«Talento Tech»

React JS

Clase 04

Buenos Aires Ciudad

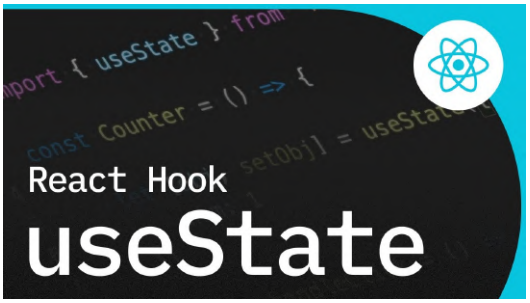
Clase N° 4 | React Hooks y Eventos

Índice:

Objetivos de la Clase:

- Aprender a gestionar el estado local de un componente usando el hook **useState**.
- Manejar eventos comunes en React, como clics y envío de formularios.
- Crear una funcionalidad básica para agregar productos a un carrito de compras.

Introducción al manejo del estado local con useState



useState es uno de los hooks más utilizados en React y se usa para manejar el estado local de los componentes funcionales. Antes de hooks como **useState**, sólo los componentes de clase podían gestionar el estado, lo que hacía que los componentes funcionales fueran menos versátiles. Ahora, con **useState**, cualquier componente funcional puede tener su propio estado.

¿Qué es el estado? El estado en React es una estructura de datos que se utiliza para almacenar información dinámica. Esta información puede cambiar a lo largo de la vida útil del componente, como un contador que se incrementa o una lista que se actualiza.

¿Cómo funciona useState? Cuando usamos **useState**, declaramos una variable de estado y una función para actualizarla. El hook acepta un valor inicial como argumento y devuelve un arreglo con dos elementos: el valor actual del estado y la función para modificarlo.

Ejemplo básico:

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

function Contador() {
    const [contador, setContador] = useState(0);

    return (
        <div>
            <p>Valor del contador: {contador}</p>
            <button onClick={() => setContador(contador +
1)}>Incrementar</button>
        </div>
    );
}
export default Contador;
```

En este ejemplo, contador es el estado inicial y setContador es la función para actualizarlo.

Reglas importantes de useState:

1. **No se debe modificar el estado directamente:** Siempre utiliza la función proporcionada por **useState** (en este caso, `setContador`). Modificar el estado directamente puede causar comportamientos inesperados.
2. **useState no combina estados automáticamente:** Si tienes múltiples variables de estado, necesitas declararlas por separado.
3. **Se ejecuta en cada renderización:** Cada vez que el estado cambia, el componente se vuelve a renderizar para reflejar los cambios.

Comprensión y manejo de eventos en React (clics, formularios)

Los eventos en React se basan en la sintaxis de JavaScript, pero tienen ciertas particularidades que los hacen más consistentes y seguros. React utiliza un sistema de delegación de eventos, lo que mejora el rendimiento al no adjuntar listeners directamente al DOM.

¿Cómo manejar eventos en React?

- Los eventos en React se escriben en camelCase (por ejemplo, onClick en lugar de onclick).
- Los manejadores de eventos suelen ser funciones que puedes definir en el mismo componente.

Ejemplo de manejo de clics:

```
function Boton() {

  function manejarClick() {

      alert('Botón clickeado!');
```

```
  }

  return <button onClick={manejarClick}>Hacer clic</button>;

}
```

Eventos más comunes en React:

1. **onClick:** Se activa cuando un elemento es clickeado.
2. **onChange:** Se usa principalmente en formularios para detectar cambios en los inputs.
3. **onSubmit:** Se dispara cuando un formulario es enviado.
4. **onMouseOver / onMouseOut:** Para manejar eventos del mouse.

Los formularios suelen ser controlados, lo que significa que el estado de los elementos del formulario (inputs, selects, etc.) se maneja a través de variables de estado. Esto permite un control más fino sobre los datos del formulario y cómo se procesan.

Ejemplo de manejo de formularios:

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

function Formulario() {

  const [nombre, setNombre] = useState('');

  function manejarEnvio(evento) {

      evento.preventDefault();

      alert(`Formulario enviado por: ${nombre}`);

  }

  return (

      <form onSubmit={manejarEnvio}>

          <input
```

```
                type="text"
                value={nombre}
                onChange={(e) => setNombre(e.target.value)}
                placeholder="Ingresa tu nombre"
            />
            <button type="submit">Enviar</button>
        </form>
    );
}
```

Explicación del ejemplo:

- value={nombre} hace que el input sea un elemento controlado, donde el valor proviene del estado.
- onChange={(e) => setNombre(e.target.value)} actualiza el estado en cada cambio del input.
- onSubmit maneja el evento de envío del formulario.

Ventajas de los formularios controlados:

1. Facilita la validación en tiempo real.
2. Permite formatear o limpiar datos antes de enviarlos.
3. Centraliza la lógica en el estado del componente.

Tu primer tarea en TalentoLab

¡Felicitaciones! 🎉 Has sido seleccionado para formar parte de TalentoLab, una empresa innovadora que desarrolla soluciones tecnológicas. Serás parte del equipo responsable de crear un eCommerce para un cliente.

Tu tarea inicial será crear una funcionalidad básica para el manejo de un carrito de compras.

Descripción de tu tarea inicial:

- Crear una aplicación que permita a los usuarios agregar productos a un carrito de compras y visualizar la lista de productos seleccionados.

1. **Requisitos:**
    - Crear un componente para listar los productos disponibles.
    - Usar el hook **useState** para manejar el estado del carrito.
    - Implementar un evento de clic que permita agregar productos al carrito.
    - Mostrar el carrito con los productos seleccionados en otro componente.
    - Crear un Layout del eCommerce
2. **Pautas Generales:**
    - La lista de productos debe incluir al menos un nombre y un precio.
    - El estado del carrito debe actualizarse dinámicamente cada vez que se agregue un producto.
    - Si el carrito está vacío, mostrar un mensaje como: "El carrito está vacío".
3. **Desafío Adicional:**
    - Implementar un botón "Vaciar Carrito" que elimine todos los productos seleccionados.

Reflexión final

Aprendimos dos conceptos clave en React: la gestión del estado local con **useState** y el manejo de eventos, ambos esenciales para crear aplicaciones web dinámicas e interactivas. Durante la práctica, aplicamos estos conocimientos para construir una funcionalidad de carrito de compras, simulando una tarea real en el desarrollo de un eCommerce. Esto no solo refuerza los conceptos teóricos, sino que también prepara a los estudiantes para resolver problemas similares en proyectos más complejos.

La gestión del estado nos permitió almacenar y manipular datos dinámicos, mientras que el manejo de eventos nos ayudó a interactuar con el usuario de manera efectiva. Este enfoque modular y práctico fortalece las bases necesarias para avanzar hacia aplicaciones más sofisticadas.

Materiales y Recursos Adicionales:

- ★ Documentación oficial de React - useState
- ★ React Events Handling - Documentación
- ★ Guía completa sobre formularios en React
- ★ Ejercicios prácticos sobre React Hooks

Preguntas para Reflexionar:

- ¿Qué beneficios ofrece React al manejar eventos comparado con JavaScript puro?
- ¿Qué aprendiste al trabajar con formularios en React? ¿Qué desafíos enfrentaste?
- ¿Cómo podrías extender la aplicación para incluir funcionalidades como eliminar productos individuales del carrito?

Próximos Pasos:

- Introducción a useEffect para efectos secundarios.
- Realización de peticiones a APIs (cargar productos).
- Manejo de estado de carga y errores.

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad