



Buenos  
Aires  
Ciudad

Agencia de Habilidades  
para el Futuro

<Talento  
Tech />

# React JS

---

**Clase 08 | Introducción a Context API**

# ¡Les damos la bienvenida!



Vamos a comenzar a grabar la clase.

## Índice

---

### Rutas Dinámicas y Protegidas

- Creación de rutas dinámicas (detalle de productos).
- Implementación de rutas protegidas (carrito, administración de productos).
- Redirección de usuarios no autenticados.

### Introducción a Context API

- **Creación de Context API para el manejo de estado global.**
- **Uso de useContext para compartir datos entre componentes.**
- **Implementación del estado global para el carrito de compras.**

### Autenticación de usuarios

- Introducción a la Autenticación de Usuarios
- Implementación de formulario de login.
- Manejo de autenticación con tokens (simulada).
- Protección de rutas usando Context API para la autenticación.

# Objetivos de la Clase

**1** Comprender qué es Context API y cuándo es útil.

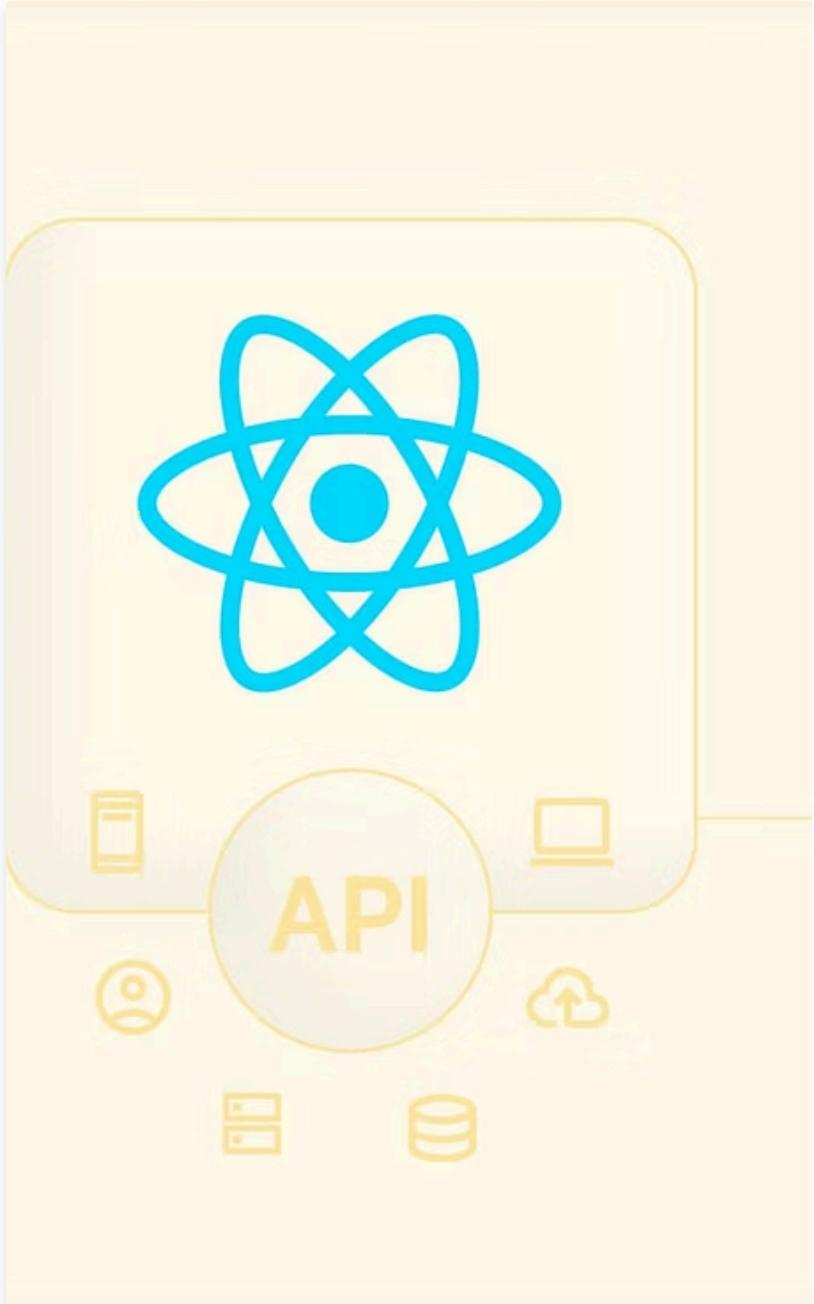
**2** Crear un contexto global para manejar el estado compartido en una aplicación React.

**3** Utilizar el hook useContext para acceder y actualizar datos globales desde diferentes componentes.

**4** Implementar un estado global para gestionar el carrito de compras.

# Context API





# ¿Qué es Context API?

---

Herramienta integrada en React para manejar estados globales sin pasar props manualmente entre componentes.

## Uso Ideal

Para compartir datos entre componentes sin relación directa en el árbol de la aplicación.

# ¿Cuándo usar Context API?

---

## 1 Acceso Múltiple

Cuando varios componentes necesitan acceder al mismo estado.



## 2 Evitar Prop Drilling

Para no pasar props a través de múltiples niveles de componentes.

## 3 Estados Globales

Ideal para manejar autenticación o carritos de compras.

# **Creación de Context API**

---

**Paso 1: Crear contexto**

```
import React, { createContext, useState } from 'react';
// Crear el contexto
export const CarritoContext = createContext();
// Proveedor del contexto
export function CarritoProvider({ children }) {
  const [carrito, setCarrito] = useState([]);
  const agregarAlCarrito = (producto) => {
    setCarrito([...carrito, producto]);
  };
  const vaciarCarrito = () => {
    setCarrito([]);
  };
  return (
    <CarritoContext.Provider value={{ carrito, agregarAlCarrito, vaciarCarrito }}>
      {children}
    </CarritoContext.Provider>
  );
}
```

## Paso 2: Envolver tu aplicación con el proveedor del contexto

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { CarritoProvider } from './context/CarritoConext';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <CarritoProvider>
      <App />
    </CarritoProvider>
  </React.StrictMode>
);
```

# **Uso de useContext**

---

# Uso de useContext

---

## 1 Importar useContext

Importar el hook useContext de React en el componente.

## 2 Importar Contexto

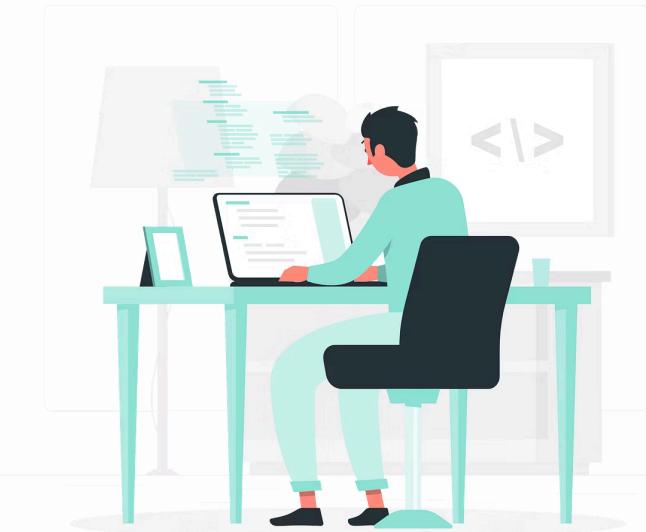
Importar el contexto creado (CarritoContext).

## 3 Consumir Contexto

Usar useContext para acceder a los valores del contexto.

## 4 Utilizar Datos

Usar los datos y funciones del contexto en el componente.



# Componente Carrito

```
import React, { useContext } from 'react';
import { CarritoContext } from '../context/CarritoContext';
function Carrito() {
  const { carrito, vaciarCarrito } = useContext(CarritoContext);
  return (
    <div>
      <h1>Carrito de Compras</h1>
      {carrito.length > 0 ? (
        <ul>
          {carrito.map((producto, index) => (
            <li key={index}>{producto.nombre} - ${producto.precio}</li>
          )))
        </ul>
      ) : (
        <p>El carrito está vacío.</p>
      )}
      {carrito.length > 0 && <button onClick={vaciarCarrito}>Vaciar Carrito</button>}
    </div>
  );
}
```

```
export default Carrito;
```

# Componente Producto

```
import React, { useContext } from 'react';
import { CarritoContext } from '../context/CarritoContext';

function Producto({ producto }) {
  const { agregarAlCarrito } = useContext(CarritoContext);
  return (
    <div>
      <h2>{producto.nombre}</h2>
      <p>Precio: ${producto.precio}</p>
      <button onClick={() => agregarAlCarrito(producto)}>Agregar al Carrito</button>
    </div>
  );
}
```

```
export default Producto;
```

# Implementación del Estado Global

---

# Implementación del Estado Global

---



## Crear Contexto

Definir el contexto y proveedor para el carrito.

## Envolver App

Aplicar el proveedor al componente principal.

## Usar Contexto

Implementar useContext en componentes relevantes.

## Gestionar Estado

Manejar el carrito a través del estado global.

# Componente Principal (App)

```
import React from 'react';
import Producto from './Producto';
import Carrito from './Carrito';

function App() {
  const productos = [
    { id: 1, nombre: 'Producto 1', precio: 100 },
    { id: 2, nombre: 'Producto 2', precio: 200 },
    { id: 3, nombre: 'Producto 3', precio: 300 },
  ];

  return (
    <div>
      <h1>Tienda Online</h1>
      <div style={{ display: 'flex', justifyContent: 'space-between' }}>
        <div>
          <h2>Productos</h2>
          {productos.map((producto) => (
            <Producto key={producto.id} producto={producto} />
          )))
      
```

```
        </div>
        <Carrito />
    </div>
</div>
);
}

export default App;
```

# Reflexión Final

---

Aprendimos a usar Context API para manejar estados globales en aplicaciones React. Vimos cómo crear y consumir un contexto con `useContext`, y cómo implementar el estado global para gestionar un carrito de compras.

El uso de Context API nos ayuda a mantener una arquitectura más limpia y escalable, especialmente en aplicaciones complejas donde múltiples componentes necesitan compartir información. Esta herramienta es fundamental para crear aplicaciones React modernas y eficientes.



# Preguntas para Reflexionar

## 1 Ventajas de Context API

¿Qué ventajas tiene usar Context API frente a pasar props entre componentes?

## 2 Context API vs Redux

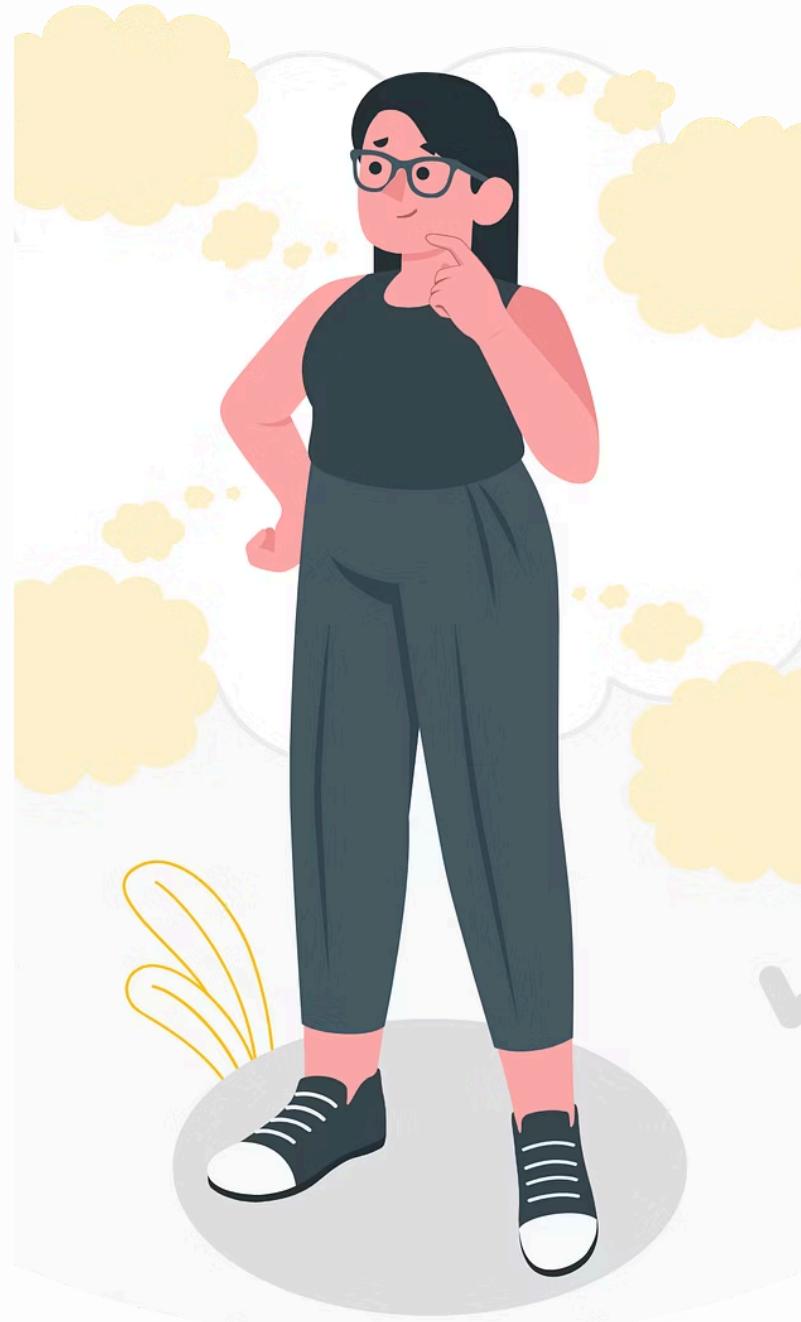
¿En qué situaciones sería más conveniente usar Redux en lugar de Context API?

## 3 Múltiples Contextos

¿Qué problemas podrías enfrentar al manejar múltiples contextos en una misma aplicación?

## 4 Optimización

¿Cómo podrías optimizar el rendimiento de una aplicación que usa Context API intensivamente?



# Próximos Pasos

---



1

## Introducción a la Autenticación de Usuarios

Qué es y cómo cuidamos nuestro sitio de personas no registradas

2

## Formulario de Login

Implementar un formulario de inicio de sesión funcional.

3

## Autenticación con Tokens

Manejar la autenticación usando tokens (simulada).

4

## Protección de Rutas

Usar Context API para la autenticación en rutas protegidas.

# Materiales y Recursos Adicionales

---



## Documentación

[Documentación oficial de Context API](#)



## Guía

[Context API en tus proyectos](#)



# Pre-Entrega de Proyecto

---



# Pre-Entrega de Proyecto

Obligatorio



¡Hemos llegado a un momento clave! Es el momento de realizar la pre-entrega del proyecto trabajado hasta el momento. Como es de costumbre en varias empresas, nos toca pronto juntarnos con nuestro cliente para ver como va creciendo su sitio web.

A continuación el equipo te detallará lo que se espera tener en esta primera presentación del sitio a nuestro cliente:



# Pre-Entrega de Proyecto

Obligatorio

---

**Requerimientos del Proyecto**

**Requerimiento #1:** Crear una funcionalidad básica para el manejo de un carrito de compras.



- 1** Crear un componente para listar los productos disponibles.
- 2** Usar el hook `useState` para manejar el estado del carrito.
- 3** Implementar un evento de clic que permita agregar productos al carrito.
- 4** Mostrar el carrito con los productos seleccionados en otro componente.
- 5** Crear un Layout del eCommerce



## Pre-Entrega de Proyecto

Obligatorio

---

### Requerimientos del Proyecto

**Requerimiento #2:** Conectar la aplicación a una API que provea información sobre los productos.

**1**

Integración con una API

**2**

Estado de carga y errores

**3**

Gestión del estado con useState

**4**

Actualizar el diseño del eCommerce

**5**

Manejo de efectos secundarios con  
useEffect

**6**

Ampliación del carrito



# Pre-Entrega de Proyecto

Obligatorio

## Requerimientos del Proyecto

**Requerimiento #3:** Integración de rutas



**1** Implementación de rutas

**3** Crear componente para cada sección

**2** Estado de carga y manejo de errores

**4** Navegar entre productos

# ⚙️ Pre-Entrega de Proyecto

Obligatorio

## Requerimientos del Proyecto

**Requerimiento #4:** Implementar rutas dinámicas y protegidas



**1** Rutas Dinámicas

**3** Rutas Protegidas

**2** Interactividad

**4** Navbar





# Pre-Entrega de Proyecto

Obligatorio

---



- ⓘ Esta pre-entrega nos va a permitir estar más en sintonía con las necesidades de nuestro cliente esperando su feedback y nos comentará por donde seguir.

¿Estás listo? ¡Confiamos en que harás un gran trabajo!



