educación IT
CODE YOUR FUTURE

ReactJS Developer

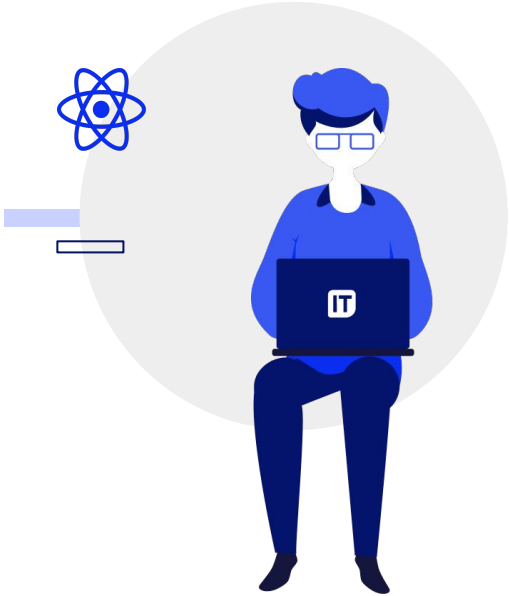
Módulo 5

Composición de componentes

Composición de componentes

React tiene un modelo de composición muy potente y recomendamos usar **composición en lugar de herencia para reutilizar código entre componentes.**

Composición puede ser denominado también como ***envoltura*** ya que consiste en ***envolver*** **un componente con otro.**



Contención

Algunos componentes no conocen sus hijos de antemano. Esto es especialmente común para componentes como `Sidebar` o `Dialog` que representan “cajas” genéricas.

En casos como estos, conviene que nuestros componentes usen la `prop` especial **`children`** para pasar elementos hijos **directamente en su resultado.**

La `prop` especial **`children`** contiene **todos los componentes que se pasan como nodos hijos.**

```
function Layout(props)
{
    return (
        <div classN any container">
            {props.children}
        </div>
    )
}

export default Layout;
```

```
import Layout from "./Layout";

function App(props)
{
    return (
        <Layout>
            <div>Contenido principal</div>
        </Layout>
    )
}

export default App;
```

Especialización

A veces pensamos en componentes como “casos concretos” de otros componentes. Por ejemplo, podríamos decir que un **`WelcomeDialog`** es un caso concreto de **`Dialog`**.

En React, esto también se consigue por composición, en la que un componente más “específico” renderiza uno más “genérico” y lo configura con `props`.

```
function BaseLayout(props)
{
    return (
        <div className={props.class}>
            {props.children}
        </div>
    )
}

export default BaseLayout;

```

```
import BaseLayout from "./BaseLayout";

function Layout(props)
{
    return (
        <BaseLayout class="container">
            {props.children}
        </BaseLayout>
    )
}

export default Layout;

```

Componente de orden superior (*HOC*)

Un componente de orden superior (*HOC* por las siglas en inglés de *Higher-Order Component*) es una técnica avanzada en React para **el reuso de la lógica de componentes.** Los *HOCs* **no son parte de la *API* de React** sino **un patrón que surge de la naturaleza composicional de React.**

En concreto, un componente de orden superior es **una función que recibe un componente y devuelve un nuevo componente.**

Mientras que un componente transforma **props en interfaz de usuario, un componente de orden superior transforma un componente en otro.**

Piensa en los *HOC* como el patrón `Decorator` aplicado a React.

Consideraciones

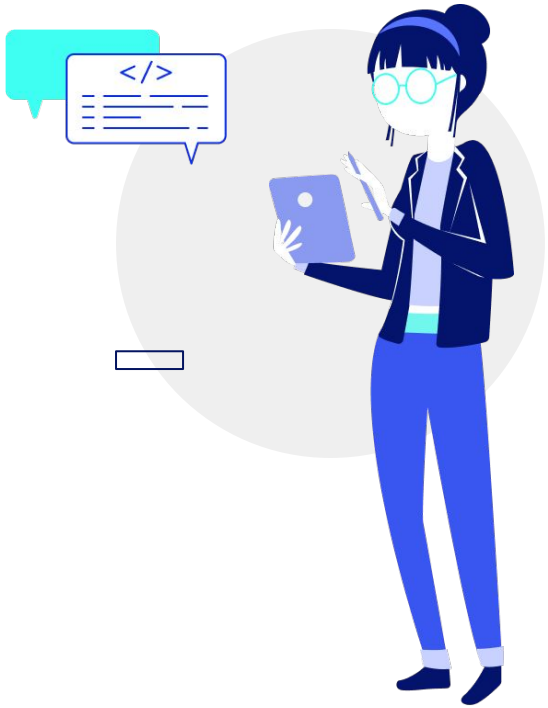
- Un componente de orden superior es una función que **recibe un componente base y devuelve el mismo componente pero con las props modificadas.**
- Recuerda que los componentes en React se escriben en **CamelCase**. Debes respetar esa sintaxis en el parámetro que contenga el componente a envolver.
- Puedes devolver **cualquier tipo de componentes.**
- Estas funciones se usan para **requisitos transversales** (como son algunas solicitudes de *API*, por ejemplo).
- Las props son el canal por el que se comunica a **la función *HOC* con el otro componente.**

```jsx
function withPosts(BaseComponent) { // HOC
    const posts = [
        { id: 1, title: "post 1" },
        { id: 2, title: "post 2" },
        { id: 3, title: "post 3" },
    ];

    return function (props) {
        return <BaseComponent posts={posts} {...props} />;
    };
}

function PostsLists(props) { // Componente base
    return (
        <ul>
            {props.posts.map((p, i) => (
                <li>{p.title}</li>
            ))}
        </ul>
    );
}

export default withPosts(PostsLists); // Conexión
```

Links de interés

Composición vs. herencia – React

Componentes de orden superior – React



educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!