Izrada projekta (10) - prop drilling, context API

Prop drilling

Kako znamo da React ima jednosmerni tok podataka, odnosno, da se podaci šalju od roditeljskih komponenti ka "deci", tradicionalni način prosleđivanja podataka kroz props može da postane nepodoban kada treba proslediti određeni podatak komponenti koja ja značajan broj nivoa ispod početne komponente, ili kada jednostavno treba proslediti podatak dosta različitih komponenti odjednom.

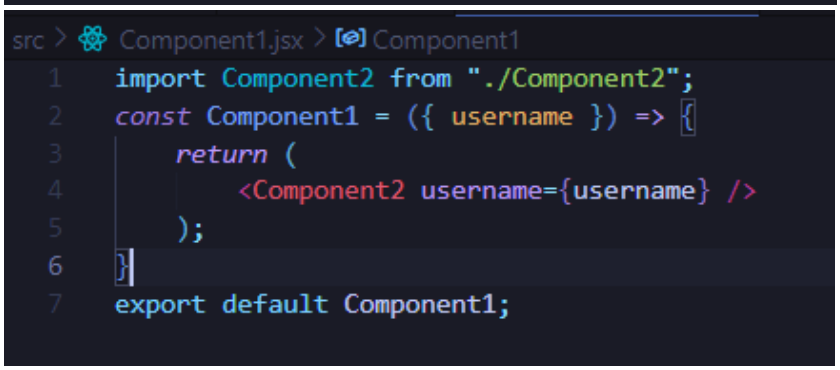
Context API

Context API omogućava komponenti da određenu vrednost jednostavno prosledi svim komponentama koje se nalaze "ispod nje" u stablu komponenti.

Korišćenje konteksta se svodi na:

1. Kreiranje konteksta.
2. Korišćenje konteksta u okviru komponente kojoj su potrebni podaci.
3. Pružanje konteksta od strane komponente u okviru koje su definisani podaci.

U nastavku možete da vidite kako smo u komponenti **App** definisali state promenljivu **username**, koju smo kroz komponentu **Component1** prosledili komponenti **Component2**.



```jsx
src > Component2.jsx > Component2
const Component2 = ({ username }) => {
    return (
        <h1>Username: {username}</h1>
    );
}
export default Component2;
```

Korak 1

U okviru **src** foldera ćemo napraviti novi fajl koji ćemo nazvati **UsernameContext.js.**

U okviru ovog fajla ćemo kodom u nastavku napraviti i export-ovati novi Context sa podrazumevanom vrednošću "" (prazan string) ,koji ćemo nazvati **UserNameContext**.

```jsx
src > UsernameContext.js > ...
import { createContext } from "react";
export const UsernameContext = createContext("");
```

Korak 2

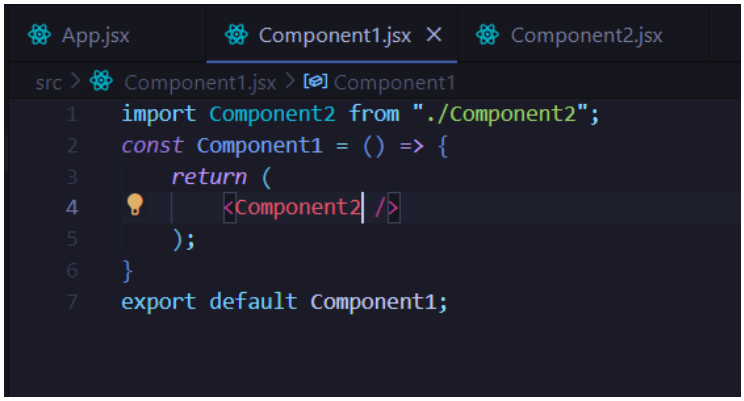
Sada kada smo napravili context, možemo da u našoj App komponenti komponentu Component1 "umotamo" u context provider kome ćemo kao vrednost postaviti vrednost username promenljive state-a. Ovime omogućavamo da sve komponente (i njihova deca) u okviru provider-a context-a dobiju pristup username promenljivoj state-a.

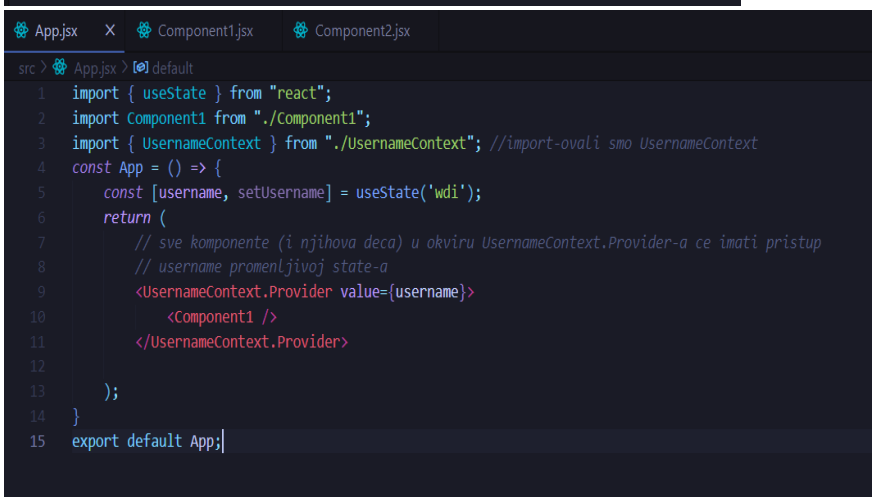
```jsx
App.jsx | Component1.jsx | Component2.jsx
src > App.jsx > App
import { useState } from "react";
import Component1 from "./Component1";
import { UsernameContext } from "./UsernameContext"; //import-ovali smo UsernameContext
const App = () => {
    const [username, setUsername] = useState('wdi');
    return (
        // sve komponente (i njihova deca) u okviru UsernameContext.Provider-a ce imati pristup
        // username promenljivoj state-a
        <UsernameContext.Provider value={username}>
            <Component1 username={username} />
        </UsernameContext.Provider>

    );
}
export default App;
```

Korak 3

Najzad, u komponenti Component2 možemo iskoristiti Hook useContext(), kome ćemo kao argument proslediti UsernameContext, čime ćemo lokalnoj promenljivoj username dodeliti vrednost UsernameContext-a (što je zapravo username promenljiva state-a komponente App). Naravno, sada možemo obrisati parametar username, jer username više ne dobijamo kao prop, već iz context-a.

Takođe, možemo obrisati username prop svuda gde se pojavljuje u komponenti Component1, kao i u komponenti App.





Ako sada pogledamo rezultat pokretanja našeg React koda, vidimo da sve još uvek radi kao i ranije.

Username: wdi