React

Setup

Zainstalować NPM (Node Package Manager) - instrukcja instalacji poniżej:

https://docs.npmjs.com/downloading-and-installing-node-js-and-npm

TypeScript:

W wielkim skrócie - PO CO?

Typescript umożliwia typowanie w JavaScript.

Np. let x: number = 5

albo let y: string = “Hello”

TypeScript jest kompilowany do czystego JavaScript.

https://www.w3schools.com/typescript/index.php

W powyższym kursie należy zapoznać się z sekcjami poczynając od TS Introduction, aż do TS Aliases & Interfaces. Wiedza ta pozwoli na podstawowe wykorzystanie TS w React.

Podstawowa inicjalizacja aplikacji

1) Stworzenie nowego projektu React

```
npx create-react-app my-app
```

lub na dodatkowe punkty z TypeScript

```
npx create-react-app my-app --template typescript
```

2) Dodanie potrzebnych rozszerzeń (nieobowiązkowe)

- ESLint
- Prettier

3) Dodanie TypeScript'a

```
npm install @types/react @types/react-dom
```

4) Zainstalowanie React DevTools

5) Uruchomienie projektu

```
cd my-app
npm run dev
```

Utwożenie aplikacji z wykożystanie Vite

```
npm install -D vite
npm create vite@latest
Ok to proceed? (y) - y
Project name - Nazwa projektu
Select a framework - React
Select a variant - TypeScript
cd Nazwa projektu
npm install
npm run dev
```

Pierwszy komponent

Co to komponent?

Przykładowy kod

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const root = ReactDOM.createRoot(document.getElementById('root'));
const element = <Welcome name="Sara" />;
root.render(element);
```

Proszę poeksperymentować:

1. Zmiana paramatru
2. Dodanie parametru
3. Dodanie kilku komponentów **Welcome**
4. Stworzenie nowego komponentu **Goodbye**
5. Stworzenie wartości dynamicznej **K / M**, która będzie wyświetlana na podstawie ostatniej litery imienia (jeżeli **a** to zakładamy, że kobieta)

Przydatne linki

Context - może się przydać do utrzymywania kontekstu zalogowanego użytkownika.

Reducer - jeżeli **state** będzie skomplikowany (dużo użyc **useState**) warto rozważyć użycie **reducer'a**

Ref - jeżeli zachodzi potrzeba bezpośredniego dostępu do elementu DOM

useNavigate - nawigacja za pomocą kodu

Zadania

Uwaga: Ilekroć w zestawie będzie mowa o umieszczeniu czegoś w folderze, należy ten folder stworzyć w lokacji src/components. Wszystkie komponenty należy testować poprzez umieszczanie ich w app.tsx.

Zadanie 1.[0.5 pkt]

Komponenty stworzone w zadaniach 1.1 oraz 1.2 proszę umieścić w folderze koszyk

Zadanie 1.1.

Proszę utworzyć następujące komponenty:

- **Koszyk**
- **Produkt**

Komponent **Produkt** powinien przyjmować property **nazwa** oraz wyświetlać tą nazwę. Komponent **Koszyk** należy umieścić w głównym komponencie aplikacji (app.tsx) a wewnątrz niego utworzyć 5 dowolnych produktów (np. jabłko, gruszka), poprzez pięciokrotne wykorzystanie komponentu **Produkt** z różnymi nazwami

Zadanie 1.2.

Modyfikacja zadania 1.1 w ten sposób, że nazwy produktów znajdują się w **tablicy Produkty** utworzonej na samej górze komponentu **NowyKoszyk** (skopiowany komponent **Koszyk**) i na tej podstawie produkty tworzone z wykorzystaniem funkcji Produkty.map wewnątrz returna. Należy skorzystać z tego samego komponentu **Produkt** co w zadaniu 1.1.

Zadanie 2.[0.5 pkt]

Komponenty stworzone w zadaniach 2.1 oraz 2.2 proszę umieścić w folderze liczniki

Zadanie 2.1.

Proszę stworzyć komponent **Licznik**. Komponent powinien wyświetlać w divie licznik, o początkowym stanie zero. Oprócz tego powinien zawierać przycisk z tekstem Dodaj, który zwiększy stan licznika o 1

Zadanie 2.2.

Modyfikacja zadania 2.1. w ten sposób, że komponent **NowyLicznik** (skopiowany komponent **Licznik**) nie zawiera bezpośrednio przycisku. Należy osobno utworzyć komponent **Przycisk**, w którym będzie znajdował się przycisk służący do inkrementacji. Sam div z licznikiem powinien pozostać w komponencie **NowyLicznik**. Podpowiedź: Należy przekazać jako prop do komponentu Przycisk funkcję, która znajduje się w komponencie nadrzędnym i odpowiada za inkrementację.

Zadanie 3.[1.5 pkt]

Komponenty stworzone w zadaniach 3.1 do 3.3 proszę umieścić w folderze formularze

Zadanie 3.1.

Proszę stworzyć komponent **Formularz**. Wewnątrz komponentu należy umieścić jeden input (typu tekstowego) oraz diva. Wpisując tekst w inpucie, ma się na żywo replikować do diva.

Zadanie 3.2.

Proszę stworzyć komponent **Haslo**. Wewnątrz komponentu należy umieścić dwa inputy (typu tekstowego) odpowiednio podpisane "Hasło" oraz "Powtórz Hasło". Poniżej należy umieścić diva, który będzie wyświetlał jedną z następujących wiadomości.

- Proszę wprowadzić hasło - jeżeli żaden z inputów nie zawiera tekstu.
- Hasła nie są zgodne - jeśli hasła nie są identyczne
- Jeżeli hasła są identyczne, div powinien pozostać pusty

Zadanie 3.3.

Modyfikacja zadania 3.2. w ten sposób, że komponent **Logowanie** (skopiowany komponent **Haslo**) dodatkowo posiada pole tekstowe **Nazwa użytkownika**. Należy usunąć diva wyświetlającego wiadomość o błędzie i zamiast tego stworzyć przycisk Logowanie o następującym zachowaniu:

- Jeżeli którekolwiek z pól tekstowych jest puste - przycisk powinien być wyłączony (atrybut disabled: https://www.w3schools.com/tags/att_button_disabled.asp)
- Jeżeli wszystkie pola tekstowe posiadają tekst, ale hasła **nie są zgodne** - przycisk ma być włączony i wysyłać alert (funkcja alert) Hasła nie są zgodne
- Jeżeli wszystkie pola tekstowe posiadają tekst i **hasła są zgodne** - przycisk ma być włączony i wysyłać alert Zalogowano poprawnie

Zadanie 4. [1.0 pkt]

Komponenty stworzone w zadaniach 4.1. oraz 4.2. proszę umieścić w folderze inne

Zadanie 4.1.

Proszę stworzyć nowy komponent Ternary. Na górze komponentu należy utworzyć dwie zmienne a oraz b. Obydwa powinny być typu boolean (Domyślnie można nadać im wartości **a** = true, oraz **b** = false). Wewnątrz funkcji return należy skorzystać z ternary operatora i stworzyć dwa divy. Każdy z nich powinien pokazywać odpowiednio "Stwierdzenie **a** jest prawdziwe" lub "Stwierdzenie a jest fałszywe" - analogicznie dla **b**. Opis działania ternary operatora: https://www.w3schools.com/react/react_conditional_rendering.asp

Zadanie 4.2.

To zadanie oraz wiele zagadnień w React, będzie korzystać z konceptu **spread operatora**. Należy zapoznać się z działaniem tego konceptu:

- (1) https://www.w3schools.com/react/react_es6_spread.asp
- (2) https://www.dhiwise.com/post/how-to-simplify-your-react-code-with-the-spread-operator Proszę stworzyć komponent Aktualizacja. Wewnątrz komponentu należy stworzyć stan produkt, którego domyślną wartość będzie obiekt { nazwa: **"Pomidor"**,

cena: 50 }. Jego reprezentacja ma się wyświetlać w jakimś divie (np. Aktualnie **Pomidor** kosztuje 50). Należy stworzyć przycisk Zmień cenę który spowoduje zmianę ceny pomidora na 100. Proszę zastosować do zmiany stanu notacji, która jest opisana w linku nr. (2) powyżej, w sekcji **Spread Operator with State** - tj. użyć **prev** oraz **spread operatora**.

Zadanie 5. [1.5 pkt]

Komponenty stworzone w zadaniach 5.1. oraz 5.2. proszę umieścić w folderze studenci

Zadanie 5.1.

Prosze stworzyć komponent **Studenci**. Na górze komponentu powinna znajdować się zadeklarowana tablica **Students** z listą przykładowych studentów. Każdy student powinien zawierać pola imie, nazwisko oraz rocznik. Należy stworzyć poprawnie interfejs Student z tymi polami w TypeScript i przypisać go jako typ tablicy T. Zapis będzie wtedy wyglądać tak: const Students: Student[] = [{ student1 }, ...] Z wykorzystaniem funkcji **map** w returnie, studentów należy umieścić w tabeli HTML.

Zadanie 5.2.

Modyfikacja zadania 5.1. w ten sposób, że komponent **StudentManager** (skopiowany komponent **Studenci**) przechowuje listę użytkowników w **stanie**, a nie w tablicy. Domyślna wartość stanu ma być taka sama, jak wartość przykładowej tablicy w zadaniu 9. Dodatkowo należy stworzyć komponent **Dodawanie**, który zawiera formularz do dodawania studentów (zawierający 3 inputy, po jednym dla każdego z pól w studencie). Komponent ten należy umieścić poniżej tabeli, wewnątrz komponentu **StudentManager**. Po kliknięciu przycisku Dodaj w komponencie **Dodawanie**, tabela powinna się zaktualizować i dodać nowego studenta zgodnie z wpisanymi danymi, a formularz powinien się wyczyścić. Należy zadbać o poprawną walidację danych (w dowolnej formie), mając na uwadze, że wszystkie dane muszą być wpisane, a dodatkowo rocznik musi być liczbą.

Zadanie 6. [1.5 pkt]

Komponenty stworzone w zadaniach 6.1 do 6.3. proszę umieścić w folderze efekty

Zadanie 6.1.

Zapoznaj się z działaniem useEffect: https://www.w3schools.com/react/react_useeffect.asp
Proszę skopiować komponent **Licznik** z folderu liczniki, który został utworzony w zadaniu 4 i kopię umieścić w folderze efekty. Za każdym razem, gdy licznik się zaktualizuje, należy wypisać w konsoli (console.log) **Licznik** zwiększył się do **X**. Dodatkowo, tuż po załadowaniu komponentu należy wypisać Hello world (useEffect z pustą tablicą dependencji).

Zadanie 6.2.

Proszę stworzyć komponent **Tytul**. Powinien zawierać on tylko jeden input, którego treść będzie na bieżąco synchronizowana jako tytuł strony - ten, który wyświetla się w karcie przeglądarki. (Podpowiedź: ustawianie tytułu strony można zrealizować używając document.title = "Nowy tytuł"). **Proszę nie modyfikować tytułu bezpośrednio w evencie onChange inputa**. Należy po

każdej zmianie inputa modyfikować stan z nim związany, a następnie wykrywać zmianę z wykorzystaniem useEffect.

Zadanie 6.3.

Proszę stworzyć komponent Odliczanie. Proszę stworzyć stan licznik, który na początku ma wartość 15 sekund. Należy go wyświetlać w jakimś divie. (Proszę wyświetlać z dokładnością do **0.1** sekundy, np. 14.4 sek) Proszę także stworzyć przycisk, który będzie miał napis **START** lub **STOP**. (Domyślnie wartość to **START**, gdyż zegar jeszcze nie odlicza) Po kliknięciu start należy rozpocząć obliczanie i zmienić wartość przycisku na **STOP** - teraz jego kliknięcie powinno zapauzować odliczanie i ponownie zmienić wartość przycisku na **START**. Po dojściu zegara do 0.0 sek należy wyłączyć przycisk (atrybut disabled) i jego wartość ustawić na **Odliczanie zakończone**. Proszę zapoznać się z linkiem poniżej - przedstawia on w jaki sposób można poprawnie zastosować setInterval wraz z UseEffect. https://www.dhiwise.com/post/the-ultimate-tutorial-on-using-react-setinterval-effectively

Zadanie 7. [1.5 pkt]

Komponenty stworzone w zadaniach 7.1 oraz 7.2. proszę umieścić w folderze produkty

Zadanie 7.1.

Proszę stworzyć komponent **Komentarz**. Powinien on przyjmować następujące propsy: id, body, postId, likes, user (user jest obiektem o polach id, username, fullName). Należy w dowolnej (atrakcyjnie graficznej) formie wyświetlić te dane, tak, aby obrazowały one komentarz dodany przez użytkownika np. pod postem / filmikiem. Dodatkowo komponent ma przechowywać otrzymaną w propsie liczbę likes w stanie z możliwością jego aktualizacji poprzez klikanie łapki w dół oraz łapki w górę. Komponent będzie niezbędny do wykorzystania w Zadaniu 7.2. Jeżeli nie planujesz jego realizacji, proszę stworzyć przykładowe wywołanie z hard-coded danymi, w celu możliwości zobaczenia efektów realizacji.

Zadanie 7.2.

Proszę stworzyć komponent **Komentarze**. Wewnątrz jego, korzystając z danych pod adresem https://dummyjson.com/comments należy pobrać (korzystając z fetch lub axios) komentarze i wyświetlić je na stronie (proszę wykorzystać utworzony w zadaniu 7.1 komponent **Komentarz**.). Podpowiedź: w useEffect z pustą tablicą dependencji (tj. po załadowaniu komponentu) należy użyć fetch, a po pobraniu danych (**.then** w fetchu) ustawić stan z pobranymi komentarzami. Aby uzyskać pełną ilość punktów za to zadanie, należy stworzyć odpowiednie interfejsy w TypeScript i wykorzystać je (przynajmniej jeden - dla komentarza)

Zadanie 8. [2.0 pkt]

Zadanie 8.1. proszę zrealizować w osobnym projekcie Reacta

Zadanie 8.1.

Proszę stworzyć komponent **Licznik** taki sam jak w zadaniu 2.1. Należy go zmodyfikować w ten sposób, aby po wyjściu i wejściu na stronę stan licznika zachował się. Proszę użyć **local storage**: https://www.w3schools.com/jsref/prop_win_localstorage.asp

Zadanie 8.2.

Proszę stworzyć następującą aplikację wykorzystując React Router (https://www.w3schools.com/react/react_router.asp) oraz local storage. Proszę zapoznać się w jaki sposób można przechowywać obiekty w local storage w poniższym poradniku: https://blog.logrocket.com/storing-retrieving-javascript-objects-localstorage/

- Aplikacja ma mieć możliwość wyświetlania oraz dodawania artykułów. Każdy z artykułów składa się z identyfikatora, tytułu oraz treści.
- Po wejściu na podstronę /blog powinna wyświetlić się lista tytułów artykułów na blogu, wraz z linkami, poprzez które można przejść na stronę artykułu.
- Po wejściu na podstronę /article/id ma wyświetlić się cały artykuł tj. zarówno jego tytuł oraz treść.
- Po wejściu na podstronę /dodaj należy wyświetlić formularz z dwoma polami tekstowymi - tytuł oraz treść. Należy umożliwić w nim dodanie nowego artykułu.
- Po kliknięciu “DODAJ” na powyższej podstronie należy dodać artykuł do localstorage i przekierować do podstrony /blog
- Strona główna / ma zawierać przywitanie i link przekierowujący do podstrony /blog
- Istnieje możliwość uzyskania dodatkowego 1.0 pkt poza bazą za atrakcyjną szatę graficzną w zadaniu.