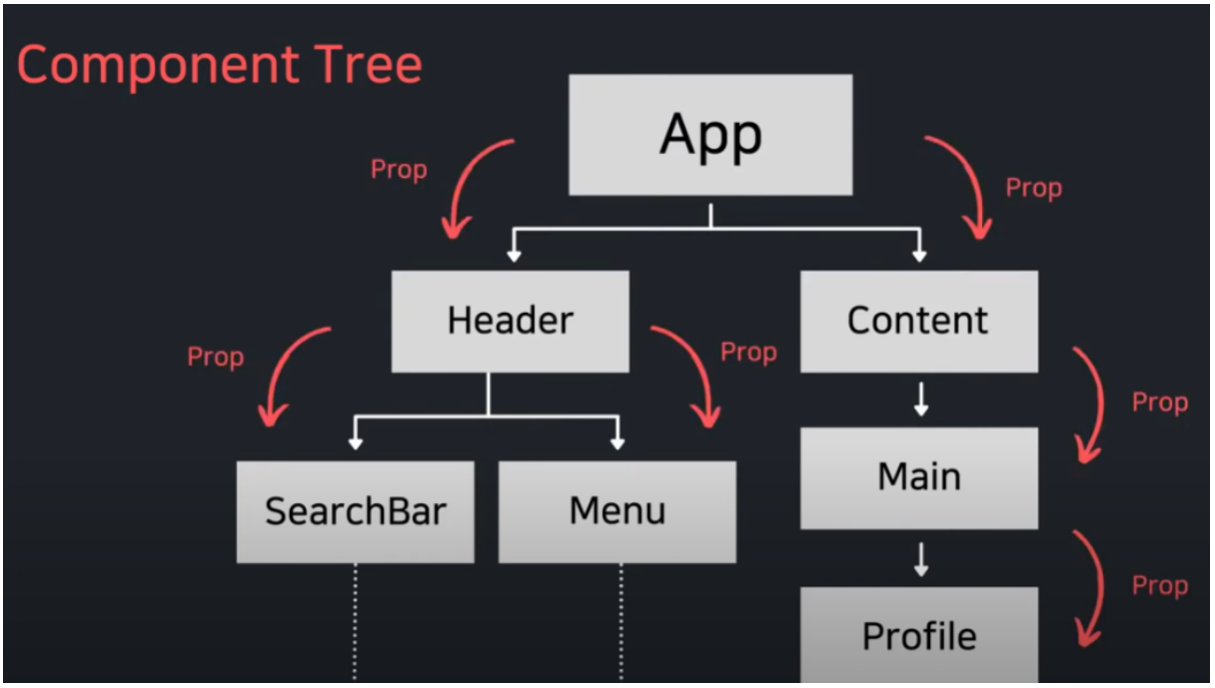
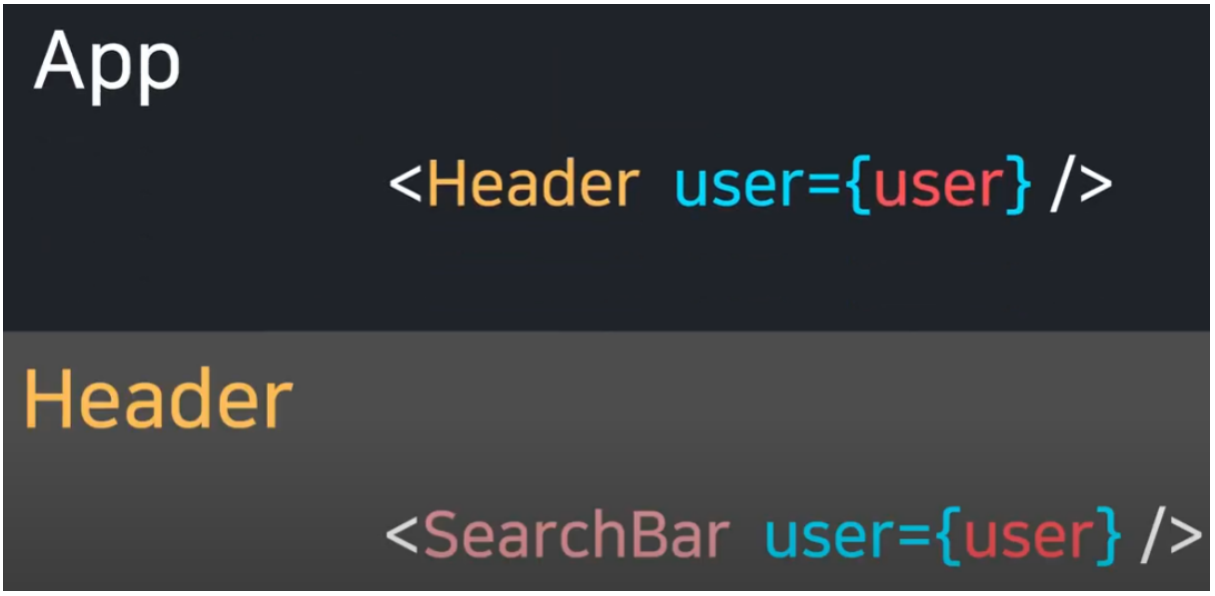
useContext 훅

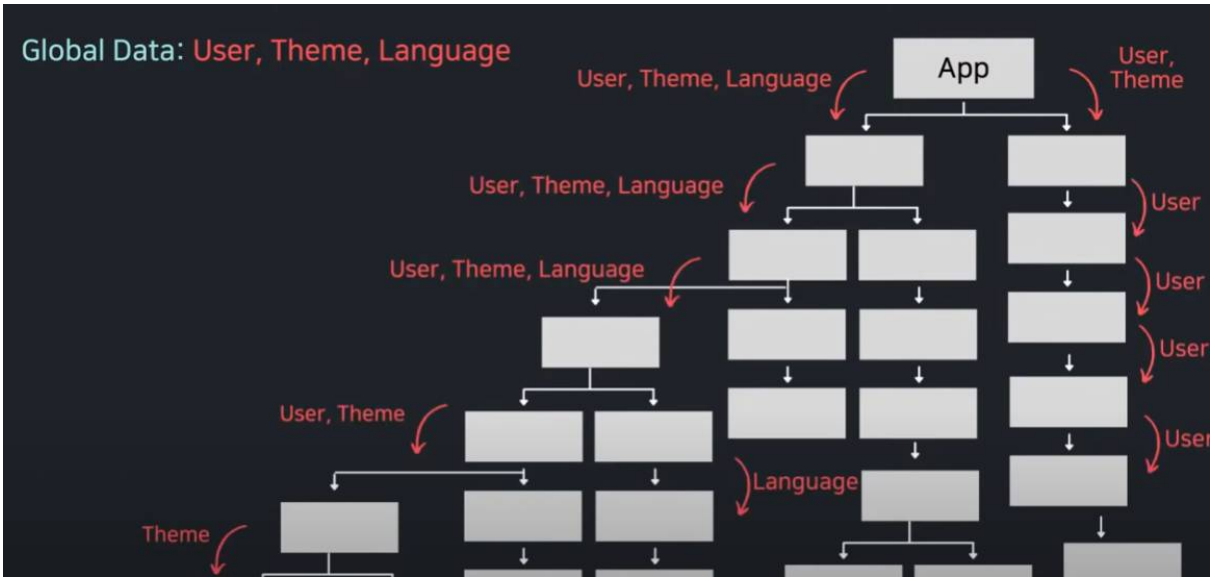
리액트에 일반적인 데이터의 흐름은 위에서 아래로 즉 부모 컴포넌트에서 자식 컴포넌트로 props를 통해서 전달이 됩니다.



props를 통해 데이터를 전달하려면 부모 컴포넌트가 자식 컴포넌트에게 일일이 단계별로 전달을 해줘야 합니다.



아래와 같이 큰 리액트 앱이 있다고 가정해 봅시다

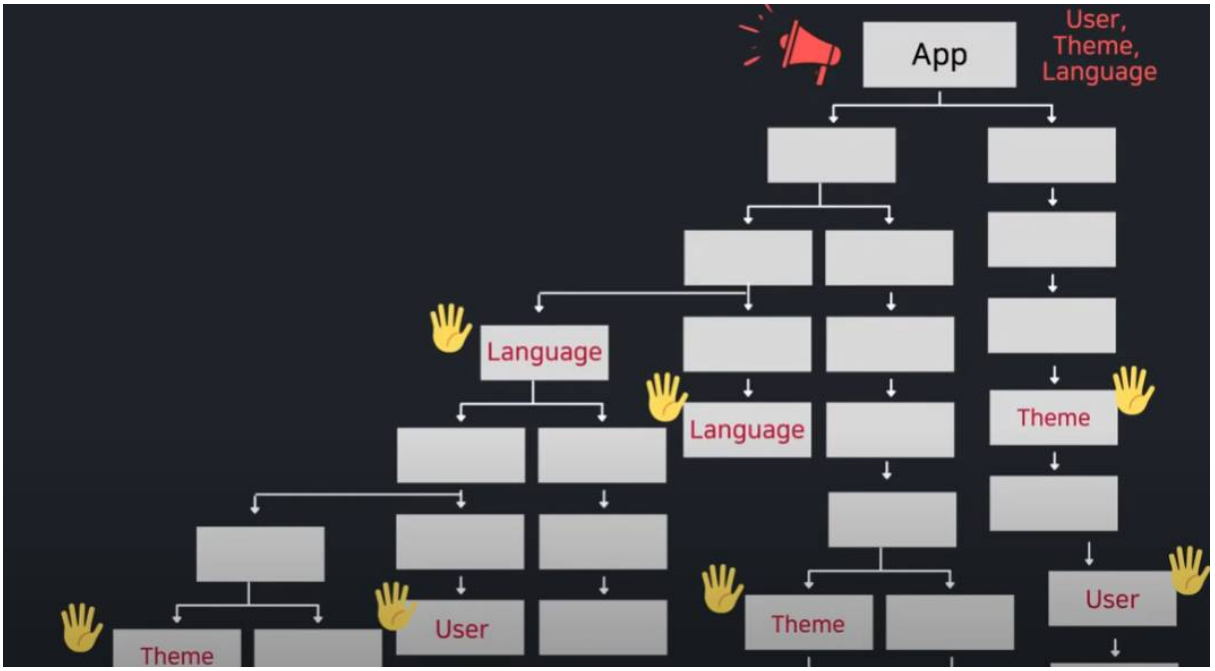


- 앱 내부에서 수많은 컴포넌트 들이 공통적으로 전역적인 데이터가 필요한 상황입니다. (로그인 된 사용자 정보, 테마, 언어 등등...)
- 이런 전역적인 데이터를 props로 일일이 단계별로 전달을 해야 한다면, 코드가 복잡해지고 수정을 하더라도 컴포넌트들을 찾아다니면서 수정을 해야 될 겁니다.
- 리액트는 이러한 문제점을 간편하게 해결해 주는 context API라는 것을 제공해 줍니다.

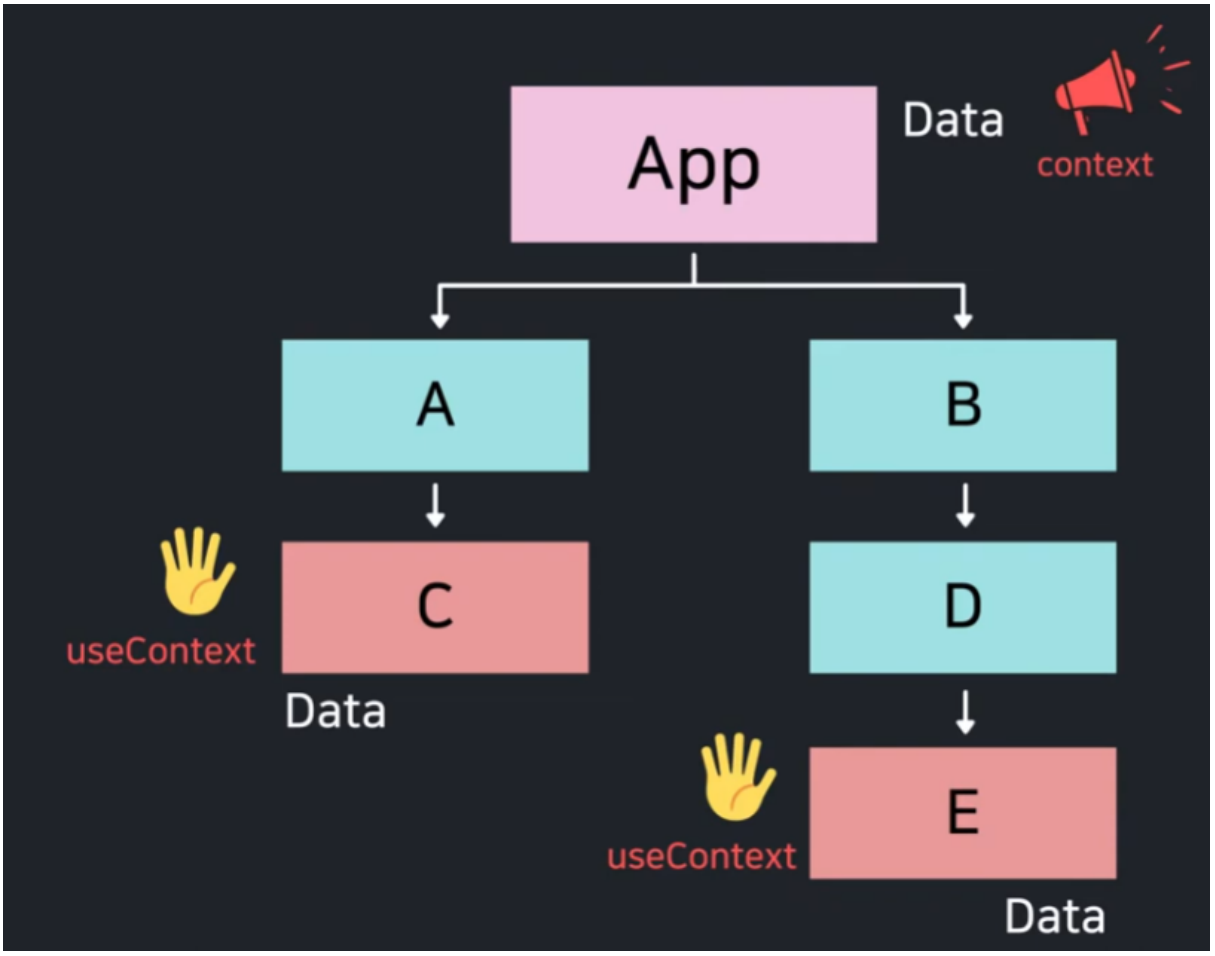
context API

> context는 앱 안에서 전역적으로 사용되는 데이터들을 여러 컴퍼넌트들이 쉽게 공유할 수 있는 방법을 제공해 줍니다.

> 해당 데이터를 갖고 있는 상위 컴포넌트가 테이터가 필요한 하위 컴포넌트들에게 다이렉트로 전달해 줄 수 있습니다.



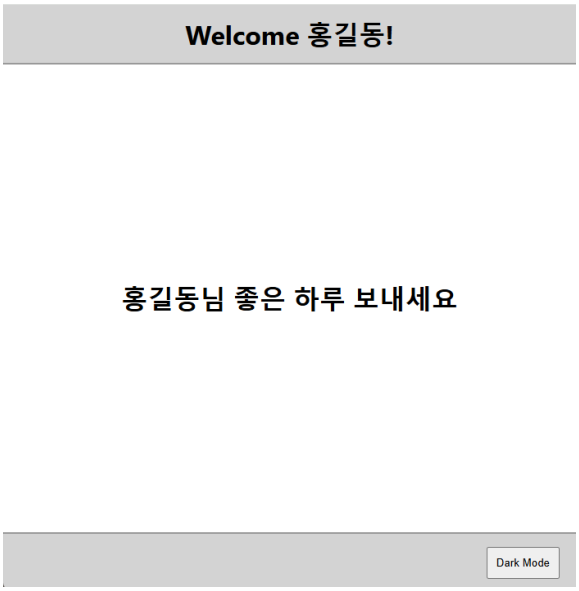
props대신 context를 사용하여 데이터를 공유하면, 그 데이터를 사용하고 싶은 컴포넌트들은 useContext훅을 사용해서 받아 오기만 하면 됩니다.

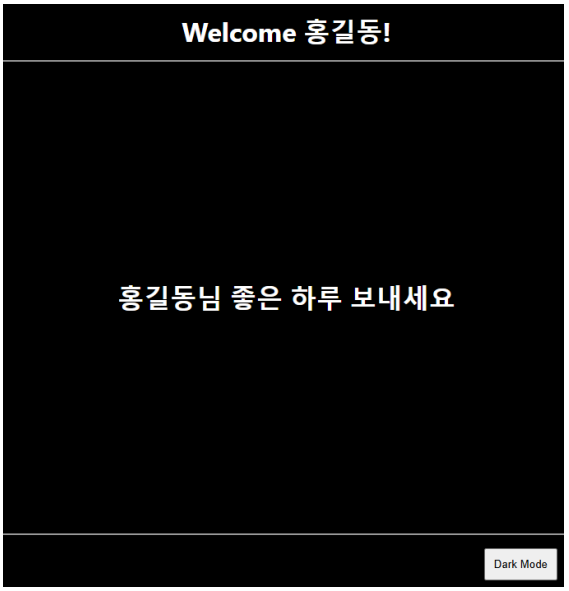


Context는 꼭 필요할때만!

- Context를 사용하면 컴포넌트를 재사용하기 어려워질 수 있다.

버튼을 클릭하면 다크 모드로 변경되는 프로젝트를 만들어 보





App.css

```css
*{margin:0; padding:0;}

.header{
  height:10vh;
  display:flex;
  align-items: center;
  justify-content: center;
  border-bottom:2px solid #999;
}

.content{
  height:80vh;
  display:flex;
  align-items: center;
  justify-content: center;
}

.footer{
  height:10vh;
  display:flex;
  align-items: center;
```

```
  justify-content: flex-end;
  border-top:2px solid #999;
}
.footer button{
  margin-right:20px;
  padding:10px;
}
```

App.js

```
import {useState} from 'react';
import './App.css';
import Page from './components/Page';

function App() {
    //현재 앱이 다크모드인지 아닌지 true, false로 제어
    const [isDark, setIsDark] = useState(false);
    //Page 컴포넌트로 isDark와 setIsDark를 props로 전달
    return <Page isDark={isDark} setIsDark={setIsDark} />
}

export default App;
```

components/Page.js

```
import React from 'react'
import Header from './Header'
import Content from './Content'
import Footer from './Footer'

//App컴포넌트로 부터 isDark와 setIsDark를 props로 전달 받음
const Page = ({isDark, setIsDark}) => {
    //Page 컴포넌트는 isDark와 seIsDark를 사용하지 않는다.
  //Header, Content, Footer 컴포넌트로 isDark와 setIsDark를 pr
ops로 전달
  return (
    <div className='page'>
```

```
      <Header isDark={isDark}/>
      <Content isDark={isDark}/>
      <Footer isDark={isDark} setIsDark={setIsDark}/>
    </div>
  )
}

export default Page
```

Header.js

```
import React from 'react'
//Page컴포넌트로 부터 isDark를 props로 전달받음
const Header = ({isDark}) => {
  //isDark가 true면 배경색을 검은색, 글자색을 흰색으로 적용
  //false면 배경색을 밝은 회색에 글자색을 검은색으로 적
  return (
    <header
      className='header'
      style={{
        backgroundColor: isDark ? 'black' : 'lightgray',
        color: isDark ? 'white' : 'black'
      }}
    >
      <h1>Welcome 홍길동!</h1>
    </header>
  )
}

export default Header
```

Content.js

```
import React from 'react'
//Header 컴포넌트와 동일
const Content = ({isDark}) => {
  return (
```

```
    <div
      className='content'
      style={{
        backgroundColor: isDark ? 'black' : 'lightgray',
        color: isDark ? 'white' : 'black'
      }}
    >
      <h1>홍길동님 좋은 하루 보내세요</h1>
    </div>
  )
}

export default Content
```

Footer.js

```
import React from 'react'
//footer의 경우 설정함수도 props로 받음
const Footer = ({isDark, setIsDark}) => {
  //함수가 호출되면 setIsDark(설정함수)를 실행하고 인자로 isDark의
결과의 반대값을 보내줍니다. (true -> false, false -> true)
  const toggleTheme = () => {
    setIsDark(!isDark);
  }
  return (
    <footer
      className='footer'
      style={{
        backgroundColor: isDark ? 'black' : 'lightgray',
      }}
    >
    {/*버튼을 클릭하면 toggleTheme함수 실행*/}
      <button className='button' onClick={toggleTheme}>
        Dark Mode
      </button>
    </footer>
  )
}
```

```
export default Footer
```

---

useContext 훅 사용

context디렉터리를 만들고 ThemeContext컴포넌트를 생성

context/ThemeContext.js

```
//context를 사용하기 위해 createContext를 임포트
import {createContext} from 'react';

//ThemeContext 컴포넌트에 createContext로 컨텍스트를 만들고 기본값을 null로 설정
export const ThemeContext = createContext(null);
```

App컴포넌트에 ThemeContext를 임포트

App.js

```
import {useState} from 'react';
import './App.css';
import Page from './components/Page';
import { ThemeContext } from './context/ThemeContext';

function App() {
    const [isDark, setIsDark] = useState(false);

    //Page컴포넌트를 ThemeContext.provider로 감싸 줍니다.
    //ThemeContext.Provider는 value라는 prop을 하나 받습니다.
    //value 안에는 전달하고자 하는 데이터를 넣으면 됩니다.
    return (
        <ThemeContext.Provider value={{isDark, setIsDark}}>
            {/*이제 props로 전달하지 않아도 됩니다(삭제)*/}
```

```
            <Page isDark={isDark} setIsDark={setIsDark} />
        </ThemeContext.Provider>
    )
}

export default App;
```

💡 **위와 같이 작성하면 ThemeContext.Provider로 감싼 모든 하위 요소는 props를 사용하지 않고 isDark와 setIsDark에 접근할 수 있게 됩니다.**

components/Page.js

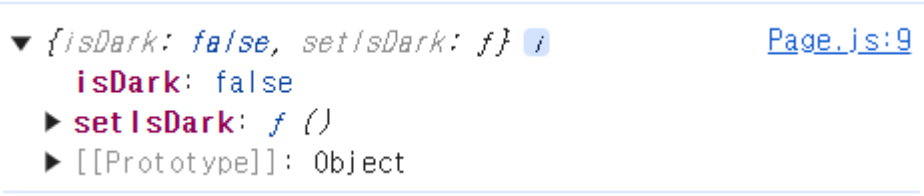
```
import React from 'react'
import Header from './Header'
import Content from './Content'
import Footer from './Footer'
//(확인하고 삭제)ThemeContext 임포트
import { ThemeContext } from '../context/ThemeContext'

//props로 전달하는 부분 삭제
const Page = ({isDark, setIsDark}) => {
    //(확인하고 삭제)useContext로 ThemeContext에서 전달받은 prop을 data에 담아 줍니다
    const data = useContext(ThemeContext);
    console.log(data)

  return (
    <div className='page'>
      <Header isDark={isDark}/>
      <Content isDark={isDark}/>
      <Footer isDark={isDark} setIsDark={setIsDark}/>
    </div>
  )
}
```

```
export default Page
```

전달 받은 isDark와 setIsDark가 들어 있는 것을 확인할 수 있음



Header.js

```
//useContext 임포트
import React, { useContext } from 'react'
//ThemeContext 임포트
import { ThemeContext } from '../context/ThemeContext';

//props로 전달받은 isDark삭제
const Header = ({isDark}) => {
    //useContext로 ThemeContext에서 전달받은 isDark를 변수에 담아
활
  const {isDark} = useContext(ThemeContext);
  console.log(isDark)

  return (
    <header
      className='header'
      style={{
        backgroundColor: isDark ? 'black' : 'lightgray',
        color: isDark ? 'white' : 'black'
      }}
    >
      <h1>Welcome 홍길동!</h1>
    </header>
  )
}
```

```
export default Header
```

Content.js

```
import React, { useContext } from 'react'
import { ThemeContext } from '../context/ThemeContext';

const Content = ({isDark}) => {
    const {isDark} = useContext(ThemeContext);
  return (
    <div
      className='content'
      style={{
        backgroundColor: isDark ? 'black' : 'lightgray',
        color: isDark ? 'white' : 'black'
      }}
    >
      <h1>홍길동님 좋은 하루 보내세요</h1>
    </div>
  )
}

export default Content
```

Footer.js

```
import React, { useContext } from 'react'
import { ThemeContext } from '../context/ThemeContext';

const Footer = ({isDark, setIsDark}) => {
    //footer의 경우 setIsDark도 전달한다.
    const {isDark, setIsDark} = useContext(ThemeContext);

  const toggleTheme = () => {
    setIsDark(!isDark);
  }
```

```
  return (
    <footer
      className='footer'
      style={{
        backgroundColor: isDark ? 'black' : 'lightgray',
      }}
    >
      <button className='button' onClick={toggleTheme}>
        Dark Mode
      </button>
    </footer>
  )
}

export default Footer
```

연습 문제

'홍길동'이라고 되어 있는 부분을 context를 활용하여 '사용자' 라는 문자로 표시하시오

1. UserContext.js 파일을 생성하고 createContext로 초기값이 null인 context를 생성
2. App.js에서 UserContext를 임포트하고 요소를 UserContext.Provider로 감싸주고 value값을 "사용자"로 합니다.
3. Header컴포넌트의 "홍길동"을 "사용자"로 변경하고 content컴포넌트의 "홍길동"도 "사용자"로 변경해 줍니다.

답