React

Context

Enrique Barra Arias
Álvaro Alonso González

Introducción

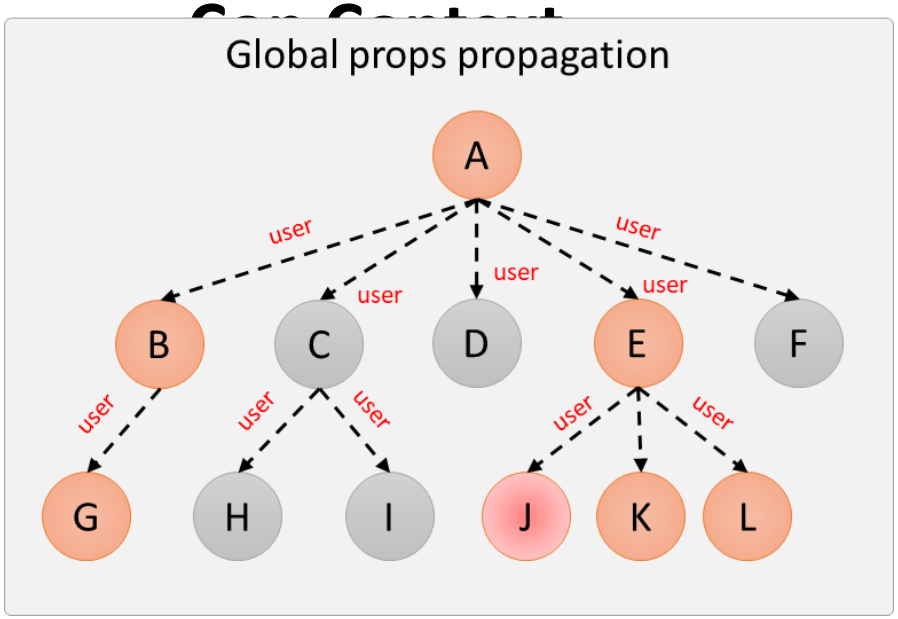
- Hay ocasiones en que ciertos tipos de props son necesarias para muchos componentes en una aplicación o incluso para todos
- Esto supone un “problema” que se llama “props drilling”

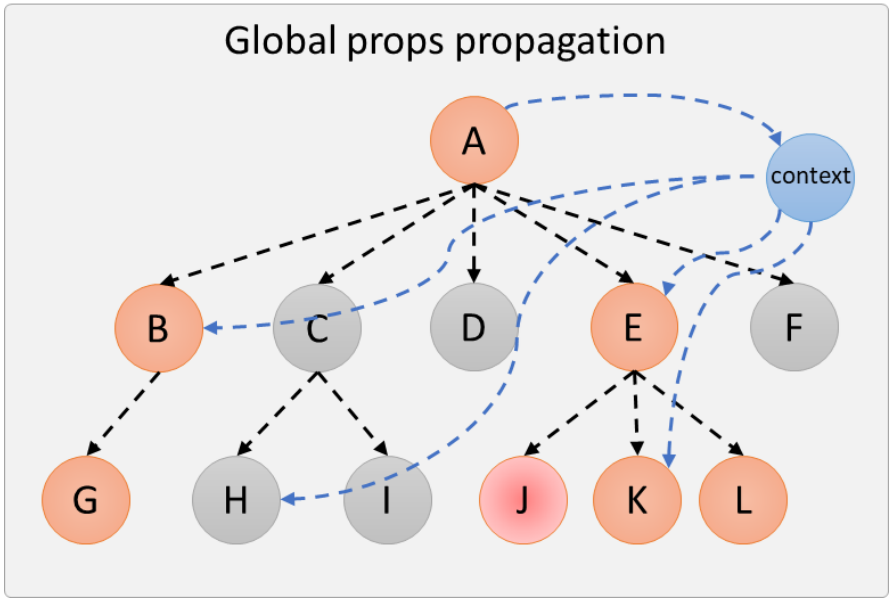
- Context provee una forma de pasar datos a través del árbol de componentes sin tener que pasar props manualmente en cada nivel
- Equivalente a variables globales en un programa
- Ejemplos:
  - El objeto user que está autenticado
  - Theme
  - Idioma
  - Currency (moneda)

- Más info y ejemplos: https://es.reactjs.org/docs/context.html

Context - motivación

Sin Context





Context - API

- Primero Creamos el Contexto (habrá que exportarlo para acceder a él en los hijos:

```
export const MyContext = React.createContext(defaultValue);
```

- Ahora usamos un Provider y le pasamos el valor del contexto

```
<MyContext.Provider value={/* algún valor */}>
/*Mi árbol de componentes*/
</MyContext.Provider>
```

Context - Consumer

- Usaremos el hook useContext (tenemos que importarlo e importar el contexto para que funcione).

```
function ThemedButton() {
  const theme = useContext(ThemeContext);
  return (
    <button style={{ background: theme.background, color: theme.foreground }}>
      I am styled by theme context!
    </button>
  );
}
```

Ejemplo completo

Repositorio del buscador de usuarios, internacionalizado en la rama context:
https://github.com/REACT-UPM/ejemplo_buscador_usuarios/tree/context

Paso 1: Crear LanguageProvider

Ver fichero en el repo porque por encima tiene las traducciones y los imports correspondientes

```
export function LanguageProvider(props) {
  const [lang, setLang] = useState("en");

  function switchLang(newLang){
    console.log("Vamos a cambiar el idioma a:", newLang);
    setLang(newLang);
  }

  const context = {
    language: lang,
    strings: langData[lang],
    switchLang: switchLang,
  };

  return (
    <LanguageContext.Provider value={context}>
      {props.children}
    </LanguageContext.Provider>
  );
};
```

Paso 2: Usar el LanguageProvider

- Usaremos el proveedor de contexto LanguageProvider que debe ser el padre de todo lo que queramos que tenga ese contexto. Eso lo podemos hacer en index.js

```
import { LanguageProvider } from "./LanguageProvider";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <LanguageProvider>
    <App />
  </LanguageProvider>
);
```

Paso 3 - usar el contexto (consumer)

- En cualquier componente hijo (da igual la profundidad podría estar hundido en el árbol bajo otros 8 componentes en la jerarquía) podré usar el hook useContext y obtengo lo que ha exportado el LanguageProvider

```
import { LanguageContext } from "./LanguageProvider";
import { useContext } from "react";

export default function Tarjeta(props){
  const langContext = useContext(LanguageContext);

  return(<li key={props.item.id}>
    <p>{langContext.strings.name} <b>{props.item.firstName}</b> {props.item.la
    <p>{langContext.strings.email} {props.item.email}</p>
    <p><img src={props.item.image} alt="Imagen de {props.item.firstName}"/></p
  </li>)
}
```

Paso 3 - usar el contexto (consumer)

- Si queremos hacer un selector de idioma para cambiar el idioma.
- App.js cambios:

```
import { LanguageContext } from "./LanguageProvider";
```

```
const langContext = useContext(LanguageContext);
```

```
const cambiaLang = (newlang)=>{
  langContext.switchLang(newlang);
}
```

```
<a href="#" onClick={()=>cambiaLang("en")}>en</a>
/
<a href="#" onClick={()=>cambiaLang("es")}>es</a>
```