Aplikacje mobilne

Laboratorium 1

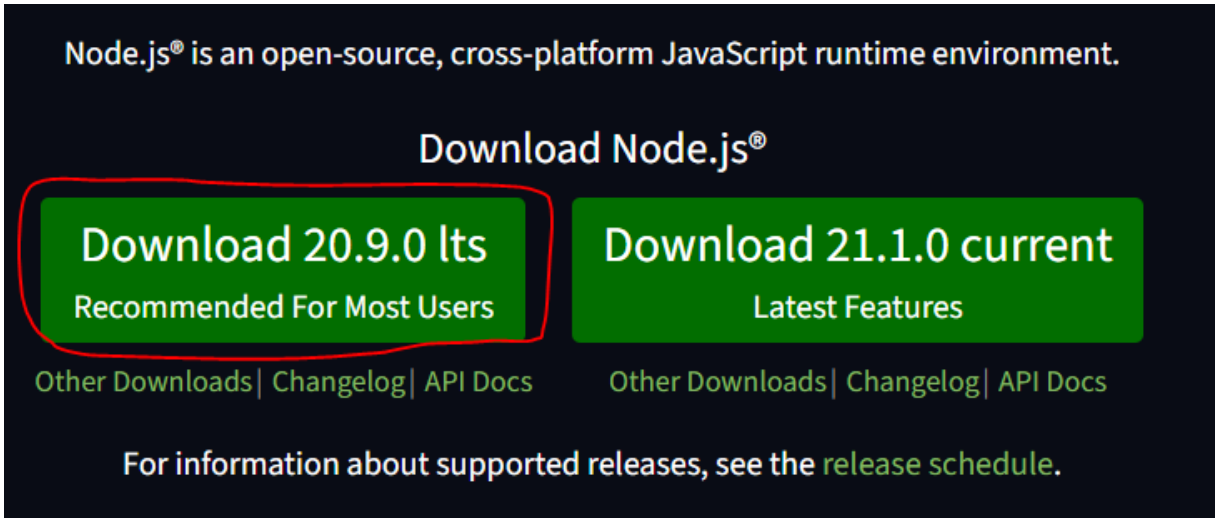
Pierwsza aplikacja / hook useState

Mateusz Pawełkiewicz

Pierwsza aplikacja

1. Instalacja narzędzi:

Instalacja Node.js i npm (lub yarn) ze strony: https://nodejs.org/.
Rekomendowane jest zainstalowanie wersji LTS.



2. Tworzenie nowego projektu:

```
# Create a project named my-app
- npx create-expo-app my-app

# Navigate to the project directory
- cd my-app
```

3. Uruchomienie serwera deweloperskiego

```
- npx expo start
```

Otworzy się okno przeglądarki z kodem QR. Zeskanuj ten kod za pomocą aplikacji Expo Go na swoim urządzeniu mobilnym (dostępne w App Store lub Google Play). Twoja aplikacja zostanie uruchomiona na Twoim urządzeniu. Ewentualnie wybierz opcję „run on android”, aby uruchomić aplikację w emulatorze.

4. Rozpoczęcie pracy nad aplikacją:

Otwórz plik `App.js` w wybranym przez siebie edytorze kodu (np. Visual Studio Code).

```
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {
  return (
    <View style={styles.container}>
      <Text style={styles.text}>Witaj w mojej aplikacji!</Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  text: {
    fontSize: 20,
    fontWeight: 'bold',
  },
});
```

Hook useState

`useState` to jeden z podstawowych hooków w React (i React Native), który pozwala na dodanie stanu do komponentów funkcyjnych. Jest to bardzo użyteczne narzędzie, które sprawia, że zarządzanie stanem w komponentach jest prostsze i bardziej czytelne.

Oto, jak używać `useState` w React Native:

1. Importowanie `useState`:

Na początek musisz zaimportować `useState` z pakietu `react`.

```
import React, { useState } from 'react';
```

2. Użycie `useState`:

`useState` przyjmuje jeden argument - początkową wartość stanu. Zwraca tablicę z dwoma elementami: aktualną wartością stanu i funkcją do aktualizacji tego stanu.

Przykład:

```
const [count, setCount] = useState(0);
```

W tym przypadku:

- `count` to zmienna reprezentująca aktualną wartość stanu.

- `setCount` to funkcja, która pozwala na aktualizację wartości `count`.

- `0` to początkowa wartość dla `count`.

3. Aktualizacja stanu:

Aby zaktualizować stan, użyj funkcji, która została zwrócona jako drugi element z `useState`.

Przykład:

```
setCount(count + 1);
```

Przykład użycia `useState` w komponencie React Native:

```
import React, { useState } from 'react';
import { View, Button, Text } from 'react-native';

const CounterApp = () => {
  const [count, setCount] = useState(0);

  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }
      <Text>{count}</Text>
      <Button title="Dodaj" onPress={() => setCount(count + 1)} />
      <Button title="Odejmij" onPress={() => setCount(count - 1)} />
    </View>
  );
}

export default CounterApp;
```

W powyższym przykładzie mamy prosty licznik. Za każdym razem, gdy naciśniesz przycisk "Dodaj", wartość `count` zostanie zwiększona o 1, a gdy naciśniesz "Odejmij", zostanie zmniejszona o 1.

To tylko podstawy `useState`. Możesz używać tego hooka do zarządzania różnymi typami danych, takimi jak obiekty, tablice itp. Ważne jest, aby pamiętać, że aktualizacje stanu są asynchroniczne, więc jeśli potrzebujesz bazować na poprzednim stanie, użyj funkcji jako argumentu dla funkcji aktualizującej stan.