



BOOTCAMP

Generación T

 streambe



Clase 3: Estado (useState) y Eventos en React

Bienvenidos a nuestra tercera sesión donde exploraremos los fundamentos de la interactividad en React. Hoy aprenderemos cómo los componentes pueden mantener y actualizar su información interna, respondiendo dinámicamente a las acciones del usuario sin necesidad de recargar la página.

Objetivos de la Clase 3: Estado (useState) y Eventos

1 Comprender el Estado en React

Aprender qué es el "estado" para recordar y actualizar información en la UI, diferenciándolo de variables comunes.

2 Aprender el Hook useState

Entender cómo declarar y actualizar estados con `useState`, y ver su impacto automático en la interfaz.

3 Dominar el Manejo de Eventos

Identificar eventos comunes (`onClick`, `onChange`) y cómo conectarlos con funciones para reaccionar a la interacción del usuario.

4 Aplicar Conocimientos en Práctica

Integrar estado y eventos creando un contador interactivo y un mini formulario funcional.

5 Ganar Confianza y Experimentar

Sentirse capaz de crear interactividad desde cero y personalizar sus propios proyectos.

¿Qué es el Estado en React?

El estado es uno de los conceptos más fundamentales en React. Funciona como la "**memoria**" de nuestros componentes, permitiéndoles:

- Recordar información entre renderizados
- Actualizar la interfaz sin recargar toda la página
- Preservar datos de usuario durante interacciones
- Controlar el comportamiento y apariencia de componentes

A diferencia de las variables regulares que se reinician con cada renderizado, el estado persiste y puede ser modificado de manera controlada.



useState Explicado para Humanos

1

Definición Simple

useState es un "gancho" (hook) que permite a los componentes funcionales tener memoria propia. Antes de los hooks, esto solo era posible con componentes de clase.

2

Sintaxis Básica

```
const [valor, setValor] =  
  useState(valorInicial);
```

- **valor:** Variable que contiene el estado actual
- **setValor:** Función para actualizar el estado
- **valorInicial:** Valor con el que comienza el estado

3

Funcionamiento Interno

Cuando llamamos a la función actualizadora (setValor), React:

1. Actualiza el valor interno del estado
2. Programa un nuevo renderizado del componente
3. Actualiza la UI con el nuevo valor



Ejemplo Básico de useState

Código de un Contador Simple

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

function Contador() {
  // Declaramos estado inicial como 0
  const [count, setCount] = useState(0);

  // Función para manejar el click
  function handleClick() {
    setCount(count + 1);
  }

  return (
```

Has hecho click {count} veces

Haz click aquí

```
); }
```

¿Qué ocurre al ejecutar?

1. El componente inicia con `count = 0`
2. Al hacer click, se llama a `handleClick()`
3. `setCount` actualiza el valor a `count + 1`
4. React vuelve a renderizar el componente
5. La UI muestra el nuevo valor de `count`



¿Por qué usar Estado?

Problema con Variables Normales

Las variables regulares se reinician cada vez que el componente se renderiza nuevamente. Cualquier cambio se pierde instantáneamente.

```
let contador = 0; // ¡Se reinicia en cada render!  
  
function MiComponente() {  
  function incrementar() {  
    contador++; // No actualiza la UI  
    console.log(contador); // Muestra el cambio  
  }  
  return
```

+

;}

Solución con useState

El estado mantiene los valores entre renderizados y actualiza automáticamente la interfaz cuando cambia.

```
function MiComponente() {  
  const [contador, setContador] = useState(0);  
  
  function incrementar() {  
    setContador(contador + 1); // Actualiza la UI  
  }  
  return (  
    <>
```

Contador: {contador}

+

;}

El estado es esencial para cualquier aplicación interactiva: formularios, contadores, filtros, autenticación, y cualquier situación donde necesites "recordar" información.



¿Qué son los Eventos?

Los eventos son la puerta hacia la interactividad

Los eventos en React son las respuestas a las acciones que realiza el usuario en nuestra aplicación, como:

Interacciones del Mouse

- **onClick**: Al hacer clic
- **onMouseOver**: Al pasar el cursor
- **onMouseOut**: Al retirar el cursor

Interacciones del Teclado

- **onKeyDown**: Al presionar tecla
- **onKeyUp**: Al soltar tecla
- **onKeyPress**: Al mantener tecla

Interacciones de Formulario

- **onChange**: Al cambiar valor
- **onSubmit**: Al enviar formulario
- **onFocus**: Al seleccionar campo

React utiliza sintaxis camelCase para nombrar eventos, a diferencia del HTML tradicional.

Manejo de Eventos en React

Características Principales

- Los eventos se escriben en camelCase: `onClick` en lugar de `onclick`
- Pasamos funciones como manejadores, no strings
- Debemos prevenir comportamientos predeterminados explícitamente con `preventDefault()`
- No necesitamos `addEventListener` como en JavaScript puro

❗ Los eventos en React son eventos sintéticos, una envoltura cross-browser alrededor de los eventos nativos del navegador.

Ejemplo de onClick

```
function Boton() {  
  function handleClick(e) {  
    // 'e' es el evento sintético  
    e.preventDefault();  
    alert('¡Botón clickeado!');  
  }  
  
  return (  

```

Haz click

); }



Ejemplo de onChange e Inputs

Definimos el estado para almacenar el valor

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

Creamos funciones manejadoras para los cambios

```
function handleNombreChange(e)  
{  
  setNombre(e.target.value);  
}  
  
function handleCiudadChange(e)  
{  
  setCiudad(e.target.value);  
}
```

Conectamos los inputs con el estado mediante onChange

```
return (  

```

Selecciona una ciudad Ciudad de
México Guadalajara Monterrey

Hola {nombre} de {ciudad}


```
);
```

Ejercicio: Contador y Mini Formulario

Objetivo del Ejercicio

Aplicar lo aprendido creando:

1. Un contador interactivo con botones de incremento y decremento
2. Un formulario simple que muestre en tiempo real lo que el usuario escribe
3. Una alerta que se active al enviar el formulario

 **Recordatorio:** Para evitar comportamientos predeterminados del formulario, usa `e.preventDefault()` en el manejador del evento `onSubmit`.

Estructura Base

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

function EjercicioContador() {
  // Código del contador aquí
}

function EjercicioFormulario() {
  // Código del formulario aquí
}

function App() {
  return (

  );
};
```



Resumen y Preguntas

Estado en React El estado es la memoria de los componentes que permite almacenar y actualizar datos que afectan el renderizado sin recargar la página.	1
Hook useState Proporciona una forma sencilla de declarar y actualizar el estado en componentes funcionales con la sintaxis <code>[valor, setValor] = useState(valorInicial).</code>	2
Eventos en React Permiten capturar las interacciones del usuario (clicks, escritura, etc.) y responder a ellas mediante funciones manejadoras.	3
	4

Aplicación Práctica

La combinación de useState y eventos es la base de toda interactividad en React, desde formularios hasta aplicaciones complejas.

Próximos Pasos

- Practicar con el ejercicio propuesto
- Explorar múltiples estados en un componente
- Investigar el hook useEffect para efectos secundarios

Cdtlyec

ABOUT US

SERVICES

CASE STUDIES

CONTACT

CONTACT



¿Alguna duda?





Muchas gracias.

Nuestras Redes

www.generaciont.org

www.streambe.com

www.instagram.com/generaciont_ar

www.tiktok.com/@generaciont

generaciont@generaciont.org

Cel: [11 61331747](tel:1161331747)



[11 6133-1747](tel:1161331747)



GENERACIÓN T

generaciont@generaciont.org



GENERACIÓN T