educación IT
CODE YOUR FUTURE

ReactJS Developer

Módulo 3

Renderizado condicional

Renderizado condicional

El renderizado condicional funciona **de la misma forma que los condicionales en JavaScript.** Al igual que JavaScript, sirve para que **el intérprete ejecute o no determinadas partes del código dependiendo del valor de verdad** (verdadero o falso) de una expresión.

Sin embargo, y a diferencia de Javascript, que condiciona líneas de código, en React condicionamos el renderizado de componentes. Dicho con otras palabras: mientras que en JavaScript condicionamos líneas, **en React condicionamos renderizado de componentes en la pantalla.**

Renderizado condicional con variables

Paso a paso

1. Cree **una variable** para contener el componente a mostrar. **No debe ser `const` ya que su valor cambiará luego.**
2. Muestre esa variable **en el lugar donde quiera renderizar el componente,** tal como lo haría con cualquier otro dato.
3. Luego de la creación de la variable y antes de su renderizado, **defina la condición necesaria con un bloque `if`.**
4. Dentro del cuerpo del `if`, cambie el valor a la variable por cualquier elemento ***JSX*.**
5. Recuerde que cada etiqueta *JSX* será procesada luego como llamadas a `React.createElement`, por lo que pueden usarse como valores de **variables.**

```jsx
function NavBar(props)
{

    let aMostrar = null; // 1


    if(props.isLoggedIn)
    {
        /**
         * La siguiente línea se convierte luego en
         * aMostrar = React.createElement('div', {}, 'Usuario Loguedo')
         * Por lo tanto, es válida esta sintaxis
         */
        aMostrar = <div>Usuario logueado</div>
    }
    else
    {
        /**
         * La siguiente línea se convierte luego en
         * aMostrar = React.createElement('div', {}, 'Sin usuario')
         * Por lo tanto, es válida esta sintaxis
         */
        aMostrar = <div>Sin usuario</div>
    }

    return (
        <nav>
            <ul>
                <li>Home</li>
                <li>About</li>
                {aMostrar /*2*/}
            </ul>
        </nav>
    );
}

export default NavBar;
```

Renderizado condicional con operador &&

Paso a paso

- El proceso en sí mismo consta de un solo paso. Cuando escribimos una condición **AND** (**&&**) el lenguaje evalúa primero **la expresión del lado izquierdo** del operador. Si esa expresión es verdadera, entonces el lenguaje evalúa luego **la expresión del lado derecho**.

- De esto se sigue que, si quiero que una expresión se ejecute solo si otra es cierta, puedo hacerlo así:

```
expresión_a_evaluar &&
expresión_a_ejecutar_si_es_cierta
```

- De esta forma sólo evaluará el lenguaje la segunda expresión si la primera es cierta.

```
1   function NavBar(props)
2   {
3       return (
4           <nav>
5               <ul>
6                   <li>Home</li>
7                   <li>About</li>
8                   { props.isLoggedIn && <div>Usuario logueado</div>}
9               </ul>
10          </nav>
11      );
12  }
13
14  export default NavBar;
15
```

Renderizado condicional con ternario

Paso a paso

- El proceso consta de un solo paso. Podemos pensar en el ternario como un operador **&&** **con la posibilidad de añadir el comportamiento ELSE.**
- El operador **&&** ejecuta y evalúa la expresión a su derecha sólo si la expresión a su izquierda es **verdadera.** Sin embargo, no contempla el caso en que la expresión sea **falsa**. Simplemente **no hace nada**.
- Acá es cuando el ternario es útil, ya que permite evaluar una expresión en caso de que la condición sea **falsa**.

  Su sintaxis es la siguiente:

```
condición ? expresión_si_verdadera :
expresión_si_falsa
```

```jsx
function NavBar(props)
{
    return (
        <nav>
            <ul>
                <li>Home</li>
                <li>About</li>
                { props.isLoggedIn ? <div>Usuario logueado</div> : <div>Sin usuario</div>}
            </ul>
        </nav>
    );
}

export default NavBar;

```

Renderizado condicional de todo el componente

Paso a paso

Esta estrategia sirve para condicionar todo el renderizado de componente, y no solo una de sus partes. Consta de **dos pasos principales**:

1. Como primer paso, debemos añadir un bloque **`if`** con las condiciones necesarias **para que el componente no se renderice.** Para que el componente no se renderice debemos **retornar `null`.**
2. El segundo paso sería escribir el *JSX* de todo el componente **como si la condición del `if` fuese falsa.**

En otras palabras: el código del componente **debe mostrar el caso en que esa condición no se cumpla o no debe mostrar nada.**

```
function NavBar(props)
{
    if(!props.isLoggedIn)
        return null;

    return (
        <nav>
            <ul>
                <li>Home</li>
                <li>About</li>
                <div>Usuario logueado</div>
            </ul>
        </nav>
    );
}

export default NavBar;
```

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!