Componentes II

Conceptualmente, los componentes son como las funciones de JavaScript. Aceptan entradas arbitrarias (llamadas "props") y retornan elementos de React que describen lo que debe aparecer en la pantalla.

Componentes funcionales

La forma más sencilla de definir un componente es escribir una función de JavaScript:

```
function Bienvenido(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Esta función es un componente de React válido porque acepta un solo argumento de objeto "props" (que proviene de propiedades) con datos y devuelve un elemento de React. Llamamos a dichos componentes "funcionales" porque literalmente son funciones JavaScript.

También existe otra forma de definir los componentes funcionales, con las funciones flecha que vienen gracias a ES6 (ECMAScript 6), las cuales se pueden declarar de la siguiente forma:

```
const Bienvenido = (props) => {
  return <h1>Hello, {props.name}</h1>;
}
```

En verdad no existe mucha diferencia al respecto una de otra, por lo que vemos son casi similares, pero actualmente con el objetivo de mantener un código más limpio, siguiendo convenciones y teniendo buenas prácticas se utilizan mayoritariamente los componente funcionales flecha.

Composición de componentes

Los componentes pueden referirse a otros componentes en su salida. Esto nos permite utilizar la misma abstracción de componente para cualquier nivel de detalle. Un botón, un cuadro de diálogo, un formulario, una pantalla: en aplicaciones de React, todos son expresados comúnmente como componentes.

```
const HolaAlumno = (props) => {
  return <h1>Hola, {props.name}</h1>;
}

const BienvenidosAlumnos = () => {
  return (
    <div>
      <HolaAlumno name="Sara" />
      <HolaAlumno name="Cahal" />
      <HolaAlumno name="Edite" />
    </div>
  );
}
```

En el caso que estamos viendo en la imagen, nuestro componente BienvenidosAlumnos, esta renderizando a otro componente el cual se llama HolaAlumno el cual retorna un h1 con un texto que esta conformado por un Hola, y lo que le llegue como props.name , en el primer caso lo que va a renderizar es el siguiente texto ' Hola, Sara ' ya se le esta mandando como name Sara.

```
const HolaAlumno = (props) => {
  return <h1>Hola, {props.name}</h1>;
}

const BienvenidosAlumnos = () => {
  return (
    <div>
      <HolaAlumno name="Sara" />
      <HolaAlumno name="Cahal" />
      <HolaAlumno name="Edite" />
    </div>
  );
}
```

Hola, Sara

Hola, Cahal

Hola, Edite

Render condicional

En algunos casos necesitamos que los componentes solo se muestren en casos puntuales, por ejemplo queremos renderizar un botón de Salir de la cuenta , cuando el usuario este dentro de la aplicación y como este ejemplo puede haber muchos casos. Y la mejor manera de verificar cuando renderizar o no un componente es con el operador lógico &&

El operador && se entiende como si fuera un "y". Es decir, en la condición de la sentencia IF, se puede añadir este operador para que en vez de una sola condición, se cumplan 2 o más condiciones.

Por ende en base a este operador lógico vamos a realizar un renden condicional, y la manera de realizarlo es encerrar el componente entre llaves { } y haciendo la validación con el operador &&

Ejemplo :

```
3  const HolaAlumno = (props) => {
4    return <h1>Hola, {props.name}</h1>;
5  }
6
7  const BienvenidosAlumnos = () => {
8    return (
9      <div>
10       <HolaAlumno name="Sara" />
11       {false && <HolaAlumno name="Cahal" />}
12       <HolaAlumno name="Edite" />
13     </div>
14   );
15 }
16
```

Hola, Sara

Hola, Edite

En este caso en la línea 11 tenemos el componente dentro de llaves y se está validando con el operar lógico, pero como le estamos mandando un false, podemos ver que en la parte derecha no aparece el Hola, Cahal.

En este caso estamos forzando un false, pero tranquilamente puede ir una validación ej: 4 > 3 && , o un useState, entre otras.