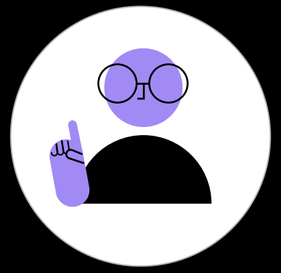
DIGITALHOUSE

Context en acción

Certified Tech Developer
The Ultimate Degree

Índice

01 createContext



Pasos para la implementación de **Context**

01 Creación mediante el método **createContext**

02 Poner cualquier valor en el Provider usando la propiedad value

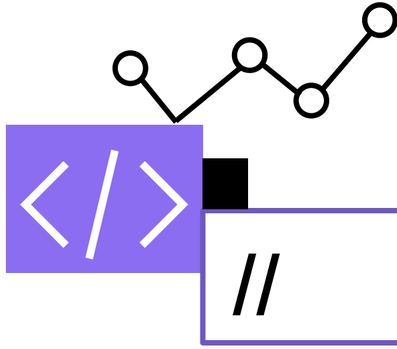
03 Usar un context Provider para encapsular un árbol de componentes.

04 Leer esta propiedad en cualquier componente encapsulado usando el hook **useContext()**

createContext

Un objeto Context se crea pasando por defecto un valor al método de React, **createContext**. Si bien createContext acepta un valor inicial, este no es obligatorio.

Como resultado de crear el contexto, obtendremos dos propiedades que utilizaremos luego: **Provider y Consumer.**



Ejemplo de createContext

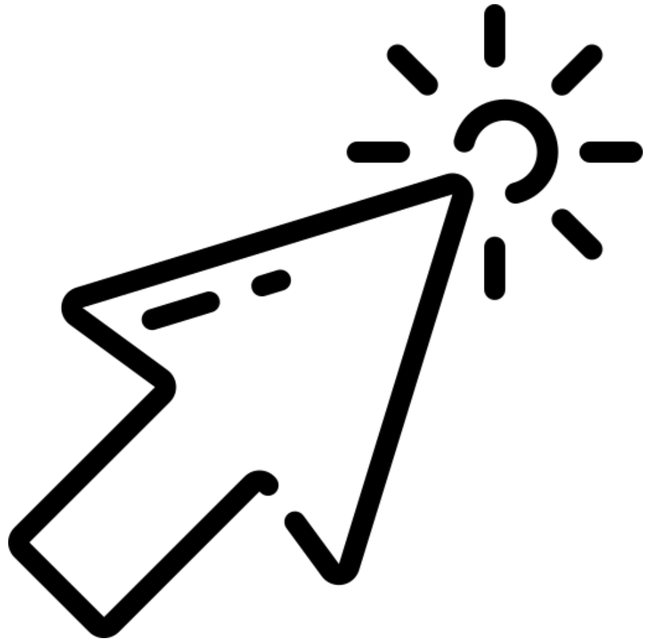
```
{}
import { createContext } from "react";

const defaultTitle = "Valor no obligatorio";

export const TitleContext = createContext(defaultTitle);
```

02 Provider

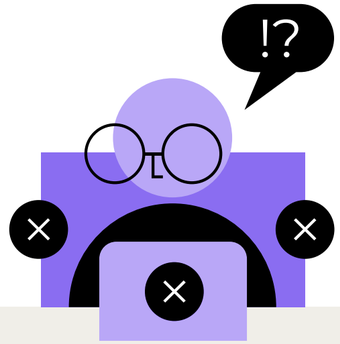
Provider

El contexto debe ser global para los componentes que lo consumen, es por esto por lo que Context nos brinda un **Provider.**

Este es un componente que toma una propiedad llamada value, es decir, el valor dado al contexto, y lo inyecta a los componentes englobados por él. Este valor puede ser el que se desee, hasta un objeto con múltiples valores.

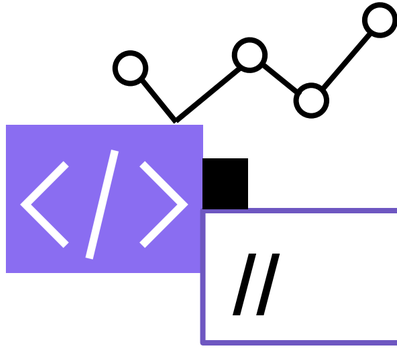


Provider



No todos los componentes descendientes del contexto estarán suscritos al contexto, salvo que declaren la suscripción al contexto. Esto es debido a que no todos los componentes necesitarán estos datos.

Cuando un componente suscrito se renderice, leerá el valor actual del contexto provisto por el Provider, por lo que la responsabilidad del Provider será dar acceso a los datos almacenados en el contexto.



Provider aplicado a la problemática inicial

```
{}
```

```
export const TitleContext = createContext(defaultTitle);

function App() {
  return (
    <TitleContext.Provider value="Soy un título">
      <Left/>
      <Main/>
    </TitleContext.Provider>
  );
}
```

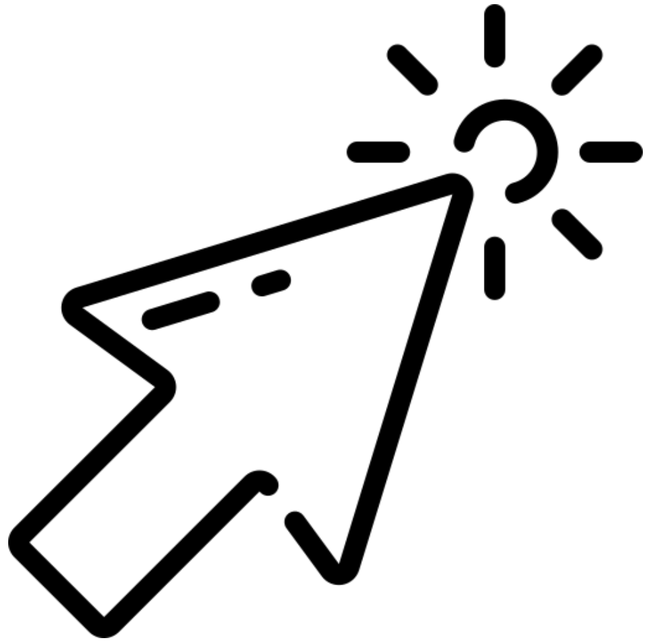
De este modo, ambos subárboles de componentes (Left y Main) podrán consumir el contexto TitleContext.

¿Qué vemos en React Dev Tools?

03 Consumer

Consumer Component vs useContext hook

Para que un componente pueda recibir el valor de un contexto, el componente debe estar suscrito al mismo. La suscripción se puede realizar mediante dos opciones: el **Consumer Component** o el hook **useContext().**



Consumer Component

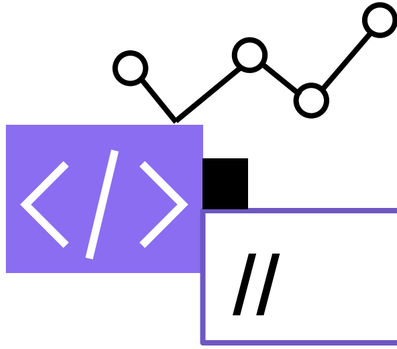
01 Este componente deberá incluir una función como hijo o children.

02 Esta función recibirá el argumento value con el valor actual del contexto, retornará un nodo React (JSX) y estará a la escucha de cambios.

03 Debemos pensar a Context como un canal para compartir datos.

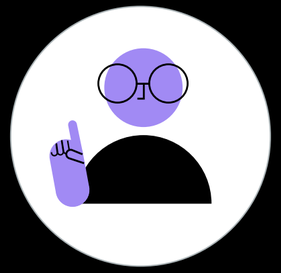
04 Respecto al árbol de componentes, el Consumer estará debajo del Provider. Esta función será llamada en el momento de renderizar.

05 Por cada Provider habrá un Consumer que sólo se relacionarán entre sí.



Consumer Component

```
{}
import { TitleContext } from './App'

const Main = ()=> {
      return (
        <TitleContext.Consumer>
            {(props) => {
               return <h1>{props.title}</h1>;
            }}
         </TitleContext.Consumer>
      )
}
```



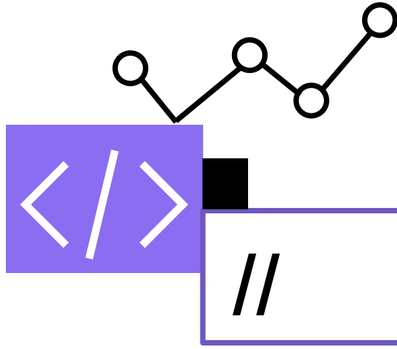
useContext()

01 Para poder consumir la información del contexto deberemos llamar a este hook dentro del componente que la requiera.

02 Como primer y único argumento recibe el objeto del contexto al que debe linkear y devuelve su valor actual.

03 Ten en cuenta que un componente que llama a useContext siempre se volverá a renderizar cuando el valor del contexto cambie.

04 En comparación con el Consumer Component aporta Mayor legibilidad en el código.



useContext

```
{}
```

```
import { useContext } from 'react'
import { TitleContext } from './App'

const Main = () => {
  const title = useContext(TitleContext)

  return (
    <div>
        <h1>{title}</h1>
    </div>
  )}
```

DIGITALHOUSE

Gracias