

## Zadanie 1 „useContext”

Hook `useContext` w React służy do **pobierania wartości z kontekstu** (czyli z obiektu `Context` utworzonego przez `React.createContext`) **bez konieczności ręcznego przekazywania propsów przez kolejne komponenty (prop drilling)**.

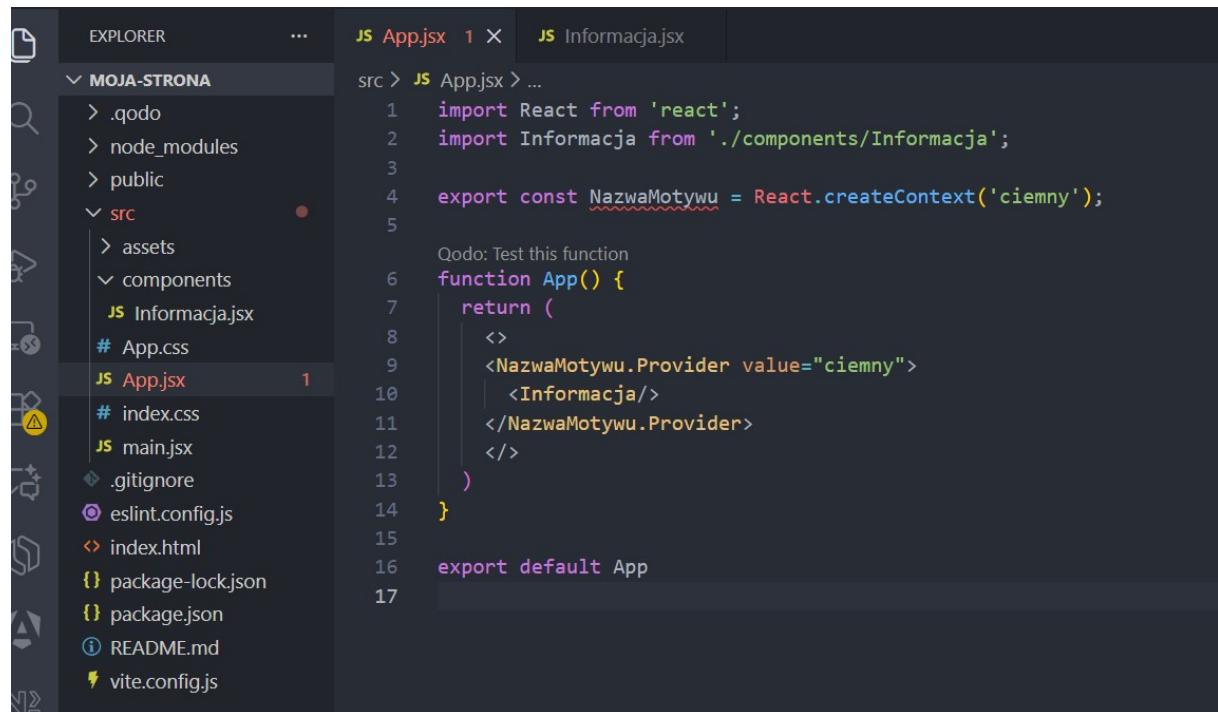
### Po co stosuje się `useContext`?

Kiedy kilka komponentów w różnych miejscach drzewa ma korzystać z tych samych danych (np. język aplikacji, motyw, dane użytkownika), zamiast przekazywać je przez propsy na każdej warstwie, używamy **Context API**.

Wykonaj Ćwiczenie:

Utwórz nowy projekt w React i nazwij go StarWars.

Projekt posiada jeden komponent „Informacja” który używa `useContext` utworzony w `App.jsx`



The screenshot shows a code editor interface with a sidebar labeled "EXPLORER" containing a file tree for a project named "MOJA-STRONA". The tree includes ".qodo", "node\_modules", "public", and a "src" folder which contains "assets", "components", "Informacja.jsx", "App.css", "App.jsx", "index.css", "main.jsx", ".gitignore", "eslint.config.js", "index.html", "package-lock.json", "package.json", "README.md", and "vite.config.js". The "App.jsx" file is currently selected and open in the main editor area. The code in "App.jsx" imports React and Informacja from their respective files. It defines a context "NazwaMotywu" using `React.createContext('ciemny')`. The `App()` component returns a provider `<NazwaMotywu.Provider value="ciemny">` containing an `<Informacja/>` component. The code ends with `export default App`. A tooltip "Qodo: Test this function" appears over the `function App() {` line. The "Informacja.jsx" file is also visible in the background, showing its definition of the `Informacja` component.

```
src > JS App.jsx > ...
1 import React from 'react';
2 import Informacja from './components/Informacja';
3
4 export const NazwaMotywu = React.createContext('ciemny');
5
6 Qodo: Test this function
7 function App() {
8     return (
9         <>
10            <NazwaMotywu.Provider value="ciemny">
11                <Informacja/>
12            </NazwaMotywu.Provider>
13        </>
14    )
15
16    export default App
17
```

```
src > components > JS Informacja.jsx > default
1 import { useContext } from 'react';
2 import { NazwaMotywu } from '../App';
3
4 Qodo: Test this function
5 function Informacja() {
6   const motyw = useContext(NazwaMotywu);
7   return(
8     <>
9       <div>Aktualny motyw to: {motyw === "ciemny" ? "Lord Sith" : "Jedi"}</div>
10    )
11 }
12 export default Informatcja;
```

## Zadanie 2 „useContext”

Utwórz nowy projekt StarWars2

Używamy obiektu a nie „initial value w createContext(‘ciemny’)

```
src > JS App.jsx > App
1 import React from 'react';
2 import Informatcja from './components/Informacja';
3
4 export const NazwaMotywu = React.createContext();
5
6 Qodo: Test this function
7 function App() {
8   return (
9     <>
10      <NazwaMotywu.Provider value={{stronamocy: "ciemna"}}>
11        <Informatcja/>
12      </NazwaMotywu.Provider>
13    )
14 }
15
16 export default App
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure under "MOJA-STRONA". Files include .qodo, node\_modules, public, src (with components, App.css, App.jsx, index.css, main.jsx, .gitignore, eslint.config.js, index.html), package-lock.json, package.json, README.md, and vite.config.js.
- EDITOR**: Two tabs are open: "JS App.jsx" and "JS Informacja.jsx". The "JS Informacja.jsx" tab is active, displaying the following code:

```
src > components > JS Informacja.jsx > Informacja
1 import { useContext } from 'react';
2 import { NazwaMotywu } from '../App';
3
4 Qodo: Test this function
5 function Informacja() {
6   const { stronamocy } = useContext(NazwaMotywu);
7   return(
8     <>
9       <div>Aktualny motyw to: {stronamocy === "ciemna" ? "Przemyśl swoje życie" : "Jesteś na dobrej ścieżce"}</div>
10    </>
11  )
12  export default Informacja;
```

### Zadanie 3

napisz aplikację w reakcje, która korzysta w całym motywie z określonego zestawu stylów:

tło czerwone, zielone, niebieskie  
odpowiednio kolor czcionki: biały, pomarańczowy, czarny  
Poniżej projekt:

3 komponenty: Header, Main, Footer

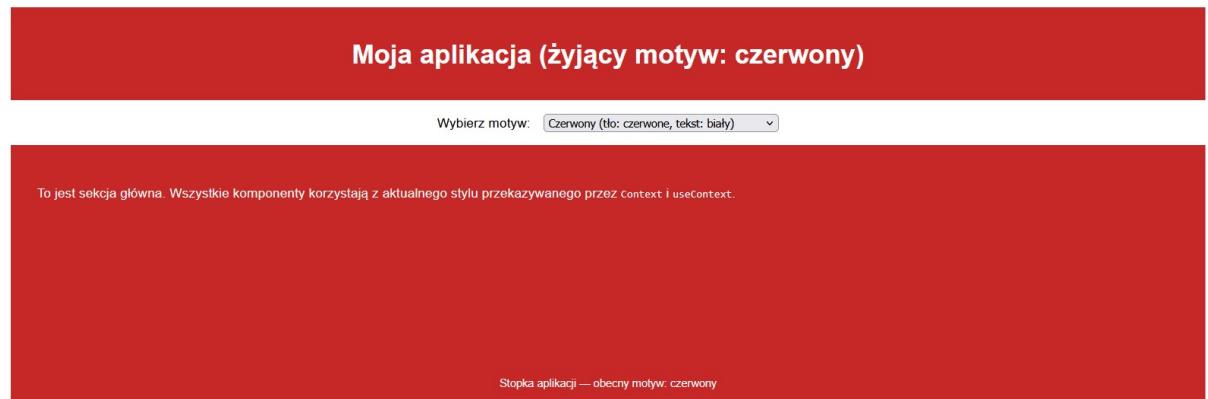
<Header/>

<Main/>

<Footer/>

Wszystkie komponenty korzystają z wybranego stylu na podstawie `useContext(styl)`

Główny komponent app posiada pole selekt który zmienia styl i przekazuje dalej przez providera do komponentów



## Moja aplikacja (żyjący motyw: zielony)

Wybierz motyw: Zielony (tło: zielone, tekst: pomarańczowy) ▾

To jest sekcja główna. Wszystkie komponenty korzystają z aktualnego stylu przekazywanego przez Context i useContext.

Stopka aplikacji — obecny motyw: zielony

The screenshot shows a web browser window with a blue header bar. The address bar displays "http://localhost:5173". The main content area has a dark blue background and features a title "Moja aplikacja (żyjący motyw: niebieski)". Below the title is a dropdown menu titled "Wybierz motyw:" with the following options: "Niebieski (tło: niebieskie, tekst: czarny)" (selected), "Czerwony (tło: czerwone, tekst: biały)", "Zielony (tło: zielone, tekst: pomarańczowy)", and "Niebieski (tło: niebieskie, tekst: czarny)". At the bottom of the page, there is a footer text "Stopka aplikacji — obecny motyw: niebieski". The browser's status bar at the bottom shows "11%" and "Zaloguj się".