Fundamentos de ReactJS

L **by LEANDRO HERNAN ZABALA IGLESIAS**

1. JSX: Sintaxis y uso

1.1 ¿Qué es JSX y por qué se usa?

Definición: JSX (JavaScript XML) es una extensión de la sintaxis de JavaScript que permite escribir elementos de interfaz de usuario de manera declarativa y en un formato similar al HTML.

Ventajas:

- Facilita la creación de interfaces de usuario complejas.
- Mejora la legibilidad del código al permitir escribir HTML y JavaScript en un mismo archivo.
- Transforma automáticamente el JSX en llamadas a React.createElement, optimizando el rendimiento.

Ejemplo básico:

```
const saludo = <h1>Hola, mundo</h1>;
```

1.2 Integración de variables y funciones en JSX

Uso de variables:

- JSX permite insertar valores dinámicos mediante {}.
- Ejemplo:

```
const nombre = "Leandro";
const saludo = <h1>Hola, {nombre}</h1>;
```

Uso de funciones en JSX:

- Puedes llamar funciones que devuelvan valores dentro de {}.
- Ejemplo:

```
function obtenerFecha() {
  return new Date().toLocaleDateString();
}
const fecha = <p>Hoy es: {obtenerFecha()}</p>;
```

1.3 Buenas prácticas con JSX

Cierre de etiquetas:

- Siempre cierra etiquetas, incluso las vacías.
- Ejemplo:

```
<img src="ruta" alt="descripción" />
```

Uso de llaves {}:

- Solo para insertar valores dinámicos.

Fragmentos:

- Usar `<>...</>` o `<React.Fragment>` para envolver múltiples elementos sin crear nodos innecesarios.

2. Componentes en React

2.1 Creación de componentes funcionales y de clase

Componentes funcionales:

- Son funciones que retornan JSX.
- Ejemplo:

```
function Saludo() {
  return <h1>¡Hola desde un componente funcional!</h1>;
}
```

Componentes de clase:

- Son clases que extienden React.Component.
- Ejemplo:

```
class Saludo extends React.Component {
  render() {
    return <h1>¡Hola desde un componente de clase!</h1>;
  }
}
```

2.2 Props: Cómo pasar datos a los componentes

Definición: Las props (propiedades) son datos que se pasan a los componentes desde su padre.

Ejemplo práctico:

```
function Saludo({ nombre }) {
  return <h1>Hola, {nombre}!</h1>;
}
// Uso del componente
<Saludo nombre="Leandro" />
```

2.3 Ejemplo práctico: creación de un componente simple

Crea un componente funcional que reciba props y renderice un saludo con el nombre y la hora actual.

```
function Saludo({ nombre }) {
  const hora = new Date().toLocaleTimeString();
  return <p>Hola, {nombre}. La hora actual es {hora}</p>;
}
```

3. Estados y ciclo de vida de componentes

3.1 Uso de useState y estados en componentes funcionales

Definición: Los estados son valores que cambian con el tiempo y afectan la renderización del componente.

Ejemplo básico:

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

  return (
    <div>
      <p>El contador es: {contador}</p>
      <button onClick={() => setContador(contador + 1)}>Incrementar</button>
    </div>
  );
}
```

3.2 Introducción a Hooks: useEffect para efectos secundarios

Definición: useEffect permite manejar efectos secundarios como llamadas a APIs, temporizadores o actualizaciones manuales del DOM.

Ejemplo básico:

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

function Reloj() {
  const [hora, setHora] = useState(new Date().toLocaleTimeString());

  useEffect(() => {
    const timer = setInterval(() => {
      setHora(new Date().toLocaleTimeString());
    }, 1000);

    return () => clearInterval(timer); // Limpieza del efecto
  }, []);

  return <p>Hora actual: {hora}</p>;
}
```

3.3 Ejercicio: Componente con estados y efectos

Crea un componente que muestre un contador que se incremente automáticamente cada segundo, y permita al usuario pausarlo o reiniciarlo.

4. Eventos en React

4.1 Manejo de eventos (onClick, onChange, etc.)

Definición: React normaliza eventos del navegador y los asocia con funciones de JavaScript.

Ejemplo básico de onClick:

```
function Boton() {
  const manejarClick = () => alert("¡Botón clicado!");
  return <button onClick={manejarClick}>Haz clic aquí</button>;
}
```

Ejemplo de onChange en un input controlado:

```
function InputControlado() {
  const [valor, setValor] = useState("");

  return (
    <input
      type="text"
      value={valor}
      onChange={(e) => setValor(e.target.value)}
    />
  );
}
```

4.2 Ejercicio: Formulario controlado con estados

Crea un formulario con un campo de texto y un botón. Al enviar el formulario:

1. Valida que el campo no esté vacío.
2. Muestra un mensaje con el valor ingresado.

Ejemplo de formulario:

```
function Formulario() {
  const [nombre, setNombre] = useState("");
  const [mensaje, setMensaje] = useState("");

  const manejarSubmit = (e) => {
    e.preventDefault();
    if (nombre.trim() === "") {
      setMensaje("Por favor, ingresa un nombre.");
    } else {
      setMensaje(`¡Hola, ${nombre}!`);
      setNombre("");
    }
  };

  return (
    <form onSubmit={manejarSubmit}>
      <input
        type="text"
        value={nombre}
        onChange={(e) => setNombre(e.target.value)}
        placeholder="Ingresa tu nombre"
      />
      <button type="submit">Enviar</button>
      {mensaje && <p>{mensaje}</p>}
    </form>
  );
}
```