שיעור 15

Hooks

ריאקט מספקת לנו מספר פונקציות בנויות בשפה שנקראיHooks
:לאט לאט במהלך השיעורים הקרובים נלמד עוד ועוד הוקים, הראשון שנלמד יהיה היום

useState

,אחד ההוקים החשובים, אם לא הכי חשוב, אם בעבר היינו שומרים ערכים שנתונים לשינוי בתוך משתנה
בריאקט את הנתון המשתנה ,נשמור בתוך useState
.עוד מעט נראה דוגמאות על הקוד, אבל יש נושא אחד מאוד חשוב בריאקט
!אסור לנו לדרוס ערכים קודמים
:מס׳ דוגמאות של דריסות

```
const array = [1,2,3]
array.push(4) // התוכן של המשתנה השתנה

const object = {name:"eitan",age:26}
object.car = 'Renault' // object השתנה
```

Immutable VS Mutable code

אחד הפיצ׳רים של שימוש בuseState
.הוא שמתי שיש שינוי במשתנה, ריאקט מרנדר מחדש את העמוד, ותמיד יציג לנו את הערך העדכני ביותר בזיכרון
הצורה שבה ריאקט עושה את זה, הוא השוואה. ריאקט עושה השוואה של הערך הקודם שהיה במשתנה לעומת הערך שיש כרגע במשתנה
:לדוגמא יצירת מערך

```
const [array,setArray] = useState([1,2,3])
```

הערך הראשון, זה המשתנה שמחזיק את המערך של הספרות 1,2,3 , והערך השני בשם setArray
.הוא הפונקציה שיכולה לערוך את הסטייט
) הבעיה היא שבמידה ואנחנו כותבים קוד דורס Mutable code).אז ריאקט לא יכול לעשות השוואה בין הערך הקודם לקיים
:דוגמא

```
const array1 = [1,2,3]
const array2 = array1
array2.push(4)
//array1 == [1,2,3,4] וגם array2 == [1,2,3,4]
```

Immutable VS Mutable code

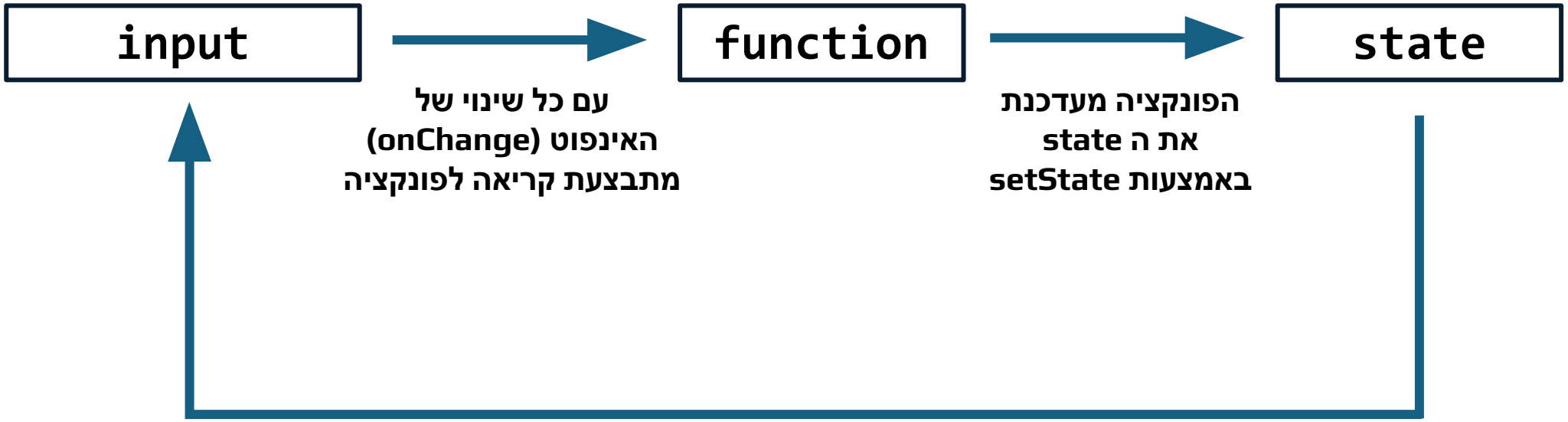
זה קורה בגלל שמערכים, אובייקטים ופונקציות נשמרים על ידי כתובת בזיכרון.
המשתנה שמחזיק את המערך לדוגמא, לא מחזיק את הערך של המערך אלא את הכתובת זיכרון איפה מוחזק הערך.
משתנים שמחזיקים ערכים כמו מחרוזות, מספרים, בוליאנים, נשמרים על ידי הערך שלהם.

ריאקט חייב לעשות השוואה בין הערך הקודם, לבין הערך העדכני, במידה והיה שינוי אז הוא מרנדר מחדש את הרכיב, במידה ולא היה שינוי, הוא לא ירנדר מחדש.
במידה ודרסנו את המערך, לריאקט אין דרך טובה להשוות כי שתיהם יהיו אותו הערך (לפני ואחריי).
לכן אנחנו תמיד מעתיקים! בצורה הבאה:

```
const array1 = [1,2,3]
const array2 = [...array1] // העתקה רדודה
array2.push(4)
//array1 == [1,2,3] array2 == [1,2,3,4]
```

וכך יש לנו שתי ערכים שונים, שריאקט יכול להשוות ביניהם.
נכנס לvs code ביחד ונראה כיצד אנחנו כותבים את זה.

onChange



מתבצעת קריאה ל render( ) שמרענן את הקומפוננטה,
צבע רקע האינפוט יתעדכן בהתאם למה שמוגדר ב state

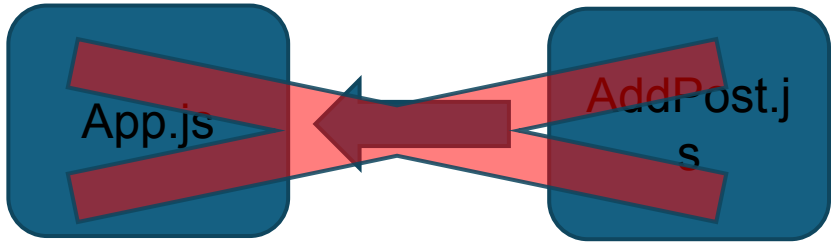
Add Post

מטרה:

הוספת פוסט חדש (קומפוננטת AddPost), כאשר נלחץ על כפתור Add הפוסט החדש יתווסף למערך הפוסטים בApp (מהשרת) ויוצג (באמצעות לולאת map כפי שכבר ראינו).

בעיה:

לא ניתן להעביר ערכים לקומפוננטה הראשית (מ AddPost אל App) אלא רק מ App אל AddPost.

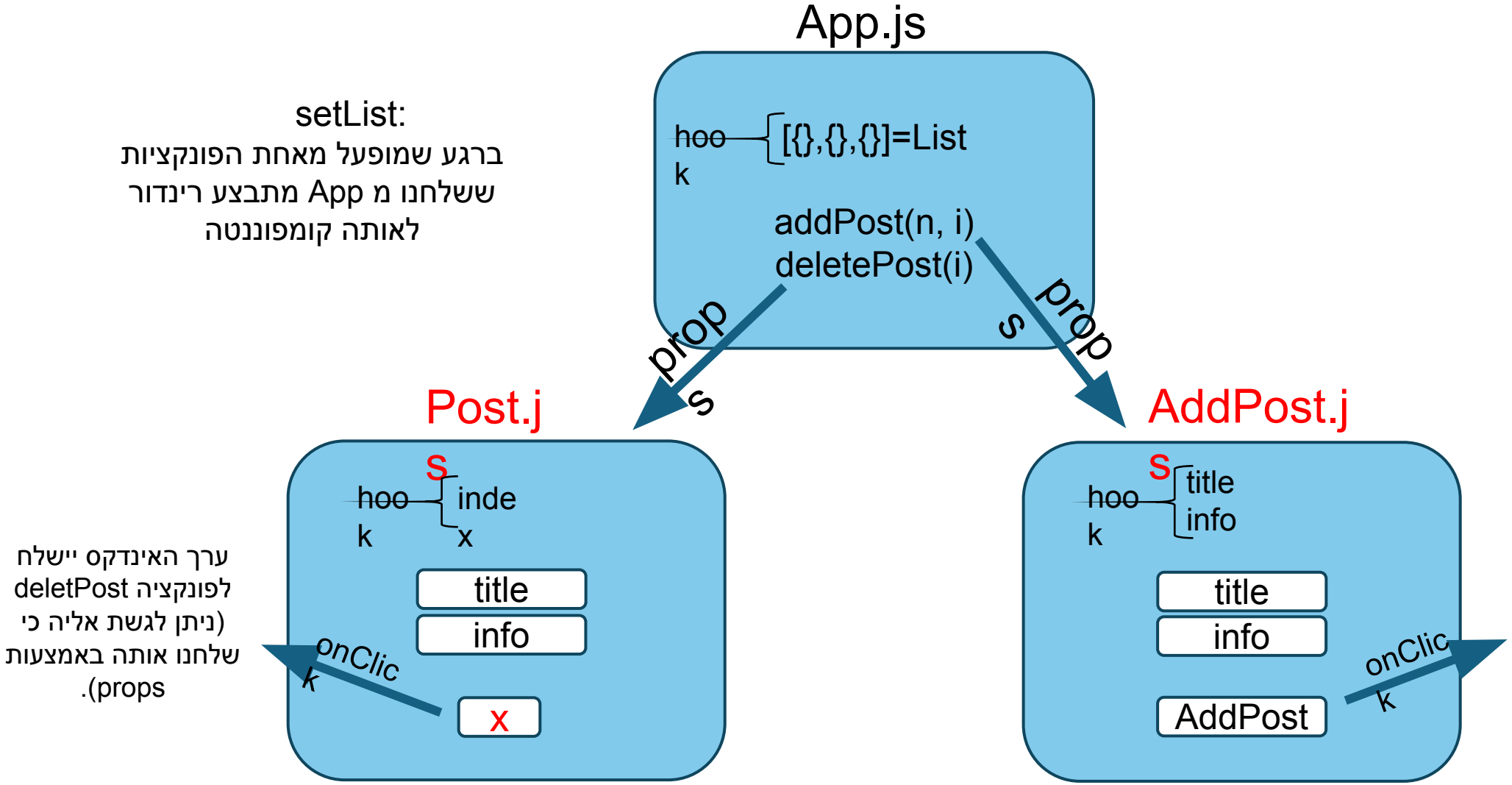


פתרון:

ניצור פונקציה בקומפוננטה הראשית ונשלח אותה לקומפוננטה AddPost בעזרת props.

בואו נראה איך זה עובד...

אז מה בעצם ראינו כאן?



Hooks

דוגמא – יצרנו כפתור שכל פעם שלוחצים עליו הוא משנה את ה state.

App.js

app4 > src > App.js > ...

```
import React,{useState} from 'react';
import './App.css';

function App() {

const [flag, setflag] = useState(false);

  return (
    <div className="App">
      <button onClick={()=>{setflag(!flag)}}>Change Flag</button><br/>
      {flag.toString()}

    </div>
  );
}

export default App;
```

סוף שיעור 15

האם יש שאלות?