# Desarrollo de Aplicaciones Informáticas

**Tema**: React `hooks`, objeto `Event` y `Conditional Rendering`

---

**Docente**: Ing. Pablo Morandi
**Ayudante**: Matías Marchesi
**Curso**: 5to año
**Especialidad**: Informática

---

**Redactado por:** Ing. Pablo Morandi
**Versión**: v1.1

---

## Índice



---

## 1. ¿Qué es el Estado en React?

El **estado** es la información que puede cambiar durante la vida de un componente. Es como la memoria del componente, donde guardamos datos que pueden modificarse y que afectan lo que se muestra en pantalla.

**¿Por qué necesitamos estado?**

- Para que la interfaz reaccione a las acciones del usuario
- Para guardar información temporal (formularios, contadores, etc.)
- Para controlar el comportamiento dinámico de los componentes

**Antes de React (JavaScript tradicional):**

```
let contador = 0;
document.getElementById("numero").innerHTML = contador;
```

**Con React:**

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

---

## 2. Hook useState - Manejo de Estado

El hook `useState` nos permite agregar estado a los componentes funcionales.

### Sintaxis Básica

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

export default function MiComponente() {
    const [estado, setEstado] = useState(valorInicial);

    return (
        // JSX que usa el estado
    );
}
```

**Explicación:**

- `estado`: Variable que contiene el valor actual
- `setEstado`: Función para modificar el estado
- `valorInicial`: Valor que tendrá al inicio (0, "", [], {}, etc.)

### Ejemplo con Contador

```
"use client"

import { useState } from "react";

export default function Contador() {
  const [cuenta, setCuenta] = useState(0);

  return (
    <div>
      <h2>Contador: {cuenta}</h2>
    </div>
  );
}
```

# 3. Hook useEffect - Efectos Secundarios

El hook `useEffect` nos permite ejecutar código cuando algo cambia en nuestro componente.

## Sintaxis Básica

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

export default function MiComponente() {
    const [estado, setEstado] = useState(0);

    useEffect(() => {
        // Código que se ejecuta cuando cambia algo
        console.log('El estado cambió:', estado);
    }, [estado]); // Array de dependencias

    return (
        // JSX
    );
}
```

**¿Cuándo se ejecuta useEffect?**

- Cuando el componente se monta por primera vez
- Cuando cambian las variables del array de dependencias
- Si no ponemos array de dependencias, se ejecuta en cada render

## Tipos de useEffect

```jsx
"use client"

import { useState, useEffect } from 'react';

export default function TiposUseEffect() {
    const [contador, setContador] = useState(0);

    // 1. Se ejecuta solo al montar el componente
    useEffect(() => {
        console.log("Componente montado");
    }, []); // Array vacío

    // 2. Se ejecuta cuando cambia 'contador'
    useEffect(() => {
        console.log("Contador cambió:", contador);
    }, [contador]);

    // 3. Se ejecuta en cada render (¡cuidado!)
    useEffect(() => {
        console.log("En cada render");
    }); // Sin array

    // 4. Cleanup - se ejecuta al desmontar
    useEffect(() => {
        const timer = setInterval(() => {
            console.log("Timer ejecutándose");
        }, 1000);

        // Cleanup function
        return () => {
            clearInterval(timer);
            console.log("Timer limpiado");
        };
    }, []);

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

---

# 4. Ejemplo Práctico: Contador

Vamos a crear una página completa con un contador que incremente y decremente.

## Creando la Página Contador

Archivo: `src/app/contador/page.js`

```
"use client"

import { useState, useEffect } from "react";

export default function ContadorPage() {
  // Estado del contador
  const [cuenta, setCuenta] = useState(0);

  // useEffect que se ejecuta cuando cambia la cuenta
  useEffect(() => {
    console.log("La cuenta cambió a:", cuenta);
    document.title = `Contador: ${cuenta}`;
  }, [cuenta]);

  // Función para incrementar
  const incrementar = () => {
    setCuenta(cuenta + 1);
  };

  // Función para decrementar
  const decrementar = () => {
    setCuenta(cuenta - 1);
  };

  // Función para resetear
  const resetear = () => {
    setCuenta(0);
  };

  return (
    <div className="contador-container">
      <h1>Página Contador</h1>
      <h2>Cuenta actual: {cuenta}</h2>

      <div className="botones">
        <button onClick={incrementar}>Incrementar (+1)</button>
        <button onClick={decrementar}>Decrementar (-1)</button>
        <button onClick={resetear}>Resetear (0)</button>
      </div>
    </div>
  );
}
```

**¿Qué hace este código?**

1. **useState(0)**: Inicializa el contador en 0
2. **useEffect**: Se ejecuta cada vez que `cuenta` cambia
3. **Funciones**: `incrementar`, `decrementar` y `resetear` modifican el estado
4. **Renderizado**: Muestra el valor actual y los botones

---

## 5. Objeto Event - Capturando Información

El **objeto Event** es información que nos da HTML cuando ocurre una acción (click, cambio en input, etc.).

### ¿Qué contiene un Event?

```
const manejarClick = (event) => {
  console.log(event); // Objeto completo
  console.log(event.type); // Tipo de evento: 'click', 'change', etc.
  console.log(event.target); // Elemento que disparó el evento
};
```

### Tipos de Eventos Comunes

- **onClick**: Cuando se hace clic
- **onChange**: Cuando cambia un input
- **onSubmit**: Cuando se envía un formulario
- **onKeyPress**: Cuando se presiona una tecla

> **Documentación**: Puedes ver todos los tipos de eventos en W3Schools - Event Object

## Ejemplo con onChange

```
"use client"

import { useState } from 'react';

export default function EjemploEvent() {
    const [texto, setTexto] = useState("");

    const manejarCambio = (event) => {
        console.log("Tipo de evento:", event.type); // 'change'
        console.log("Elemento completo:", event.target); // El input
        console.log("Valor actual:", event.target.value); // Lo que escribió el
usuario

        // Actualizar el estado con el nuevo valor
        setTexto(event.target.value);
    };

    return (
        <div>
            <input
                type="text"
                onChange={manejarCambio}
                placeholder="Escribe algo..."
                value={texto}
            />
            <p>Has escrito: {texto}</p>
        </div>
    );
}
```

## Propiedades Importantes del Event.target

```
const analizarEvent = (event) => {
  console.log("Tipo de input:", event.target.type); // 'text', 'email', etc.
  console.log("Valor actual:", event.target.value); // Lo escrito
  console.log("Placeholder:", event.target.placeholder);
  console.log("Está enfocado:", event.target === document.activeElement);
};
```

---

# 6. Conditional Rendering - Renderizado Condicional

**Conditional Rendering** es mostrar u ocultar elementos JSX basándose en el valor de variables o estados.

## ¿Por qué es importante?

- Mostrar mensajes de "Cargando..." mientras obtenemos datos
- Ocultar/mostrar formularios, menús o secciones
- Mostrar diferentes contenidos según el estado del usuario
- Manejar errores y mensajes informativos

## Método 1: Operador && (Más común)

```
"use client"

import { useState } from 'react';

export default function CondicionalBasico() {
    const [mostrarMensaje, setMostrarMensaje] = useState(false);

    return (
        <div>
            <button onClick={() => setMostrarMensaje(!mostrarMensaje)}>
                {mostrarMensaje ? 'Ocultar' : 'Mostrar'} Mensaje
            </button>

            {/* Solo se muestra si mostrarMensaje es true */}
            {mostrarMensaje && (
                <p>¡Este mensaje se muestra condicionalmente!</p>
            )}
        </div>
    );
}
```

**Explicación del &&:**

- Si `mostrarMensaje` es `true`, se renderiza el elemento
- Si `mostrarMensaje` es `false`, no se renderiza nada

Método 2: Operador Ternario (? 😃

```
"use client"

import { useState } from 'react';

export default function CondicionalTernario() {
    const [estaLogueado, setEstaLogueado] = useState(false);

    return (
        <div>
            {estaLogueado ? (
                <div>
                    <h2>¡Bienvenido!</h2>
                    <p>Estás logueado en el sistema</p>
                    <button onClick={() => setEstaLogueado(false)}>
                        Cerrar Sesión
                    </button>
                </div>
            ) : (
                <div>
                    <h2>Iniciar Sesión</h2>
                    <p>Por favor, ingresa tus credenciales</p>
                    <button onClick={() => setEstaLogueado(true)}>
                        Iniciar Sesión
                    </button>
                </div>
            )}
        </div>
    );
}
```

## Método 3: If/Else dentro de Funciones

```
"use client"

import { useState } from 'react';

export default function CondicionalConFuncion() {
    const [estado, setEstado] = useState('cargando');

    const renderizarContenido = () => {
        if (estado === 'cargando') {
            return <div>⏳ Cargando datos...</div>;
        }

        if (estado === 'error') {
            return (
                <div style={{color: 'red'}}>
                    ❌ Error al cargar datos
                    <button onClick={() => setEstado('cargando')}>
                        Reintentar
                    </button>
                </div>
            );
        }
        if (estado === 'exito') {
            return (
                <div style={{color: 'green'}}>
                    ☑ Datos cargados correctamente
                    <ul>
                        <li>Dato 1</li>
                        <li>Dato 2</li>
                        <li>Dato 3</li>
                    </ul>
                </div>
            );
        }
    };

    return (
        <div>
            <h1>Estado de la Aplicación</h1>
            <div>
                <button onClick={() => setEstado('cargando')}>Cargar</button>
                <button onClick={() => setEstado('exito')}>Simular Éxito</button>
                <button onClick={() => setEstado('error')}>Simular Error</button>
            </div>

            {/* Renderizar según el estado */}
            {renderizarContenido()}
        </div>
    );
}
```