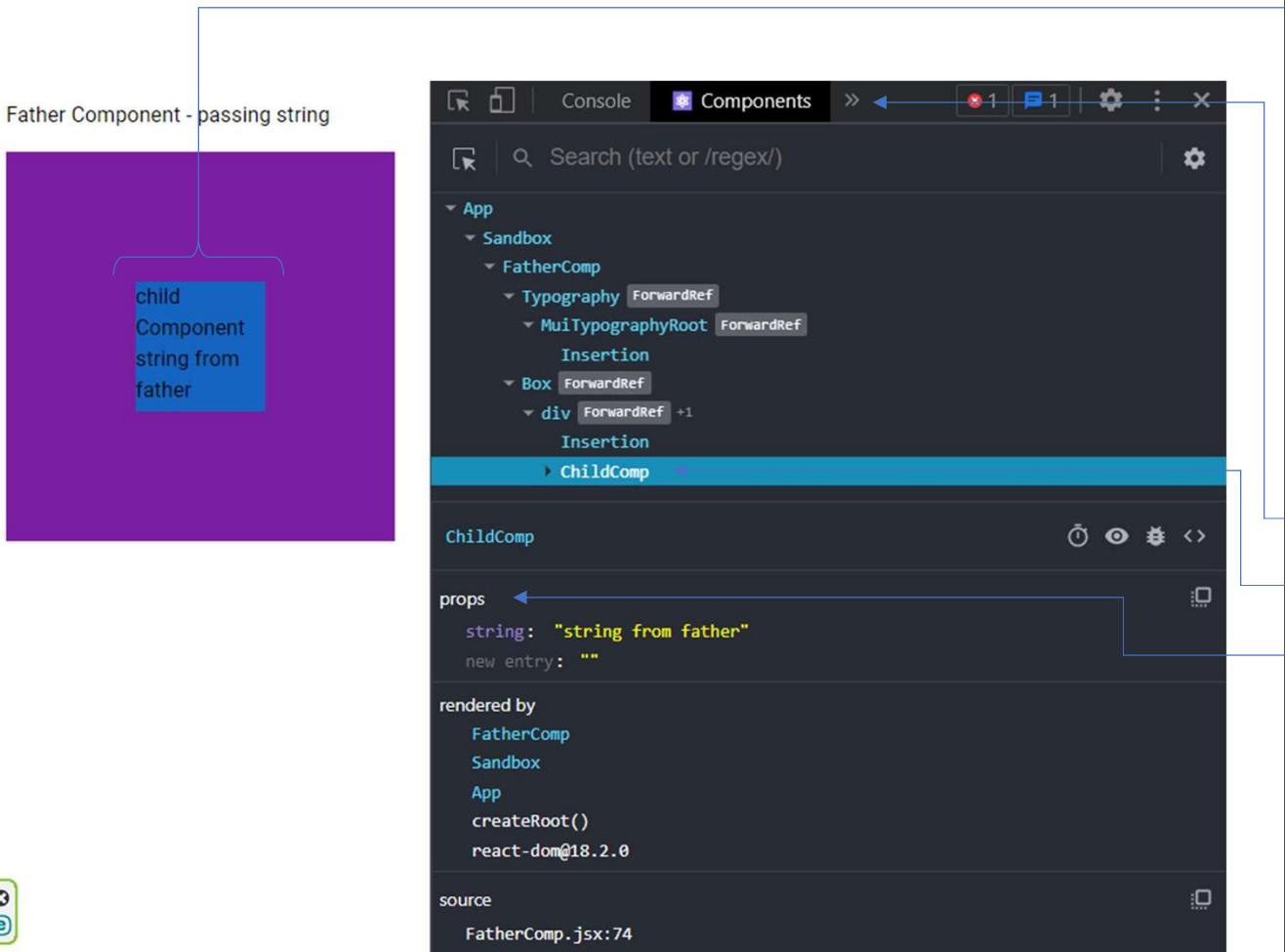
```
❖ FatherComp.jsx U X
client > src > sandbox > props > ❖ FatherComp.jsx > ...
1 import { Box, Typography } from "@mui/material";
2 import React from "react";
3 import ChildComp from "./ChildComp";
4
5 const FatherComp = () => {
6   const string = "string from father"; ←
7
8   return (
9     <>
10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

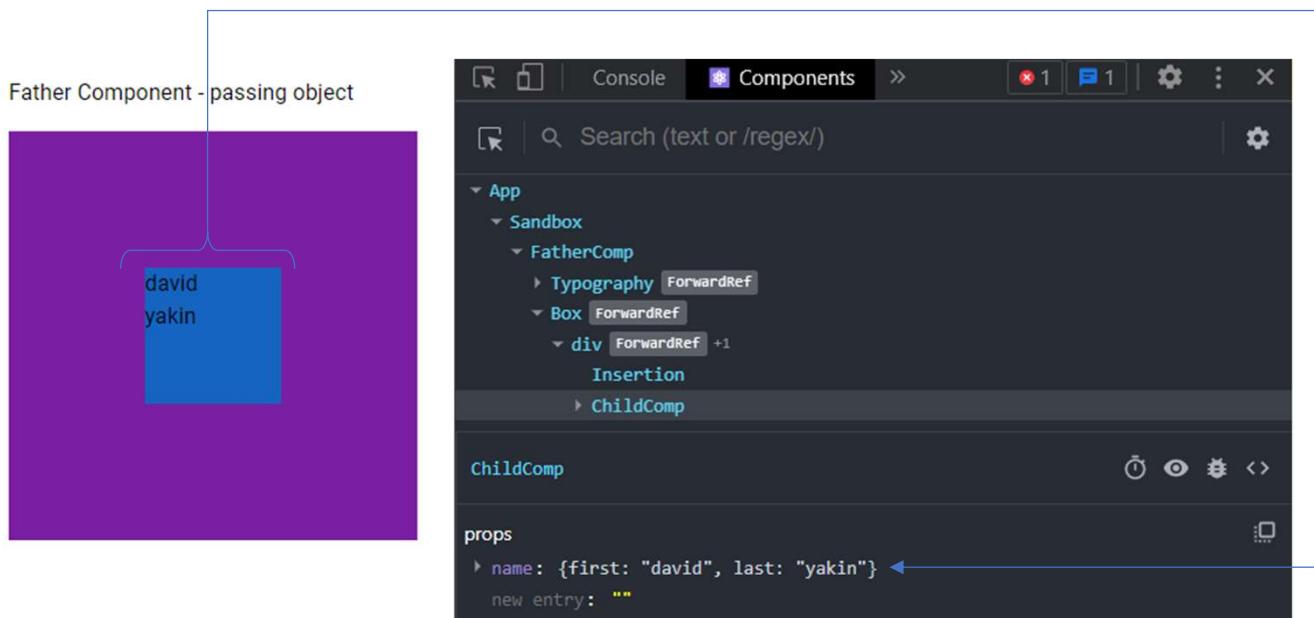
```
31 const FatherComp = () => {
32   const name = { first: "david", last: "yakin" }; ←
33
34   return (
35     <>
36       <Typography variant="body1" m={2}>
37         {" "}
38         Father Component - passing object
39       </Typography>
40       <Box
41         sx={{
42           m: 2,
43   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

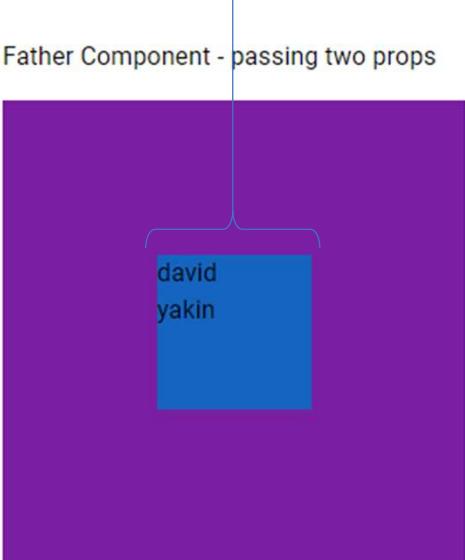
Father Component

- 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



A screenshot of the React DevTools Components tab. The tree structure shows the following components and their props:

- App**:
 - Sandbox**:
 - FatherComp**:
 - Typography** [ForwardRef]
 - Box** [ForwardRef]
 - div** [ForwardRef] +1
 - Insertion**
 - ChildComp**

The **ChildComp** component is selected. In the bottom pane, the **props** section shows:

```
props
  first: "david"
  last: "yakin"
  new entry: ""
```

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;
```

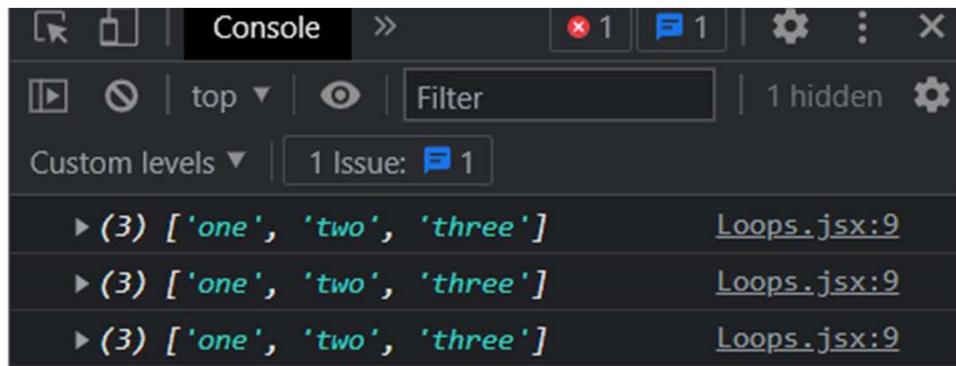
- ייצרנו קבוע בשם `arrayOfString` והשווינו את ערכו לערך של מחרוזות תווים
- בחלק המיועד לו-HTML
 - אנחנו מבצעים לולאה על הקבוע `arrayOfString` באמצעות METHOD `map` שמקבלת עד שלושה פרמטרים:
 - Item – האיבר במערך
 - Index – מספר האינדקס של האיבר במערך
 - array – המערך שעליו נערךת האיטרציה
 - בתוך הלולאה 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.