Aplikacje mobilne

Laboratorium 4 – Context, asyncstorage, axios, json-server

Mateusz Pawełkiewicz

1) useContext

useContext to hook w React, który umożliwia korzystanie z wartości kontekstu w komponencie funkcyjnym. Kontekst w React jest mechanizmem, który pozwala przekazywać dane przez drzewo komponentów bez konieczności przekazywania props przez każdy poziom komponentu.

Przykładowa sytuacja, w której useContext może być przydatne, to gdy chcemy dostarczyć globalne dane, takie jak informacje o użytkowniku czy preferencje stylów.

Oto prosty przykład:

```
import React, { createContext, useContext } from 'react';

// Tworzymy kontekst
const UserContext = createContext();

// Komponent dostarczający dane do kontekstu
const UserProvider = ({ children }) => {
  const user = { username: 'JohnDoe', role: 'user' };

  return (
    <UserContext.Provider value={user}>
      {children}
    </UserContext.Provider>
  );
};
```

```jsx
// Komponent, który korzysta z danych z kontekstu za pomocą useContext
const UserProfile = () => {
  const user = useContext(UserContext);

  return (
    <div>
      <h2>Profil Użytkownika</h2>
      <p>Użytkownik: {user.username}</p>
      <p>Rola: {user.role}</p>
    </div>
  );
};
```

```jsx
// Główny komponent, który korzysta z UserProvider i renderuje UserProfile
const App = () => {
  return (
    <UserProvider>
      <UserProfile />
    </UserProvider>
  );
};

export default App;
```

2) AsyncStorage

AsyncStorage w React Native to prosty, asynchroniczny, trwały, klucz-wartość system przechowywania danych, który można używać do przechowywania danych lokalnie na urządzeniu użytkownika.

a) Instalacja

```
npm install @react-native-async-storage/async-storage
```

Lub

```
yarn add @react-native-async-storage/async-storage
```

b) Użycie

```
import AsyncStorage from '@react-native-async-storage/async-storage';
```

Zapis danych:

```
const storeData = async (value) => {
  try {
    await AsyncStorage.setItem('@storage_Key', value)
  } catch (e) {
    // saving error
  }
}
```

Odczyt danych:

```
const getData = async () => {
  try {
    const value = await AsyncStorage.getItem('@storage_Key')
    if(value !== null) {
      // value previously stored
    }
  } catch(e) {
    // error reading value
  }
}
```

Usuwanie danych:

```
const removeValue = async () => {
  try {
    await AsyncStorage.removeItem('@storage_Key')
  } catch(e) {
    // remove error
  }
}
```

3) Axios

Axios to biblioteka do obsługi żądań HTTP w języku JavaScript, która działa zarówno w środowisku przeglądarki, jak i na platformie Node.js. Zapewnia łatwy sposób wysyłania żądań HTTP do serwera i obsługiwania odpowiedzi.

a) Instalacja

```
npm install axios
```

lub

```
yarn add axios
```

b) Podstawowe użycie:

Axios udostępnia metody do różnych rodzajów żądań HTTP, takich jak GET, POST, PUT, DELETE, itp. Oto przykład użycia dla GET:

```
// Import Axios w kodzie
import axios from 'axios';

// Wywołanie żądania GET
axios.get('https://api.example.com/data')
  .then(response => {
    console.log(response.data); // Dane z odpowiedzi
  })
  .catch(error => {
    console.error('Błąd żądania:', error);
  });
```

c) Przesyłanie danych:

Możemy również przesyłać dane w ciele żądania, na przykład podczas wysyłania żądania POST:

```
// Wywołanie żądania POST z danymi
axios.post('https://api.example.com/post-data', { key: 'value' })
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error('Błąd żądania:', error);
  });
```

d) Interceptory

Axios umożliwia korzystanie z interceptorów do przechwytywania żądań i odpowiedzi przed ich obsługą. To przydatne do manipulacji danymi przed wysłaniem lub po otrzymaniu odpowiedzi.

```
// Dodanie interceptorów
axios.interceptors.request.use(config => {
  // Manipulacja konfiguracją przed wysłaniem żądania
  return config;
}, error => {
  return Promise.reject(error);
});

axios.interceptors.response.use(response => {
  // Manipulacja odpowiedzią przed przekazaniem jej do kodu obsługującego
  return response;
}, error => {
  return Promise.reject(error);
});
```

4) Json-server

a) Instalacja
npm i json-server –g

b) Tworzenie pliku db.json

```
{
  "posts": [
    { "id": 1, "title": "json-server", "author": "typicode" }
  ],
  "comments": [
    { "id": 1, "body": "some comment", "postId": 1 }
  ],
  "profile": { "name": "typicode" }
}
```

c) Uruchomienie serwera

json-server --watch db.json --port 3004 --host <adres ip>