createContext

`createContext`를 사용하면 컴포넌트가 Context를 제공하거나 읽을 수 있습니다.

```
const SomeContext = createContext(defaultValue)
```

레퍼런스

`createContext(defaultValue)`

컴포넌트 외부에서 `createContext`를 호출하여 컨텍스트를 생성합니다.

```
import { createContext } from 'react';

const ThemeContext = createContext('light');
```

아래 예시를 참고하세요.

매개변수

- `defaultValue` : 컴포넌트가 컨텍스트를 읽을 때 상위에 일치하는 컨텍스트 제공자가 없는 경우 컨텍스트가 가져야 할 값입니다. 의미 있는 기본값이 없으면 `null` 을 지정하세요. 기본값은 “최후의 수단”으로 사용됩니다. 이 값은 정적이며 시간이 지나도 변경되지 않습니다.

반환값

`createContext` 는 컨텍스트 객체를 반환합니다.

컨텍스트 객체 자체는 어떠한 정보도 가지고 있지 않습니다. 다른 컴포넌트가 읽거나 제공하는 어떤 컨텍스트를 나타냅니다. 일반적으로 상위 컴포넌트에서 컨텍스트 값을 지정하기 위해 `SomeContext` 를 사용하고, 아래 컴포넌트에서 읽기 위해 `useContext(SomeContext)` 를 호출합니다. 컨텍스트 객체에는 몇 가지 속성이 있습니다.

- `SomeContext` 는 컴포넌트에게 컨텍스트 값을 제공합니다.
- `SomeContext.Consumer` 는 컨텍스트 값을 읽는 대안이며 드물게 사용됩니다.
- `SomeContext.Provider` 는 React 19 이전 버전에서 사용되는 오래된 컨텍스트 값 제공 방법입니다.

`SomeContext` Provider

컴포넌트를 컨텍스트 제공자Provider로 감싸서 이 컨텍스트의 값을 모든 내부 컴포넌트에 지정합니다.

```
function App() {
  const [theme, setTheme] = useState('light');
  // ...
  return (
    <ThemeContext value={theme}>
      <Page />
    </ThemeContext>
  );
}
```

중요합니다!

React 19부터는 `<SomeContext>` 를 제공자Provider로 렌더링 할 수 있습니다.

오래된 React 버전은 `<SomeContext.Provider>` 를 사용합니다.

Props

- `value` : 이 제공자 내부의 컨텍스트를 읽는 모든 컴포넌트에 전달하려는 값입니다. 컨텍스트 값은 어떠한 유형이든 될 수 있습니다. 제공자 내부에서 `useContext(SomeContext)` 를 호출하는 컴포넌트는 그 위의 가장 가까운 해당 컨텍스트 제공자의 `value` 를 받게 됩니다.

`SomeContext.Consumer`

`useContext` 가 등장하기 전에 컨텍스트를 읽는 이전 방식이 있었습니다.

```
function Button() {
  // 🟡 이전 방식 (권장하지 않음)
  return (
    <ThemeContext.Consumer>
      {theme => (
        <button className={theme} />
      )}
    </ThemeContext.Consumer>
  );
}
```

이 예전 방식은 여전히 작동하지만, **새로 작성된 코드는 대신 `useContext()` 로 컨텍스트를 읽어야 합니다.**

```
function Button() {
  // ✅ 권장하는 방법
  const theme = useContext(ThemeContext);
```

```
  return <button className={theme} />;
}
```

Props

- `children` : 함수입니다. React는 `useContext()` 와 동일한 알고리즘으로 결정된 현재 컨텍스트 값을 전달하여 함수를 호출하고, 이 함수에서 반환하는 결과를 렌더링합니다. 부모 컴포넌트에서 컨텍스트가 변경되면 React는 이 함수를 다시 실행하고 UI를 업데이트합니다.

사용법

컨텍스트 생성

컨텍스트를 사용하면 컴포넌트가 정보를 깊게 전달할 수 있습니다.

컴포넌트 외부에서 `createContext` 를 호출하여 하나 이상의 컨텍스트를 생성합니다.

```
import { createContext } from 'react';

const ThemeContext = createContext('light');
const AuthContext = createContext(null);
```

`createContext` 는 컨텍스트 객체 를 반환합니다. 컴포넌트는 이를 `useContext()` 에 전달하여 컨텍스트를 읽을 수 있습니다.

```
function Button() {
  const theme = useContext(ThemeContext);
  // ...
}

function Profile() {
  const currentUser = useContext(AuthContext);
  // ...
}
```

기본적으로, 그들이 받는 값은 컨텍스트를 생성할 때 지정한 기본값 이 됩니다. 그러나 자체적으로 이는 유용하지 않습니다. 왜냐하면 기본값은 절대 변경되지 않기 때문입니다.

컨텍스트는 **다른 동적 값들을 컴포넌트에서 제공**할 수 있기 때문에 유용합니다.

```
function App() {
  const [theme, setTheme] = useState('dark');
  const [currentUser, setCurrentUser] = useState({ name: 'Taylor' });

  // ...

  return (
    <ThemeContext value={theme}>
      <AuthContext value={currentUser}>
        <Page />
      </AuthContext>
    </ThemeContext>
  );
}
```

이제 `Page` 컴포넌트와 그 안의 모든 컴포넌트, 얼마나 깊든지 간에 전달된 컨텍스트 값을 "볼" 수 있습니다. 전달된 컨텍스트 값이 변경되면, React는 컨텍스트를 읽는 컴포넌트를 다시 렌더링합니다.

컨텍스트를 읽고 제공하는 방법에 대해 더 알아보고 예시를 확인하세요.

파일에서 컨텍스트 가져오기 및 내보내기

종종 서로 다른 파일에 있는 컴포넌트들이 동일한 컨텍스트에 접근해야 할 때가 있습니다. 이것이 별도의 파일에서 컨텍스트를 선언하는 것이 일반적인 이유입니다. 그런 다음 `export` 문을 사용하여 다른 파일에서 사용할 수 있도록 컨텍스트를 사용할 수 있습니다.

```
// Contexts.js
import { createContext } from 'react';

export const ThemeContext = createContext('light');
```

다른 파일에서 선언된 컴포넌트는 이 컨텍스트를 읽거나 제공하기 위해 `import` 문을 사용할 수 있습니다.

```
// Button.js
import { ThemeContext } from './Contexts.js';

function Button() {
  const theme = useContext(ThemeContext);
  // ...
}
```

```
// App.js
import { ThemeContext, AuthContext } from './Contexts.js';

function App() {
  // ...
  return (
    <ThemeContext value={theme}>
      <AuthContext value={currentUser}>
        <Page />
      </AuthContext>
    </ThemeContext>
  );
}
```

이것은 컴포넌트 `import` 및 `export` 하기와 유사하게 동작합니다.

문제 해결

컨텍스트 값을 바꾸는 방법을 모르겠어요

이런 코드는 *기본* 컨텍스트 값을 지정합니다.

```
const ThemeContext = createContext('light');
```

이 값은 절대 변경되지 않습니다. React는 상위에 일치하는 제공자를 찾을 수 없는 경우에만 이 값을 기본값으로 사용합니다.

컨텍스트가 시간에 따라 변경되도록 만들려면, State를 추가하고 컴포넌트를 컨텍스트 제공자로 감싸세요.

이전
captureOwnerStack

다음
lazy

Meta Open Source

uwu?

React 학습하기

- 빠르게 시작하기
- 설치하기
- UI 표현하기
- 상호작용성 더하기
- State 관리하기
- 탈출구

API 참고서

- React APIs
- React DOM APIs

커뮤니티

- 행동 강령
- 팀 소개
- 문서 기여자
- 감사의 말

더 보기

- 블로그
- React Native
- 개인 정보 보호
- 약관