

תוכן עניינים

[מבוא 3](#_Toc486758018)

[1. יסודות 4](#_Toc486758018)

[1.1. מבנה האתר 4](#_Toc486758019)

[1.2. תרגילים 5](#_Toc486758019)

[2. תקשורת אבא לבן 6](#_Toc486758018)

[2.1. העברת נתונים בשימוש עם props 6](#_Toc486758019)

[2.1. תרגילים 6](#_Toc486758019)

[3. שימוש ב-state 7](#_Toc486758018)

[3.1 העברת נתונים בשימוש עם state 7](#_Toc486758019)

[3.2. תרגילים 9](#_Toc486758019)

[4. הפרדה לתת-רכיבים 10](#_Toc486758018)

[5. מחזור החיים של רכיב ושימוש באירועים 11](#_Toc486758018)

[5.1 שימוש ב componentWillMount 11](#_Toc486758019)

[5.2. תרגילים 14](#_Toc486758019)

[6. תקשורת בן-אבא באמצעות אירועים 15](#_Toc486758018)

[6.1 קשירת פונקציות בין קומפונטות 16](#_Toc486758019)

[6.2. תרגילים 18](#_Toc486758019)

[7. מתודת shouldComponentUpdate 19](#_Toc486758018)

[7.2. תרגילים 19](#_Toc486758019)

[8. תקשורת עם שרת ובקשת GET 20](#_Toc486758018)

[8.1. תרגילים 23](#_Toc486758019)

[9. שימוש בROUTER 24](#_Toc486758018)

[9.2. תרגילים 25](#_Toc486758019)

מבוא

חוברת זו עוקבת אחר התהליך של בניית אתר באמצעות REACT.

מטרתה לשמש כחומר עזר לדוגמאות, ולהמחיש את הנושאים מהמצגת.

בכל הדוגמאות הוסרה תיקיית Node modules כדי להקל על המשקל.   
יש לבצע התקנהnpm install לפני הרצה.

הפרויקט שלנו הוא אתר של חברת התיירות "ברייס טורס". בואו נתחיל!

1. יסודות

נתחיל בפתיחת פרויקט ריאקט חדש באמצעות הפקודה NPX CREATE REACT APP ונקרא לפרויקט שלנו BRYCETOURS

<https://github.com/facebook/create-react-app>

לאחר התקנת node modules נתחיל להוסיף 3 קומפוננטות לאתר

|  |
| --- |
| HEADER |
| MAIN |
| FOOTER |

נכתוב בכל קומפוננט את השם שלו בH1 ונריץ.

נסו לייצר קומפוננטות בצורת הכתיבה של ES5 וES6.   
בפרויקט שלנו FOOTER כתוב בES5 והשאר בES6.

import React, { Component } from 'react';

import Header from './components/Header';

import Main from './components/Main';

import { Footer } from './components/Footer';

class App extends Component {

render() {

return (

<div>

<Header />

<Main />

<Footer />

</div>

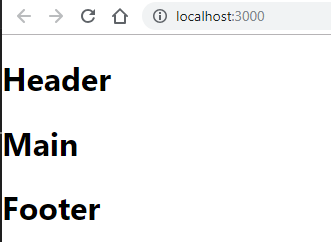
);

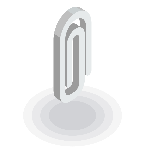
}

}

export default App;

נריץ באמצעות NPM START ונבדוק





מצורפת דוגמה 01.zip

****

**תרגילים**

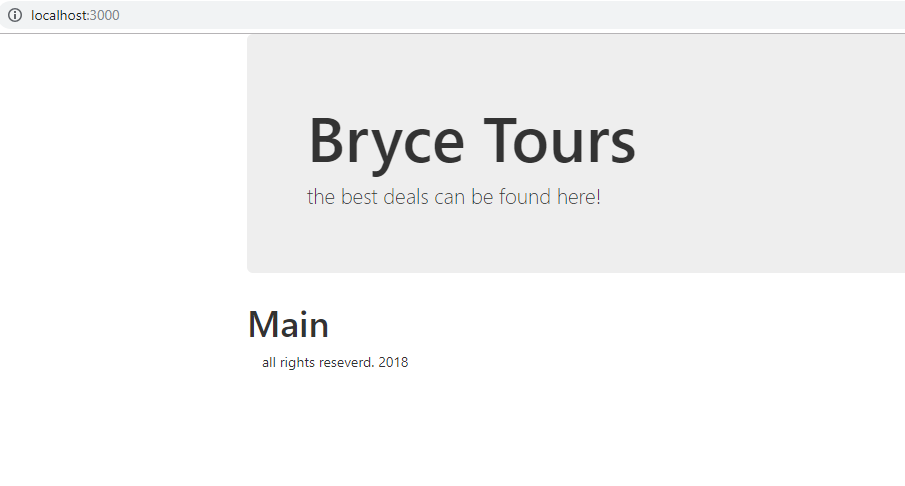
1. הוסיפו קומפוננט sidebar שיוצג מימין לאתר.
2. הוסיפו CSS של bootstrap בעמוד index והשתמשו בקלאסים שלו כדי לייצר אתר שנראה טוב יותר
3. בצעו אכיפה לסוגי המשתנים באמצעות PROPTYPES כדי להבטיח שמתקבלים string , מספר וכו'...

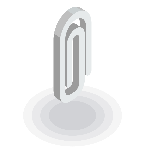
2. תקשורת בין רכיבים – אבא לבן

האתר שלנו כבר מכיל קומפוננט ראשי (APP) שמארח 3 קומפוננטות. נרצה להעביר כמה תכנים מהאבא לילדים, כדוגמת שם האתר לHEADER וה-copyrights לFOOTER

<Header title="Bryce Tours" className="row" />

<Footer copy="all rights reseverd. 2018" className="row" />





מצורפת דוגמה 02.zip

****

**שימו לב**, התווסף בדף האינדקס CSS של bootstrap לנראות טובה יותר

****

**שימו לב**, לצורת הכתיבה בקלאס של ES6 לעומת פונקציה בES5

****

**תרגילים**

1. העבירו משתנים מורכבים יותר כדוגמת מערך או אובייקט עם כמה ערכים בתוכו דרך PROPS

והציגו אותם

3. שימוש ב-State

בהמשך לתרגיל בדוגמה קודמת, בדוגמה זו העברנו את פרטי התפריט כמערך ואת תמונת הרקע כאובייקט.

לכל קומפוננט יש STATE משלו וכעת באתר נרצה להציג שעון עצר עבור המבצעים,  
דבר שימשוך את הקונה לבצע הזמנה כי בעוד כמה דקות המצבע יתחלף.   
props לא ניתנים לשינוי אך המשתנים ב- state כן.

class App extends Component {

constructor(props) {

super(props);

this.state = {

deals: [

{

to: "London",

endsIn: 95,

image: "https://media.gettyimages.com/photos/houses-of-parliament-at-night-westminster-london-uk-picture-id508151164"

},

{

to: "Paris",

endsIn: 60,

image: "https://d39gusjpdm7p1o.cloudfront.net/data/layout\_grouping/static\_page\_step/20784/a330628091ede7eb1548d6cda58e0357.jpg?ver=1477297804"

},

{

to: "Barcelona",

endsIn: 78,

image: "https://images.lucasfox.com/location/4x3\_960w/836D68DCBD.jpg"

}

]

};

}

render() {

let image = {

title: "bryceTours",

src: "https://img.maximummedia.ie/her\_ie/eyJkYXRhIjoie1widXJsXCI6XCJodHRwOlxcXC9cXFwvbWVkaWEtaGVyLm1heGltdW1tZWRpYS5pZS5zMy5hbWF6b25hd3MuY29tXFxcL3dwLWNvbnRlbnRcXFwvdXBsb2Fkc1xcXC8yMDE3XFxcLzA2XFxcLzI2MTU1MTI0XFxcL2lTdG9jay0xNTU0MzkzMTUxLmpwZ1wiLFwid2lkdGhcIjo3NjcsXCJoZWlnaHRcIjo0MzEsXCJkZWZhdWx0XCI6XCJodHRwczpcXFwvXFxcL3d3dy5oZXIuaWVcXFwvYXNzZXRzXFxcL2ltYWdlc1xcXC9oZXJcXFwvbm8taW1hZ2UucG5nP3Y9NVwifSIsImhhc2giOiI0YTFiNWEzOGQyMGY0OTUzODVjYTdmYmM4NjVlMTRiNGUwMjZhNmQ1In0=/istock-1554393151.jpg"

}

return (

<div className="container">

<Header title="Bryce Tours" className="row" />

<Main mainImage={image} className="row" deals={this.state.deals} />

<Footer copy="all rights reserved. 2018" className="row" />

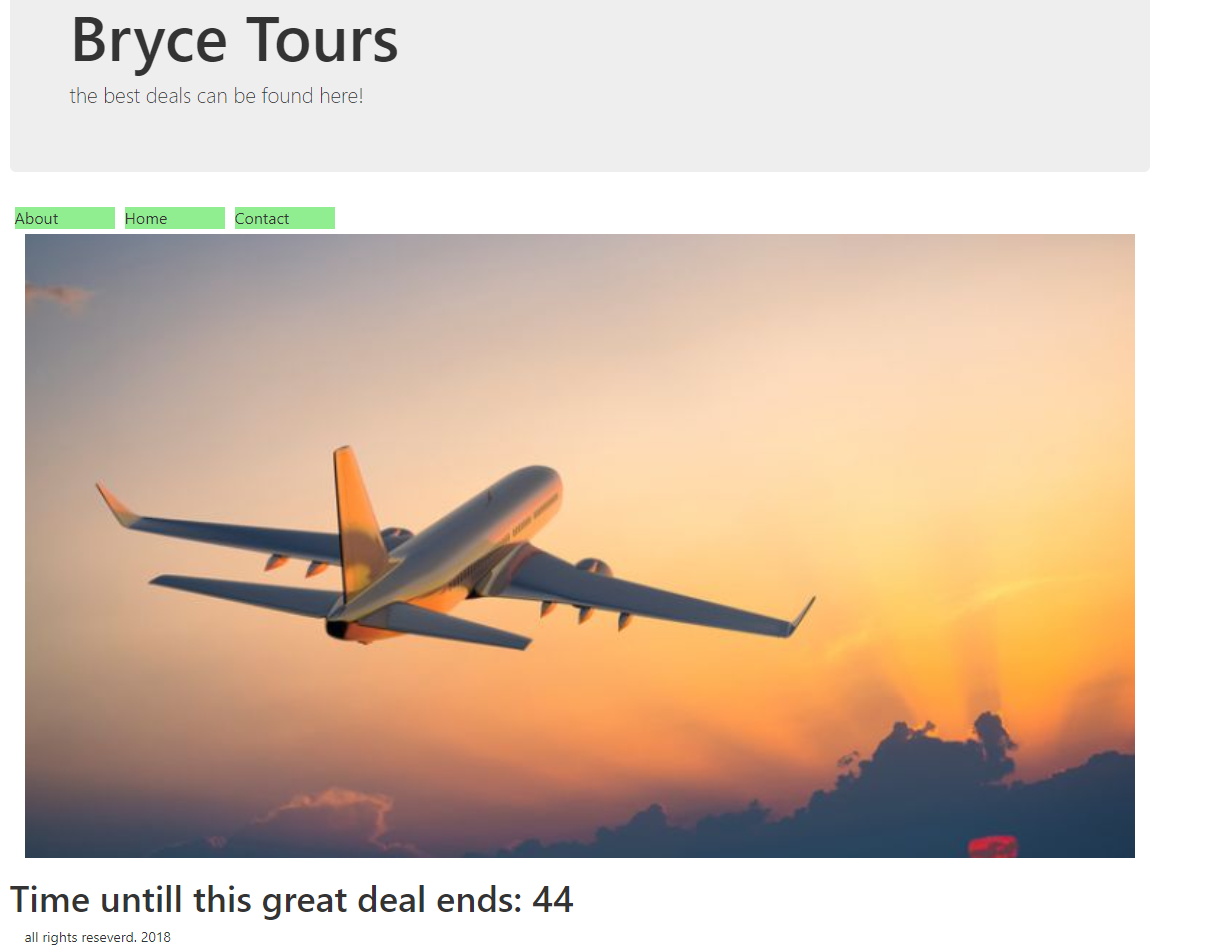
</div>

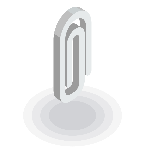
);

}

}

export default App;





מצורפת דוגמה 03.zip

****

**תרגילים**

1. צרו DEAL קומפוננט, שיציג מבצע. לכל מבצע זמן תפוגה שונה, תמונה ויעד שונים (לדוגמה- פריז, לונדון...) הציגו מערך המבצעים שיימסר ל-MAIN מהאבא.

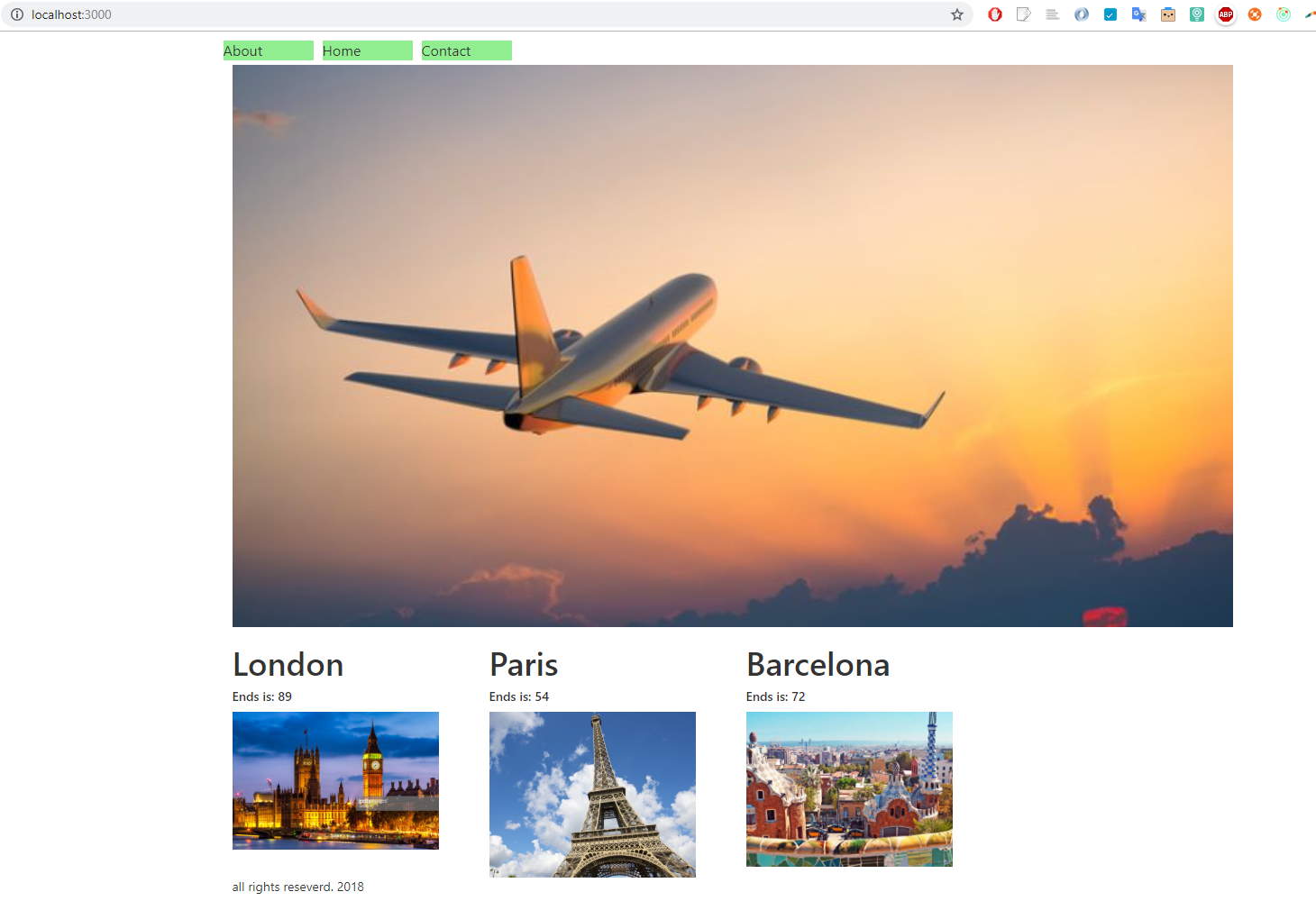
4. הפרדה לתת-רכיבים

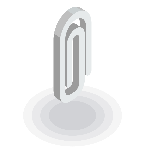
בדוגמה זו נפתור את התרגיל מהדוגמה הקודמת ונוסיף מערך של DEALS ושעון עצר בכל קומפוננט.

<div className="row">

{this.props.deals.map((deal) => <Deal deal={deal} />)}

</div>

התוצאה צריכה להיראות בערך כך:  




מצורפת דוגמה 04.zip

5. מחזור החיים של הרכיב ואירועים

לכל קומפוננט יש בנאי אך גם יש לו מתודות של מחזור חיים.

נעדיף להשתמש בהן כדי לוודא שהטיימר מתחיל לרוץ לאחר הציור של הקומפוננט (ולא לפני), ומפסיק לרוץ בירידה של הקומפוננט מהמסך.

כמו כן, נרצה להוסיף כפתור של BUY DEAL שיעצור את שעון העצר.

class Deal extends Component {

timerInerval = null;

constructor(props) {

super(props);

this.state = { timer: this.props.deal.endsIn };

}

componentDidMount() {

this.startCountDown();

}

render() {

return (

<div className="col-sm-3 deal">

<h1> {this.props.deal.to}</h1>

<h5>Ends is: {this.state.timer}</h5>

<img src={this.props.deal.image} />

<button onClick={this.stopTimer.bind(this)} className="btn btn-primary">Buy deal</button>

</div>

);

}

componentWillUnmount() {

this.stopTimer();

}

startCountDown() {

this.timerInerval = setInterval(() => {

if (this.state.timer > 0) {

let updatedTimer = this.state.timer;

updatedTimer = updatedTimer - 1;

this.setState({ timer: updatedTimer });

}

}, 1000)

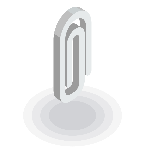
}

stopTimer() {

clearInterval(this.timerInerval);

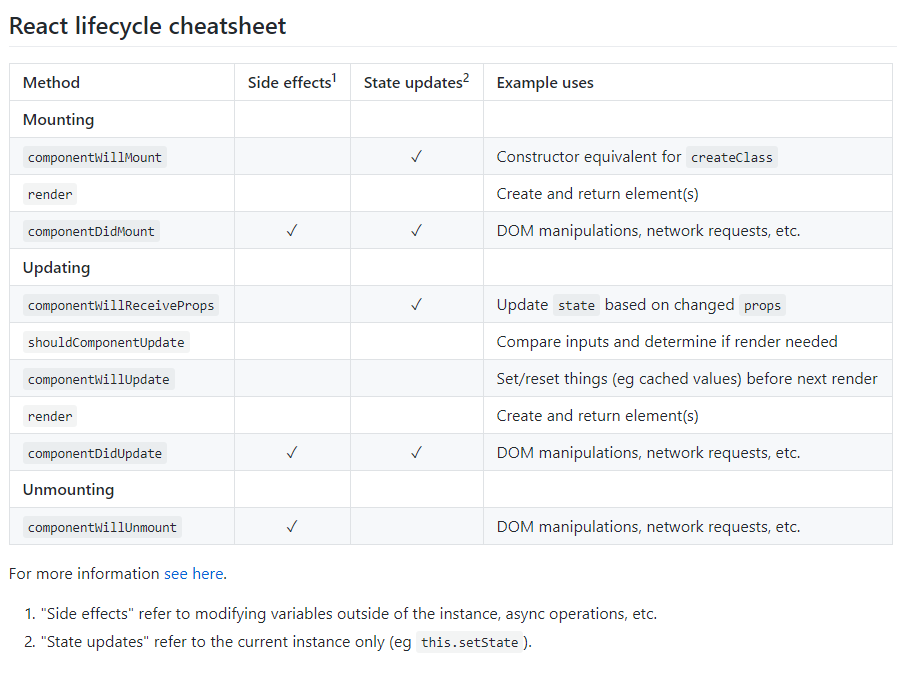
}

}



מצורפת דוגמה 05.zip

ברשימה המצורפת בעמוד הבא מתוארות כל מתודות מחזור החיים.



בגרסה 16 של ריאקט התווספו מתודות נוספות ונדרסו מתודות קיימות על מנת להוציא אותן משימוש.

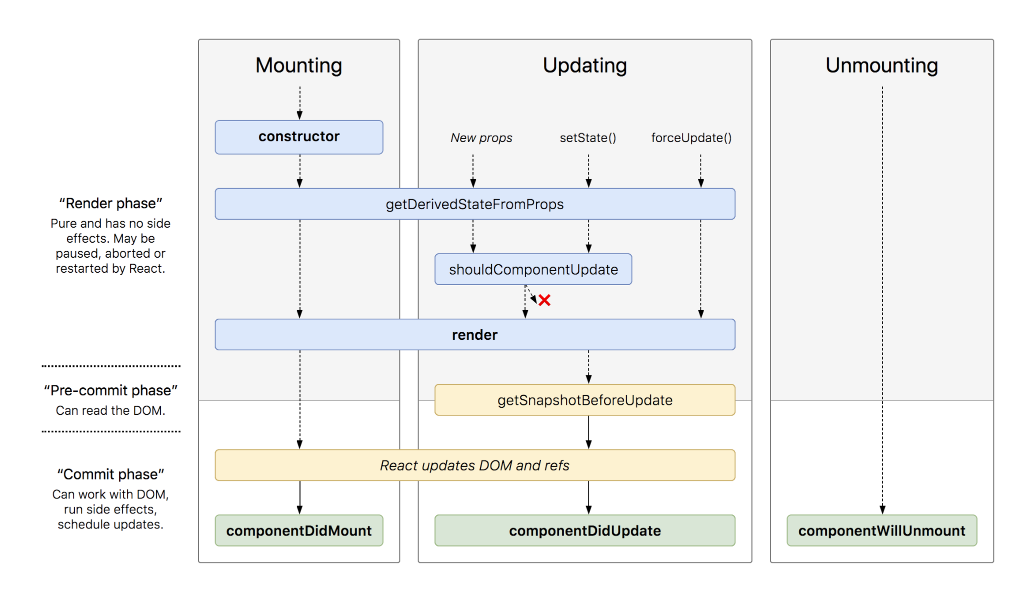
הסיבה העיקרית לדריסה היא שימוש לא נכון של מפתחים בעיקר בזמן קריאות API.

כדי ליישר קו ולעבוד טוב יותר בצורה אסינכרונית, שונו השמות והתווספו מתודות חדשות

איך נדע במה מותר להשתמש ומה נדרס? המתודות שנדרסו סומנו כunsafe בתחילתן

נדרסו:

componentWillMount, componentWillReceiveProps , componentWillUpdate  
  
נוספו:

getDerivedStateFromProps, getSnapshotBeforeUpdate , getDerivedStateFromError  
  
בציור הבא נוכל לאות את השימוש במתודות החדשות והחלפת הישנות בהן  
  


* נכון לכתיבת שורות אלו (גרסה 1,6.6) אלו המתודות שנוספו התכונית היא בגרסה 17 להוציא משימוש לגמרי את המתודות שסומנו כUNSAFE.
* ריאקט היא טכנולוגיה שמתעדכנת כל הזמן וחשוב לעקוב גם בעתיד אחר השינויים בין הגרסאות בדף: <https://reactjs.org/versions/>

****

**תרגילים**

1. הוסיפו כפתור הסתר/ הצג לכל מבצע שיאפשר למשתמש להסתיר מבצעים שהוא לא מעוניין בהם.
2. הוסיפו צ'קבוקס FOLLOW שיתחיל לא לחוץ ושבאמצעותו המשתמש יוכל לסמן חופשה למעקב.
3. הוסיפו את כל מתודות מחזור החיים לקומפוננט לבחירתכם ותעדו אותן בקוד בשימוש בconsole.log תוכלו לראות לאחר מכן את הסדר שלהן ומתי להשתמש בהן.

6. תקשורת בן-אבא באמצעות אירועים

בריאקט, נוכל להפעיל פונקציות אצל האבא באמצעות הילד. נשתמש בכפתור Buy now על מנת להגדיל את כמות הרוכשים אצל האבא.

נהוג לנהל את הSTATE בקומפוננט העליון ביותר כדי להימנע ממגוון מקומות בהם המידע משתנה (דבר זה נוח יותר לתחזוקה).

נשנה קצת את מבנה הנתונים ונוסיף לכל עסקה האם נרכשה או לא.

כמו כן קיבלנו אזהרה בשימוש בMAP שלא הכרזנו על ID (לא בוצע שימוש בKEY) ייחודי, ולכן כדאי להוסיף למבנה הנתונים ID ואותו לקשור במיפוי פריטים רבים לפונקציה MAP ובכך להעלים את האזהרה שהופיעה בconsole.

נקשור פונקציה מAPP שתעבור לMAIN (הבן שלו) מתוך MAIN נקשור פונקציה שתופעל בDEAL (הבן של MAIN והנכד של APP). אח"כ כדי לשנות את הערך בAPP נצטרך "לטפס" שוב מDEAL ל MAIN ול APP בהתאמה.

מי שישנה את הSTATE בפועל יהיה APP וכל שינוי בSTATE יגרום לעדכון.

class App extends Component {

constructor(props) {

super(props);

this.state = {

deals: [

{

id:1,

to: "London",

endsIn: 95,

image: "https://media.gettyimages.com/photos/houses-of-parliament-at-night-westminster-london-uk-picture-id508151164"

,numOfBuyers:0

},

{

id:2,

to: "Paris",

endsIn: 60,

image: "https://d39gusjpdm7p1o.cloudfront.net/data/layout\_grouping/static\_page\_step/20784/a330628091ede7eb1548d6cda58e0357.jpg?ver=1477297804"

,numOfBuyers:0

},

{

id:3,

to: "Barcelona",

endsIn: 78,

image: "https://images.lucasfox.com/location/4x3\_960w/836D68DCBD.jpg"

,numOfBuyers:0

}

]

};

}

render() {

let image = {

title: "bryceTours",

src: "https://img.maximummedia.ie/her\_ie/eyJkYXRhIjoie1widXJsXCI6XCJodHRwOlxcXC9cXFwvbWVkaWEtaGVyLm1heGltdW1tZWRpYS5pZS5zMy5hbWF6b25hd3MuY29tXFxcL3dwLWNvbnRlbnRcXFwvdXBsb2Fkc1xcXC8yMDE3XFxcLzA2XFxcLzI2MTU1MTI0XFxcL2lTdG9jay0xNTU0MzkzMTUxLmpwZ1wiLFwid2lkdGhcIjo3NjcsXCJoZWlnaHRcIjo0MzEsXCJkZWZhdWx0XCI6XCJodHRwczpcXFwvXFxcL3d3dy5oZXIuaWVcXFwvYXNzZXRzXFxcL2ltYWdlc1xcXC9oZXJcXFwvbm8taW1hZ2UucG5nP3Y9NVwifSIsImhhc2giOiI0YTFiNWEzOGQyMGY0OTUzODVjYTdmYmM4NjVlMTRiNGUwMjZhNmQ1In0=/istock-1554393151.jpg"

}

return (

<div className="container">

<Header title="Bryce Tours" className="row" />

<Main buyDeal={this.addBuyer.bind(this)} mainImage={image} className="row" deals={this.state.deals} />

<Footer copy="all rights reseverd. 2018" className="row" />

</div>

);

}

addBuyer(dealID)

{

//copy array and change

let newDeals= this.state.deals.slice();

newDeals[dealID-1].numOfBuyers++;

this.setState({

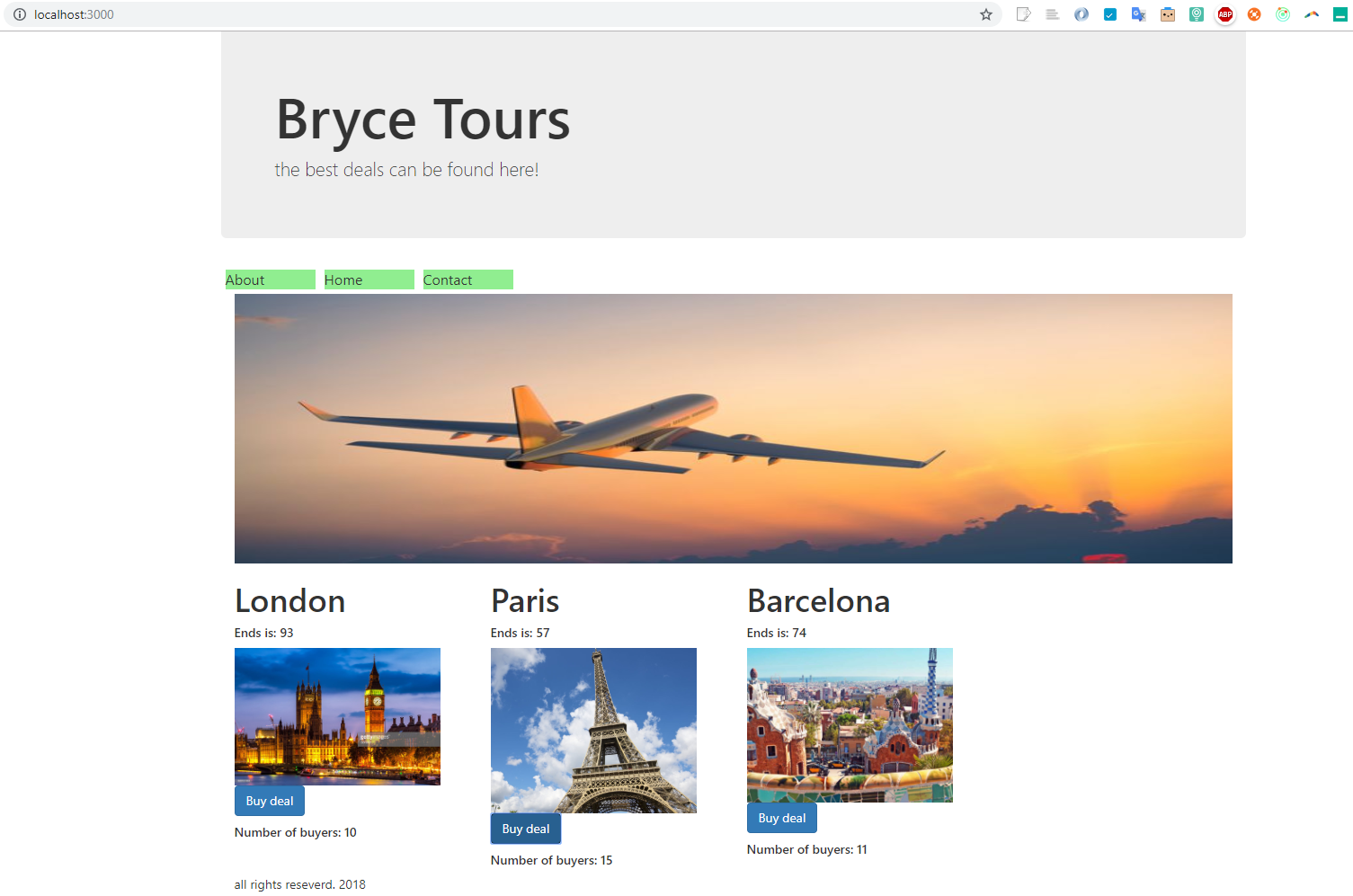
deals:newDeals

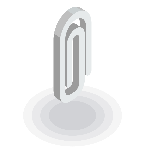
})

}

}

export default App;





מצורפת דוגמה 06.zip

****

**תרגילים**

1. הוסיפו לכל מבצע כמות מקסימלית לרכישה. כשכמות הרוכשים תתמלא הכפתור יופיע כלא פעיל ולא יתווספו רוכשים.
2. הוסיפו תיבת טקסט בראש האתר שתשמור את שם הרוכש ושמרו עבור כל רוכש גם את שמו.

7. מתודת shouldComponentUpdate

את הגבלת כמות הרוכשים ניתן גם למנוע באמצעות מחזור החיים. לדוגמה נחליט שלכל דיל יש כמות מקסימלית של 10 רוכשים.

הקוד יראה כך:

shouldComponentUpdate(nextProps, nextState)

{

if(nextProps.deal.numOfBuyers>10)

{

return false;

}

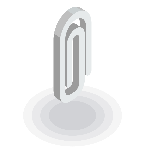
else

{

return true;

}

}



מצורפת דוגמה 07.zip

****

**תרגילים**

1. צרו עוד 2 קומפוננטות ABOUT וCONTACT

קשרו אותן שיופיעו וייעלמו בהתאם ללחיצה על התפריט כשאר HOME מציג את MAIN

8. תקשורת עם שרת ובקשת GET

בעקבות תרגיל קודם השגנו סוג של ניווט מזויף באתר.

מזויף כיוון שהכתובת לא משתנה וכך לא נוכל לשלוח לחבר קישור פנימי לאתר אלא רק לדף הבית.

כמו כן כעת המבצעים הינם סטטיים.

המבצעים נטענים מתוך האתר עצמו ויש צורך בתקשורת עם שרת מרוחק שמחובר לDB.

על מנת לדמות שרת נשתמש בJSON SERVER, זהו פרויקט צד שרת בNODE שנריץ אותו באמצעות NPM START ונכוון אותו לקובץ JSON שישמש כDB.

ניתן לקרוא בהרחבה על ההתקנה וקישור לקובץ DB כאן:

<https://github.com/typicode/json-server>

וכאן:

<https://www.npmjs.com/package/json-server>

קובץ הדטבייס מצורף בדוגמה 8 וכעל אלינו למשוך את הנתונים באמצעות קריאת FETCH.

class App extends Component {

constructor(props) {

super(props);

this.state = {

page: "Home",

deals: []

};

}

componentDidMount() {

fetch('http://localhost:3000/deals')

.then(response => response.json())

.then(data => this.setState({ deals: data }));

}

render() {

let image = {

title: "bryceTours",

src: "https://img.maximummedia.ie/her\_ie/eyJkYXRhIjoie1widXJsXCI6XCJodHRwOlxcXC9cXFwvbWVkaWEtaGVyLm1heGltdW1tZWRpYS5pZS5zMy5hbWF6b25hd3MuY29tXFxcL3dwLWNvbnRlbnRcXFwvdXBsb2Fkc1xcXC8yMDE3XFxcLzA2XFxcLzI2MTU1MTI0XFxcL2lTdG9jay0xNTU0MzkzMTUxLmpwZ1wiLFwid2lkdGhcIjo3NjcsXCJoZWlnaHRcIjo0MzEsXCJkZWZhdWx0XCI6XCJodHRwczpcXFwvXFxcL3d3dy5oZXIuaWVcXFwvYXNzZXRzXFxcL2ltYWdlc1xcXC9oZXJcXFwvbm8taW1hZ2UucG5nP3Y9NVwifSIsImhhc2giOiI0YTFiNWEzOGQyMGY0OTUzODVjYTdmYmM4NjVlMTRiNGUwMjZhNmQ1In0=/istock-1554393151.jpg"

}

let pageToDraw;

if (this.state.page == "Home") {

pageToDraw = <Main changePage={this.changePage.bind(this)} buyDeal={this.addBuyer.bind(this)} mainImage={image} className="row" deals={this.state.deals} />

}

else if (this.state.page == "About") {

pageToDraw = <About ></About>

}

else {

pageToDraw = <Contact ></Contact>

}

return (

<div className="container">

<Header title="Bryce Tours" className="row" />

<Menu changePage={this.changePage.bind(this)} listOfItems={["About", "Home", "Contact"]} ></Menu>

<div className="row">

{pageToDraw}

</div>

<Footer copy="all rights reseverd. 2018" className="row" />

</div>

);

}

addBuyer(dealID) {

//copy array and change

let newDeals = this.state.deals.slice();

newDeals[dealID - 1].numOfBuyers++;

this.setState({

deals: newDeals

})

}

changePage(page) {

this.setState({

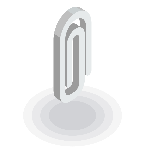
page: page

});

}

}

export default App;

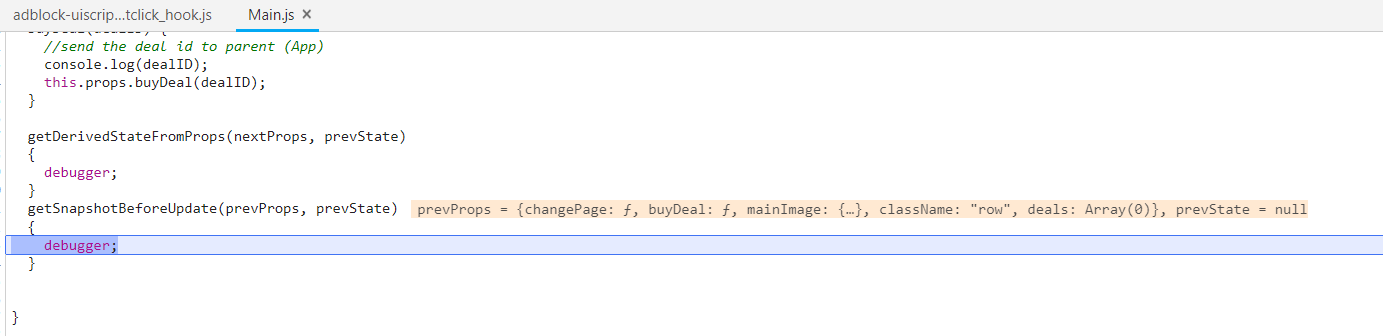


מצורפת דוגמה 08.zip

**שימוש במתודות החדשות של מחזור החיים והעברת נתונים בין אבא ובן בצורה אסינכרונית**

שימו לב להטמעה של המתודות החדשות בריקאט 16.3+

APP הוא אבא של MAIN והוא התחיל עם מערך ריק שאותו העביר לבנו. כמו כן, יצר בקשה אסינכרונית לשרת והתשובה חזרה לאחר מספר שניות. במתודות החדשות נוכל להתחבר לרגע לפני העדכון, כדי לבצע שמירה של המצב הקודם וגם לקבל את העדכון עצמו בו נמצא PROPS חדשים



****

**תרגילים**

1. בקומפוננט הCONTACT הכניסו טופס צור קשר שישלח בPOST לשרת.
2. הוסיפו דף הוספת מבצעים שיאפשר להוסיף לשרת ב POST מבצעים חדשים.

9. שימוש בROUTER

בבואנו להוסיף ניווטים לאתר נצטרך להתקין את ספריית הניווט

npm i react-router-dom –save

נסיר את כל הניווטים בלחיצה שהיו לנו ונשנה את הקוד שישתמש ברכיבים מתוך הספריה החדשה שייבאנו.

הרכבים שבמשימוש הינם תגית ROUTER שתעטוף הכל, ובתוכה קומפוננטות שנרצה שיישארו לעומת כאלו שנרצה שיושפעו מכתובת הURL

<Router>

<div className="container">

<Header title="Bryce Tours" className="row" />

<Menu changePage={this.changePage.bind(this)} listOfItems={["About", "Home", "Contact"]} ></Menu>

<div className="row">

<Route exact path="" render={() => (<Redirect to="/home" />)} />

<Route path="/about" component={About} />

<Route path="/contact" component={Contact} />

<Route path="/home" render={() => (<Main changePage={this.changePage.bind(this)} buyDeal={this.addBuyer.bind(this)} mainImage={image} className="row" deals={this.state.deals} />)} />

</div>

<Footer copy="all rights reseverd. 2018" className="row" />

</div>

</Router>

על מנת לשמור על סדר נוכל גם להוציא את הניווטים למערך חיצוני ובכך להוסיף עוד ועוד ניווטים.

const routes = [

{

path: '',

component: Redirect,

data: () => (<Redirect to="/home" />)

},

{

path: '/about',

component: About

},

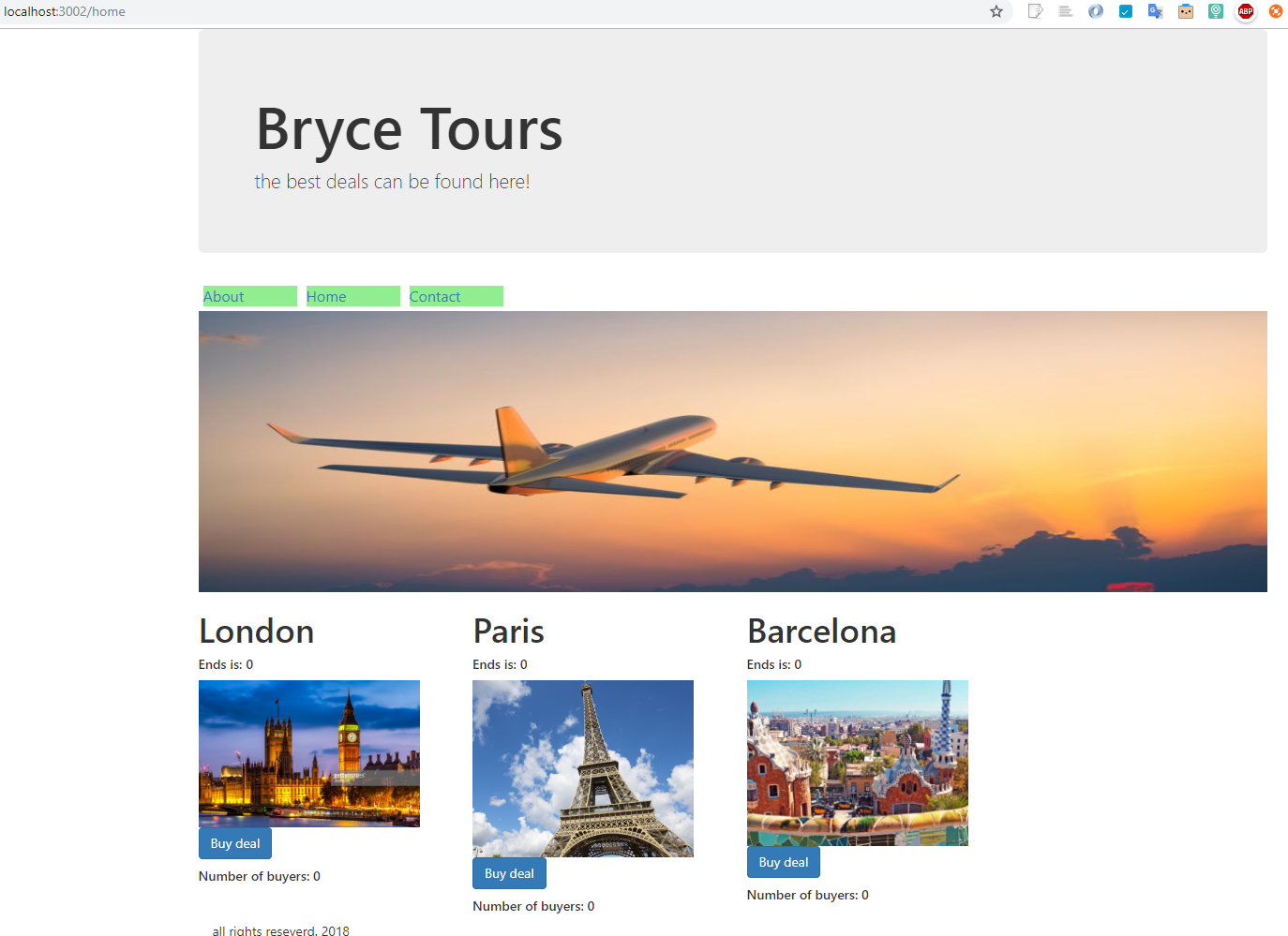
{

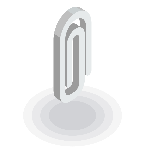
path: '/contact',

component: Contact

}

]





מצורפת דוגמה 09.zip

****

**תרגילים**

1. קראו את הדוקומנטציה בREACT ROUTER והוסיפו עמוד 404 שיגיעו אליו כשמקלידים כתובת שלא קיימת
2. אפשרו ניוון למבצע ספציפי כשהURL ייראה כך: deals/1 ינווט למבצע 1 וכו'...
3. בעבר היה פעיל דף המבצעים בכתובת brycetours.com/sales , הדף הוסר אך עדיין קיים במיילים שיצאו ללקוחות... טפלו בניווט הישן שיבצע REDIRECT לדף הבית במקום המבצעים.

**בהצלחה**!