React

שיעור שביעי

שליחת פרמטרים בניווט עמודים

- בניווט בין עמודים ניתן גם לשלוח פרמטרים בURL ולקבל אותם בקומפוננט. לדוגמא לאחר עמוד לוגין נעביר את הפרמטר של שם המשתמש לעמוד אליו הועבר ובעמוד זה ניתן לשלום את שמו ולהשתמש ולהציג איפה שצריך.
- בשביל זה בהגדרת ה Routes נצטרך להגדיר שניתוב הקומפוננט מסוגל לקבל פרמטרים ואת כל הפרמטרים שנרצה לשלוח. אם לא נגדיר – לא נוכל לשלוח, אם הגדרנו ולא שלחנו זה בסדר ומגיע NULL.

```
<Route exact path="/about/:userName" element={<About />} />
```

- בקומפוננט בו נרצה לנווט לעמוד ולשלוח את הפרמטרים נגדיר זאת באמצעות Link עם הפעלת הקומפוננט כרגיל, בתוספת של שליחת הפרמטרים – בצורה הבאה

```
<button onClick={() => navigate(`/about/${userName}`)}>About</button>
```

לכל פרמטר נוסיף סלש, $ ובתוך צומדיים את שם הפרמטר אותו נעביר.

- ניתן גם להעביר מספר פרמטרים בצורה הבאה:

```
<Route exact path="/about/:userName/:password" element={<About />} />
```

```
<button onClick={() => navigate(`/about/${userName}/${password}`)}>About</button>
```

React

שימוש בפרמטרים שנשלחו בקומפוננט

- בקומפוננט אליה שלחנו את הנתונים ניתן לקבל אותם באמצעות האובייקט ()useParams המגיע מreact-router-dom בצורה הבאה:
- נייבא את האובייקט מהספריה:

```
import {useParams } from 'react-router-dom';
```

- נקבל את הנתונים שנשלחו ע"י יצירת משתנה בדיוק באותו שם של הפרמטר אותו העברנו לקומפוננט ונשווה לאובייקט ()useParams react-router-dom יודע בצורה הזו לשלוף בדיוק את משתנה שכמו השם שהגדרנו כאן מתוך הפרמטרים שנשלחו

```
const { userName } = useParams();
const {pass} = useParams();
```

נוכל להשתמש בפרמטר זה בכל מקום בו נרצה בקומפוננט לדוגמא:

```
<h2>Hello {userName}</h2>
```

useRef() hook

- React useRef() hook מאפשר לגשת ישירות לאלמנט ה DOM
- useRef מקבל כערך התחלתי ארגומנט ומחזיר reference אליו.
- ל useRef יש רכיב מיוחד הנקרא current ניתן לגשת באמצעותו ישירות לאלמנט אותו הגדרנו כ useRef ולשנותו בכל רגע נתון ע"י שמעדכנים את ערך ה current.
- יצירת אובייקט reference:
  נייבא את useRef מ react :
  ```
  import { useRef } from 'react';
  ```
  נאתחל את אובייקט ה Reference:
  ```
  const refObject = useRef("some value")
  ```
  לאובייקט זה ניתן לגשת על מנת להציג או לשנות רק מתוך אובייקט ה current.
  גישה לאובייקט זה ע"י הcurrent לדוגמא ב return:
  ```
  <h1>Reference object value: {refObject.current}</h1>
  ```
- כדי לשנות את value של אובייקט זה – ניגש אליו ג"כ מתוך ה current
  ```
  refObject.current = "new value"
  ```

useRef() על אלמנטי DOM

- יתרון גדול של אובייקטי reference היא היכולת גישה ישירות לאלמנטי DOM ולשנות אותם באמצעות ה current.
- במקרה זה נצטרך לבצע את אותם פעולות כמו שביצענו קודם:
  ייבוא useRef מ react.
  איתחול אובייקט כ reference
```
const inputRef = useRef()
```
- אם נרצה להשוות לאובייקט DOM צריך להוסיף פעולה נוספת של איתחול ה ref באלמנט DOM בצורה הבאה:
```
<input ref={inputRef}></input>
```
- ref הוא attribute של html המקצה אובייקט עם הפניה לרכיב DOM.
- עכשיו לגשת ולקבל או לשנות את תוכן רכיב DOM זה – (input) ניגש ישירות ע"י ה current.
- לדוגמא בשביל לקבל את תוכן ה input:

```
<input ref={inputRef} onChange={(e)=> getInputValue()}></input>
```

```
const getInputValue = () =>{
    const inputValue = inputRef.current.value
    console.log(inputValue)
}
```

React

useRef() על אלמנטי DOM

- לשנות את vlaue אלמנט זה :

```
inputRef.current.value = "new value"
```

- ניתן לגשת בצורה כזו לכל attribute האלמנטים (גם צבעים ו style) ולשנות איך ומתי שצריך באמצעות גישה מתוך ה current.
- לדוגמא שינוי צבע ה input בעת הכנסת תוכן:

```
<input ref={inputRef} onChange={(e) => changeInputColor()}></input>
```

```
const changeInputColor = () => {
    inputRef.current.style.color = "green"
}
```

useRef() במקום useState()

- ה – יתרון של useRef() הוא אפשרות השימוש שלו בדיוק כמו useState() – לשנות ערך משתנים אבל עם יתרון גדול – useState בכל פעם שמופעל, ז"א בכל שימוש ב set של המשתנה זה גורם לרינדור הקומפוננט מחדש. מה שקורה שהקומפוננט מתרנדרת המון פעמים.
- בעת גישה ל current האובייקט מיד משתנה ללא רינדור נוסף.
- מסקנה – מתי נשתמש ב useState() ומתי ב useRef()
רק useRef נותן לנו את הגישה לאיברי ה DOM אז במקרים כאלו בוודאי נשתמש ב useRef().
במשתנים רגילים - useState() הוא המקובל יותר בכך שמחזיר שני דברים – את המשתנה המכיל את הערך ואת האפשרות לשנות אותו באמצעות הפונקציה.
במקרים בהם נשנה את הערכים המון פעמים – עדיך להשתמש ב useRef() למנוע רינדור אינסופי, במשתנים אחרים שבהם שינוי הערכים הוא רגיל, נשתמש ב useState()

React hook form

- React hook form שונה מ forms אחרים בעיקר בכך שהיא מבוססת על שימוש באובייקט ref על מנת לאתחל את נתוני ה form ולשנות אותם במקום להיות מבוססת על states ובכך גורמת לטפסים להיות ביצועיים יותר ומפחיתה את מספר ה re-render – רינדורי הקומפוננט.
- בשביל להשתמש בה צריך להתקין את הספריה ע"י הפקודה: npm I react-hook-form
- ל react-hook-form יש אובייקט בשם useForm באמצעותו נגדיר את ה form.
ניצור קומפוננט חדש ל form בתוכו נייבא את הספריה:

```
import { useForm } from "react-hook-form";
```

- בתוך הקומפוננט ניצור את האובייקטים הבאים:
register, handleSubmit ישמשו אותנו ב form.
ה register נועד להיות בתוך ה inputs של הקלט
באמצעותו נוכל לעשות עליהם אימות ולעקוב אחר השינויים.
נעביר אותו בשדות בצורה הבאה:
כך ה input נוצר כ ref
לכל input חייב להיות attribute name עם ערך ייחודי משלו.

```
function ReactHookForm() {
    const { register, handleSubmit } = useForm();
```

```
<input name="name" {...register('name')} />
```

React

המשך - React hooks form

- ה handleSubmit כמו שנשמע – אחראי על העברת הטופס לאחר שליחתו – (לחיצה על submit) הוא צריך להיות מועבר כערך ה onSubmit props של קומפוננט הטופס.
- טופס יכול לטפל בשני פונקציות.
  האחת תופעל לאחר הצלחת שליחת הטופס באים הוולידציה בשדות תקינה.
  השניה תופעל אם הוולידציה בשדה אינה תקינה.
- אנחנו מגדירים את פונקציות אלו. לדוגמא אם נרצה שבהצלחת שליחת הטופס ירשמו הנתונים ללוג ניצור את פונקציית ההצלחה:

```
const handleRegistration = (data) => console.log(data);
```

ונממש אותה ביצירת הטופס:

```
<form onSubmit={handleSubmit(handleRegistration)}>
```

כאשר הטופס יצליח להישלח יופעל הפונקציה שהגדרנו.

React

אילוצים - React hooks form

- ניתן לבצע אימות על השדות באמצעות העברת פרמטרים של אימות לפונקציית רישום הטופס.
- ניתן להשתמש בשביל אימות על השדות במאפיינים הבאים:
  required – אם השדה הוא שדה חובה – true או לא חובה - false.
  minlength, maxlength – מינימום או מקסימום תווים שיכולים להיות בשדה.
  min, max – ערך המינימלי או המקסימלי שיכול להיות בשדה.
  type – סוג השדה כגון email, password, text וכו.
  pattern – הגדרת תבנית מסוימת עבור הקלט.
- דוגמא לשדה עם required:

```
<input name="name" {...register('name', { required: true })} />
```

אפשר גם לתת הערה מתאימה כגון:

```
<input name="name" {...register('name', { required: "Name is rwquired!!" })} />
```

errors - React hooks form

- בשביל לראות את ההערות צריך להוסיף את אובייקט ה errors ל ()useForm בצורה הבאה:

```
const { register, handleSubmit , formState: { errors }} = useForm();
```

- בכל מקום בו נרצה שיוצג ההערה המתאימה, כמובן אם יצרנו אותה כגון ב required וכד' נשתמש באובייקט errors
לדוגמא – להצגת ההערה אם לא ניתן ערך ב input שהוא required עם הערה:

```
<input name="name" {...register('name', { required: "Name is rwquired!!" })} />
{errors?.name && errors.name.message}
```

- הסבר: במיקום בו נרצה להציג את ההערה נשים את הנ"ל.
- אובייקט ה error ידע לאיזה שדה להתייחס לפי ה name שבשדה השווה לאובייקט שנתנו לו – errors.name צריך להיות אותו שם.

ניהול האימותים בטופס - React hooks form

- דרך נוחה לנהל את כל האילוצים שנשים על השדות באובייקט אחד, ולממש בכל שדה את שהגדרנו לו.
- לדוגמא אובייקט הגדרת אילוצים:

```
const registerOptions = {
    name: { required: "Name is required" },
    email: { required: "Email is required" },
    password: {
      required: "Password is required",
      minLength: {
        value: 8,
        message: "Password must have at least 8 characters"
      }
    }
  };
```

- המימוש בשדות:

```
<input name="name" {...register('name', registerOptions.name)} />
```

React

שיעורי בית

- ניווט עמודים + react hooks form :
צרו קומפוננט Login המכיל שני inputs להכנסת שם משתמש והכנסת סיסמה.
כפתור לוגין שבלחיצה עליו בודקת האם שם המשתמש הוא תקין (אפשר כל סוג של בדיקה שאתן רוצות לדוגמא אם מכיל ערך או מכיל שם מסוים...) אם השם תקין – מנווט לעמוד Home עם שליחת הפרמטר של שם המשתמש לקומפוננט זה. אם שם המשתמש לא תקין – המשתמש יועבר לקומפוננט Registration – הרשמה המכיל react hooks form להכנסת פרטי המשתמש והרשמה.
בקומפוננט זה הכניסו וולידציות מתאימות.

- useRef : צרו קומפוננט נפרד עם שימוש ב useRef : המכיל 3 buttons. במקושרים כ ref לאובייקטים המוגדרים כ useRef.
בלחיצה על כל כפתור צבע רקע הכפתור ישתנה לצבע שתגדירו לו (לש משנה איזה)

בהצלחה!

React