Zaawansowany Programista JS

Dzień 9

Plan

- Stylowanie aplikacji Reactowej
- Firebase
  - Pobieranie danych z firebase
  - Zapisywanie danych do firebase
  - Usuwanie danych z firebase
  - Modyfikowanie danych w firebase
  - Deploy aplikacji za pomocą Firebase Hosting

Stylowanie aplikacji Reactowej

W React.JS stylowanie aplikacji odbywa się najczęściej za pomocą arkuszy stylów CSS lub poprzez użycie bibliotek stylów takich jak Bootstrap, Material UI czy Styled Components. Istnieje kilka sposobów na dodanie stylów do komponentów w React.JS:

Dodanie stylów do arkuszy stylów CSS: można umieścić arkusz stylów CSS w pliku i zaimportować go do komponentu za pomocą instrukcji import, a następnie przypisać klasę lub identyfikator do elementu.

```
import React from 'react';
import './styles.css';

function MyComponent() {
  return (
    <div className="my-component">
      <h1>Hello World</h1>
    </div>
  );
}
```

Użycie bibliotek stylów: `React.JS` jest zgodny z wieloma popularnymi bibliotekami stylów takimi jak Bootstrap czy Material UI. Dzięki temu można łatwo wykorzystać gotowe komponenty i style z tych bibliotek.

```
import React from 'react';
import { Button } from 'react-bootstrap';
```

```
function MyComponent() {
  return (
    <div>
      <h1>Hello World</h1>
      <Button>Click me!</Button>
    </div>
  );
}
```

Użycie bibliotek do stylowania komponentów: biblioteki takie jak Styled Components pozwalają na tworzenie stylów w plikach JavaScript i zastosowanie ich bezpośrednio w komponentach.

```
import React from 'react';
import styled from 'styled-components';

const Button = styled.button`
  background: blue;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
`;

function MyComponent() {
  return (
    <div>
      <h1>Hello World</h1>
      <Button>Click me!</Button>
    </div>
  );
}
```

Niezależnie od wybranego sposobu, stylowanie w React.JS polega na tworzeniu reużywalnych, łatwych do utrzymania i skalowalnych komponentów interfejsu użytkownika.

Firebase

Firebase to platforma stworzona przez Google, która oferuje narzędzia dla programistów umożliwiające tworzenie i wdrażanie aplikacji mobilnych i webowych. Firebase zawiera wiele usług w chmurze, takich jak hosting, autentykacja użytkowników, baza danych NoSQL, przechowywanie plików w chmurze, przetwarzanie wiadomości, analizy i wiele innych. Firebase jest bardzo popularny w społeczności programistów, ponieważ oferuje wiele narzędzi do budowy aplikacji zorientowanych na użytkownika w sposób szybki i łatwy, bez konieczności zarządzania infrastrukturą serwera. Firebase jest także często stosowany w połączeniu z React.JS, ponieważ wiele narzędzi Firebase ma natywne biblioteki dla React.JS.

Pobieranie danych z firebase

Aby pobrać dane z Firebase Realtime Database przy użyciu React.JS, należy najpierw zainicjować połączenie z bazą danych Firebase. Można to zrobić przez utworzenie

instancji Firebase w pliku konfiguracyjnym i wywołanie funkcji inicjalizującej z odpowiednimi parametrami. Następnie można wykorzystać hook useState, aby przechowywać pobrane dane w stanie komponentu.

Przykładowy kod, który pobiera dane z Firebase Realtime Database i wyświetla je w komponencie React.JS:

```
import React, { useState, useEffect } from 'react';
import firebase from 'firebase/app';
import 'firebase/database';

const config = {
// konfiguracja Firebase
};

// inicjalizacja Firebase
firebase.initializeApp(config);

const App = () => {
  const [data, setData] = useState({});

  useEffect(() => {
    const dbRef = firebase.database().ref();
    dbRef.on('value', (snapshot) => {
      setData(snapshot.val());
    });
  }, []);

  return (
    <div>
      <h1>{data.title}</h1>
      <p>{data.description}</p>
    </div>
  );
};

export default App;
```

W powyższym kodzie używamy hooka `useEffect`, aby pobrać dane z Firebase Realtime Database w momencie renderowania komponentu. Wykorzystujemy metodę on na referencji do bazy danych, aby słuchać zmian w bazie danych. Gdy zmiany wystąpią, wywołujemy funkcję `setData`, aby zaktualizować stan komponentu i wyświetlić pobrane dane w renderze.

Zapisywanie danych do firebase

Aby zapisać dane do Firebase Realtime Database przy użyciu React.JS, należy utworzyć instancję bazy danych Firebase i wywołać na niej odpowiednią metodę, taką jak `set`, `push` lub `update`, w zależności od wymagań. Następnie można przekazać dane, które chcemy zapisać, do wywołania metody.

Poniżej przedstawiam przykładowy kod, który zapisuje dane do Firebase Realtime Database po naciśnięciu przycisku w komponencie React.JS:

```
import React, { useState } from 'react';
import firebase from 'firebase/app';
import 'firebase/database';

const config = {
// konfiguracja Firebase
};

// inicjalizacja Firebase
firebase.initializeApp(config);

const App = () => {
  const [inputValue, setInputValue] = useState('');

  const handleInputChange = (event) => {
    setInputValue(event.target.value);
  };

  const handleSaveData = () => {
    const dbRef = firebase.database().ref();
    dbRef.push(inputValue);
    setInputValue('');
  };

  return (
    <div>
      <input type="text" value={inputValue} onChange={handleInputChange} />
      <button onClick={handleSaveData}>Zapisz do bazy danych</button>
    </div>
  );
};

export default App;
```

W powyższym kodzie używamy hooka `useState`, aby przechowywać wartość wprowadzoną przez użytkownika w stanie komponentu. Gdy użytkownik kliknie przycisk "Zapisz do bazy danych", wywołujemy funkcję handleSaveData, która tworzy referencję do bazy danych i używa metody push do zapisania wartości z inputu do bazy danych Firebase. Następnie czyszczone jest pole input, aby umożliwić użytkownikowi wprowadzenie kolejnej wartości do zapisu.

Usuwanie danych z firebase

Aby usunąć dane z Firebase Realtime Database przy użyciu React.JS, należy utworzyć referencję do węzła, który chcemy usunąć, a następnie wywołać na niej metodę `remove()`.

Poniżej przedstawiam przykładowy kod, który usuwa dane z Firebase Realtime Database po naciśnięciu przycisku w komponencie React.JS:

```
import React, { useState, useEffect } from 'react';
import firebase from 'firebase/app';
```

```
import 'firebase/database';

const config = {
// konfiguracja Firebase
};

// inicjalizacja Firebase
firebase.initializeApp(config);

const App = () => {
  const [data, setData] = useState([]);

  useEffect(() => {
    const dbRef = firebase.database().ref();
    dbRef.on('value', (snapshot) => {
      const val = snapshot.val();
      setData(val);
    });
  }, []);

  const handleDeleteData = (key) => {
    const dbRef = firebase.database().ref(key);
    dbRef.remove();
  };

  return (
    <div>
      <ul>
        {Object.keys(data).map((key) => (
          <li key={key}>
            {data[key]}
            <button onClick={() => handleDeleteData(key)}>Usuń</button>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default App;
```

W powyższym kodzie używamy hooka `useEffect`, aby pobrać dane z bazy danych Firebase przy pierwszym renderowaniu komponentu i przypisać je do stanu komponentu. Następnie wyświetlamy wartości z bazy danych w liście HTML i dodajemy przycisk "Usuń", który wywołuje funkcję handleDeleteData z kluczem węzła jako argument. Funkcja ta tworzy referencję do węzła o podanym kluczu i używa metody remove() do usunięcia go z bazy danych Firebase.

Modyfikowanie danych w firebase

Aby zmodyfikować dane w Firebase Realtime Database przy użyciu React.JS, należy utworzyć referencję do węzła, który chcemy zmodyfikować, a następnie wywołać na niej

metodę `set()` z nową wartością.

Poniżej przedstawiam przykładowy kod, który umożliwia zmianę danych w Firebase Realtime Database po naciśnięciu przycisku w komponencie React.JS:

```
import React, { useState, useEffect } from 'react';
import firebase from 'firebase/app';
import 'firebase/database';

const config = {
// konfiguracja Firebase
};

// inicjalizacja Firebase
firebase.initializeApp(config);

const App = () => {
  const [data, setData] = useState([]);

  useEffect(() => {
    const dbRef = firebase.database().ref();
    dbRef.on('value', (snapshot) => {
      const val = snapshot.val();
      setData(val);
    });
  }, []);

  const handleUpdateData = (key, newData) => {
    const dbRef = firebase.database().ref(key);
    dbRef.set(newData);
  };

  return (
    <div>
      <ul>
        {Object.keys(data).map((key) => (
          <li key={key}>
            {data[key]}
            <button onClick={() => handleUpdateData(key, 'nowa wartość')}>
              Zmień
            </button>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default App;
```

W powyższym kodzie używamy hooka useEffect, aby pobrać dane z bazy danych Firebase przy pierwszym renderowaniu komponentu i przypisać je do stanu komponentu. Następnie wyświetlamy wartości z bazy danych w liście HTML i dodajemy przycisk "Zmień", który

wywołuje funkcję handleUpdateData z kluczem węzła i nową wartością jako argumenty. Funkcja ta tworzy referencję do węzła o podanym kluczu i używa metody set() do zmiany jego wartości na nową wartość.

Deploy aplikacji za pomocą Firebase Hosting

Aby wykonać deploy aplikacji React.JS napisanej w Create React App przy użyciu Firebase Hosting, należy postępować zgodnie z poniższymi krokami:

1. Zainstaluj Firebase CLI na swoim komputerze. Można to zrobić za pomocą polecenia `npm install -g firebase-tools`.
2. Zaloguj się do Firebase za pomocą Firebase CLI, wykonując polecenie `firebase login`.
3. Wejdź do katalogu projektu Create React App za pomocą konsoli.
4. Zainstaluj zależności Firebase do projektu, wykonując polecenie `npm install firebase`.
5. Zainicjuj projekt Firebase, wykonując polecenie `firebase init`. Wybierz opcję "Hosting", a następnie postępuj zgodnie z instrukcjami wyświetlanymi w konsoli.
6. W pliku `firebase.json` znajdziesz sekcję hosting. W polu "public" wpisz "build" (lub inny katalog, w którym znajduje się plik index.html).
7. Zbuduj aplikację React.JS w trybie produkcyjnym, wykonując polecenie `npm run build`.
8. W katalogu build znajdziesz pliki, które musisz przesłać na Firebase Hosting. Można to zrobić za pomocą Firebase CLI, wykonując polecenie firebase deploy. Firebase CLI przesłał pliki z katalogu build na Firebase Hosting.
9. Po zakończeniu przesyłania plików Firebase wyświetli adres URL, pod którym można zobaczyć opublikowaną stronę internetową.