HOOKIE

Hookie są dodatkiem, które pozwalają używanie i funkcjonalności bez użycia klas. Uproszczają kod, zwiększają jego czytelność, pozwalają na pobieranie danych w prosty i intuicyjny sposób

Przykładowe hookie

1. useState

używamy go, jeżeli będziemy chcieli wartość stanu wyświetlić na stronie

deklarujemy go w funkcji App, przed return

const [nazwa,nazwaState]=useState();

nazwa-to nazwa stanu (zmiennej)
nazwaState – to funkcja, która zmienia stan
useState() – w nawiasie możemy przypisać wartość początkową stanu

np. **const [klasa,klasaState]=useState(„3d”);**

do stanu klasa na początku wpisze 3d

potem w kodzie możemy np. napisać

klasaState(„4d”);

i wtedy do stanu klasa mamy zapisane 4d

uwaga, pamiętajmy , że na górze musimy mieć
import {useState} from ‘react’;

2. useEffect()

służy do uruchamiania kodu **po wyrenderowaniu komponentu** albo **po zmianie jakichś danych**

```
useEffect(() => {
  //działa przy każdej zmianie
});


useEffect(() => {
//działa tylko raz, przy pierwszym uruchomieniu komponentu (strony)
}, []);


useEffect(() => {
//działą tylko wtedy gdy zmieni się wartość tego co wpiszemy w nawiasy kwadratowe
}, [count]);
```

W useEffect możemy użyć funkcji setInterval, czyli fukcję wbudowaną w JS, która uruchamia się co jakiś (określony) czas, np.

```
useEffect(() => {

  const timer = setInterval(() => {
        console.log(„wyświetlę ten tekst co 2 sekundy”);
  }, 2000);

  return () => clearInterval(timer);

}, []);
```

2000 oznacza dwie sekundy (ten czas podajemy w ms, przy czym 1000ms=1s)

Gdy używamy setInterval, na koniec musimy usunąć interwał, żeby nie tworzyły się jego kopie, robimy, to w taki sposób:

```
return () => clearInterval(timer);
```

timer to zmienna. Pod którą przypisaliśmy setInterval

Oczywiście pamiętajmy, że na samym górze musimy mieć import

import {useEffect} from ‘react’;

Jeżeli używamy zarówno useState i useEffect, to możemy zrobić jeden import

import {useState,useEffect} from ‘react’;

3. useRef()

Używamy go, gdy

a. gdy chcemy przechowywać wartość, której nie wyświetlamy na stronie

```
const licznik=useRef(0);
```

licznik to zmienna, która na początku otrzyma wartość zero

tej zmiennej nie wyświetlisz na stronie, ale możesz w konsoli w taki sposób:

```
console.log(licznik.current);
```

b. gdy chcemy uzyskać dostęp do elementów DOM

```
const pole = useRef();

const focusInput = () => {
```

```
  pole.current.focus();
};

return (
  <>
    <input ref={pole} />
    <button onClick={focusInput}>Ustaw fokus</button>
  </>
);
```

Pamiętaj, że jak odwołujesz się do jakiegokolwiek elementu HTML to musisz w nim napisać ref={nazwa}, gdzie nazwa to nazwa hooka Ref

Pamiętaj też, o imporcie

import {useRef} from 'react';

Poniżej masz przykłady, co możesz oprócz focus() pisać po current :

.offsetWidth szerokość elementu (z paddingiem i borderem)
.offsetHeight wysokość elementu
.clientWidth szerokość **bez borderów**, ale z paddingiem
.clientHeight wysokość bez borderów
.scrollWidth cała szerokość przewijalnej zawartości
.scrollHeight cała wysokość przewijalnej zawartości
.getBoundingClientRect() dokładne położenie i rozmiar (x, y, width, height)

zmiana tekstu divRef.current.textContent = "Nowy tekst"
zmiana HTML divRef.current.innerHTML = "tekst"
zmiana styl divRef.current.style.backgroundColor = "red"
dodanie klasy CSS divRef.current.classList.add("aktywny")
usunięcie klasy CSS divRef.current.classList.remove("aktywny")

pobranie wartości inputRef.current.value
usunięcie wartości inputRef.current.value = "tekst"
ustawienie focus (kursor) inputRef.current.focus()
usunięcie focus inputRef.current.blur()
sprawdzenie zaznaczenia checkboxRef.current.checked

odtwarzanie audioRef.current.play()
pauzowanie audioRef.current.pause()
sprawdzanie czasu audioRef.current.currentTime
ustawienie głośności audioRef.current.volume = 0.5
sprawdzenie, czy gra audioRef.current.paused (true/false)