educación IT
CODE YOUR FUTURE

ReactJS Developer

Módulo 3

Formularios

Formularios

Un formulario es **un dispositivo de entrada de datos.** A diferencia de un formulario *HTML* común, los formularios en React **almacenan internamente el estado actual de los valores.**

Esto sirve para implementar validaciones más **personalizadas** y **respuestas más interactivas.**

Vamos a lograr esto mediante una estrategia llamada **“Componentes controlados”**.

Un componente controlado no es nada nuevo, consiste simplemente en crear o almacenar el estado de un campo de entrada de formulario en un componente React, de forma tal que ocurran dos comunicaciones principales:

- Que al cambiar el estado del componente, **cambie el valor del campo del formulario.**
- Que al cambiar el valor del campo del formulario, **cambie el estado del componente.**

Evento *onSubmit*

El primer aspecto a tener en cuenta, antes de pasar al diseño de componentes controlados, es el evento **`onSubmit`**. Este evento se dispara cuando se decide **enviar un formulario.** Podemos asignar un `handler` a este evento (como vimos en otros ejemplos).

En este `handler` podemos hacer tres tareas importantes:

- **Evitar que el formulario se envíe automáticamente.**
- Realizar **validaciones correspondientes.**
- Enviar el formulario únicamente **cuando sea necesario.** En este punto podemos optar o bien por el envío normal de un formulario *HTML* o bien por **enviar los datos de forma asincrónica.**

```
function LoginForm(props)
{
    const handleSubmit = e => {
        e.preventDefault(); // 1. Evitamos el envío automático
        let enviar = true;
        /**
         * Acá hacemos todas las validaciones
         * correspondientes, definiendo si se debe
         * o no enviar este formulario
         */
        if(enviar)
        {
            // Si el formulario se envía podemos optar por dos alternativas
            /** ALTERNATIVA 1: Envio HTML común */
            e.target.submit();

```

...

```
16  ...
17              /** ALTERNATIVA 2: Envio asincrónico */
18              const data = new FormData();
19              // agregamos a FormData los datos a enviar acá.
20              fetch('/login', {
21                  method: 'POST',
22                  body: data
23              }).then(r => console.log(r));
24          }
25
26      };
27
28      return (
29          <form action="/login" method="POST" onSubmit={handleSubmit}>
30              {/* Campos del formulario aquí */}
31              <button>Enviar</button>
32          </form>
33      );
34  }
35
36  export default LoginForm;
```

Componentes controlados

Llamamos “Componentes controlados” a una **estrategia de diseño de componentes, un patrón de diseño.**

En campos de formularios *HTML*, el estado es mantenido **dentro del campo.** En React podemos mantener el estado de ese componente en el estado del mismo (en lugar de dentro del campo *HTML*). Esto sirve para **mejorar la interacción del campo *HTML* con React.**

Un componente controlado es un componente normal con la salvedad de que gestiona **dos comunicaciones puntuales:**

1. **Del estado al campo de entrada:** cambiar el valor del campo de entrada **cuando cambia el estado del componente.**
2. **Del campo de entrada al estado:** cambiar el valor del estado **cuando cambia el valor del campo de entrada.**

De esta forma, tenemos reflejado en el estado del componente el valor del campo de entrada, lo que permite que React **gestione también el valor del campo** con todo lo que eso conlleva (reactividad, interactividad, etc.).

Paso a paso

1. Define **una porción del estado** para almacenar **el valor actual del campo de entrada.**
2. Asigna como valor de la propiedad **`value`** del campo de entrada **el valor de esa porción del estado.**
3. Asigna **un `handler` al evento `onChange / onInput` del campo de entrada.** En ese `handler` **haz el cambio de estado correspondiente.**
4. Ahora tienes reflejado en el estado **el valor del campo de entrada** y puedes leerlo desde ahí.

```jsx
import { useState } from "react";

function LoginForm(props) {
    const [nombre, setNombre] = useState("");

    return (
        <form>
            <input
                type="text"
                name="nombre"
                value={nombre}
                onChange={(e) => setNombre(e.target.value)}
            />
            <button>Enviar</button>
        </form>
    );
}

export default LoginForm;

```

Componentes controlados Caso *<textarea>*

En *HTML*, la etiqueta **`textarea`** ofrece su valor actual a través de sus hijos (*innerHTML*).

En React, ese valor se ofrece a través de la propiedad **`value`.**

```jsx
import { useState } from "react";

function LoginForm(props) {
    const [texto, setTexto] = useState("");

    return (
        <form>
            <textarea
                value={texto}
                onChange={(e) => setTexto(e.target.value)}
            ></textarea>
            <button>Enviar</button>
        </form>
    );
}

export default LoginForm;
```

Componentes controlados Caso *<select>*

En *HTML*, la etiqueta **`select`** ofrece su valor actual a través de la propiedad **`value`** pero se selecciona un elemento por código mediante el atributo **`selected`**.

En React se hacen ambas cosas a través de la propiedad **`value`.**

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

function LoginForm(props) {
    const [texto, setTexto] = useState("");

    return (
        <form>
            <textarea
                value={texto}
                onChange={(e) => setTexto(e.target.value)}
            ></textarea>
            <button>Enviar</button>
        </form>
    );
}

export default LoginForm;
```

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!