React Nivel Intermedio: Formularios y Validaciones 📝

En este nivel, aprenderás a manejar formularios en React, uno de los aspectos más importantes para crear aplicaciones web interactivas. Cubriremos cómo trabajar con inputs controlados, realizar validaciones, y gestionar diferentes tipos de inputs como checkbox, radio buttons y selects. Dominar estos conceptos te permitirá crear formularios complejos y robustos para cualquier proyecto.

Temas que Cubriremos 📚

1. ¿Qué es un formulario controlado en React?
2. Validación de formularios y manejo de errores.
3. Manejo del evento `onSubmit` para capturar datos.
4. Checkbox y radio buttons: cómo gestionarlos en React.
5. Creación de selects dinámicos y manejo del valor seleccionado.

---

1. ¿Qué es un Formulario Controlado en React? 🤔

Un formulario controlado es aquel donde los valores de los inputs son gestionados completamente a través del estado del componente usando `useState`. En un formulario controlado, cada input tiene un valor que se almacena en el estado, lo que permite actualizar la UI en tiempo real a medida que el usuario escribe.

Ejemplo:

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

function FormularioControlado() {
  const [nombre, setNombre] = useState("");
  const [email, setEmail] = useState("");

  return (
    <form>
      <input
        type="text"
        value={nombre}
        onChange={(e) => setNombre(e.target.value)}
        placeholder="Nombre"
      />
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        placeholder="Email"
      />
      <p>Nombre: {nombre}</p>
      <p>Email: {email}</p>
    </form>
  );
```

```
}

export default FormularioControlado;
```

En este ejemplo, los valores de los inputs están controlados por el estado del componente. Esto permite actualizar la vista en tiempo real a medida que el usuario escribe.

---

2. Validación de Formularios y Manejo de Errores 🛡️

La validación de formularios es esencial para asegurar que los datos ingresados por los usuarios sean correctos. En React, puedes validar los datos en eventos como `onBlur` (cuando el usuario pierde el foco del input) o `onSubmit` (cuando se envía el formulario).

Ejemplo de Validación de Email:

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

function FormularioValidado() {
  const [email, setEmail] = useState("");
  const [error, setError] = useState("");

  const validarEmail = (email) => {
    const regex = /^[^\s@]+@[^\s@]+.[^\s@]+$/;
    return regex.test(email);
  };

  const handleBlur = () => {
    if (!validarEmail(email)) {
      setError("Email no válido");
    } else {
      setError("");
    }
  };

  return (
    <div>
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        onBlur={handleBlur}
        placeholder="Email"
      />
      {error && <p style={{ color: "red" }}>{error}</p>}
    </div>
  );
}

export default FormularioValidado;
```

En este ejemplo, se valida el email al perder el foco del input (`onBlur`). Si el email no cumple con el formato, se muestra un mensaje de error.

---

3. Manejo del Evento `onSubmit` para Capturar Datos 📤

El evento `onSubmit` permite manejar el envío del formulario. Puedes capturar los datos ingresados y realizar acciones como mostrarlos en una alerta o enviarlos a un servidor.

Ejemplo:

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

function FormularioSubmit() {
  const [nombre, setNombre] = useState("");
  const [email, setEmail] = useState("");

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Nombre: ${nombre}\nEmail: ${email}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        value={nombre}
        onChange={(e) => setNombre(e.target.value)}
        placeholder="Nombre"
      />
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        placeholder="Email"
      />
      <button type="submit">Enviar</button>
    </form>
  );
}

export default FormularioSubmit;
```

En este ejemplo, al hacer clic en el botón "Enviar", los datos del formulario se muestran en una alerta.

---

4. Manejo de Checkbox y Radio Buttons 🔘

Los inputs de tipo checkbox y radio button se manejan de forma similar a los inputs de texto, pero suelen requerir un poco más de lógica para controlar su estado.

Ejemplo:

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

function FormularioCheckboxRadio() {
  const [aceptaTerminos, setAceptaTerminos] = useState(false);
  const [genero, setGenero] = useState("");
```

```
  return (
    <div>
      <input
        type="checkbox"
        checked={aceptaTerminos}
        onChange={(e) => setAceptaTerminos(e.target.checked)}
      />
      <label>Acepto los términos y condiciones</label>

      <div>
        <input
          type="radio"
          value="Masculino"
          checked={genero === "Masculino"}
          onChange={(e) => setGenero(e.target.value)}
        />
        Masculino
        <input
          type="radio"
          value="Femenino"
          checked={genero === "Femenino"}
          onChange={(e) => setGenero(e.target.value)}
        />
        Femenino
      </div>

      <p>Género seleccionado: {genero}</p>
    </div>
  );
}

export default FormularioCheckboxRadio;
```

En este ejemplo, mostramos cómo manejar el estado de un checkbox y radio buttons.

---

5. Select Dinámico - Creación de un Dropdown 🌍

Un select dinámico permite mostrar opciones basadas en datos que pueden provenir de un array o una API.

Ejemplo:

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

function SelectDinamico() {
  const [pais, setPais] = useState("");
  const paises = ["España", "México", "Argentina", "Colombia"];

  return (
    <div>
      <select value={pais} onChange={(e) => setPais(e.target.value)}>
        <option value="">Selecciona un país</option>
        {paises.map((pais) => (
          <option key={pais} value={pais}>
```

```
            {pais}
          </option>
        ))}
      </select>
      <p>País seleccionado: {pais}</p>
    </div>
  );
}

export default SelectDinamico;
```

En este ejemplo, mostramos cómo manejar el estado de un select dinámico basado en un array de opciones.

Conclusión 🏁

En este nivel, aprendiste a manejar formularios en React, validar inputs, y trabajar con diferentes tipos de elementos de formulario. Estos conceptos son esenciales para crear aplicaciones interactivas y robustas. ¡Sigue practicando para perfeccionar tus habilidades! 🚀💻