Programacion III
Profe : Chocobar, Matias Sebastian

Clase de Introducción a React con CRA

Clase de Introducción a React con CRA (Create React App)

✨1. Introducción

React es una biblioteca de JavaScript (JS) creada por Facebook en 2013. Su objetivo es facilitar la creación de interfaces de usuario (UI) interactivas, reutilizables y rápidas, principalmente en aplicaciones web de una sola página (SPA).

Emplea JSX, una extensión de sintaxis que permite escribir código similar a HTML dentro de JavaScript.
Create React App (CRA) proporciona una configuración inicial para desarrollar SPA sin configuración manual.

En una SPA, la navegación no recarga la página, y el DOM se actualiza de forma eficiente según el estado.

```jsx
import React, { useState } from 'react';
import './App.css';

function App() {
  const [count, setCount] = useState(0);
  return (
    <div className="app">
      <h1>Contador: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Incrementar</button>
    </div>
  );
}

export default App;
```

Ejemplo de Componente con estado local que incrementa un contador al hacer clic.

Beneficios respecto a HTML/CSS/JS tradicionales:

Componentes reutilizables.

DOM Virtual: mejor rendimiento.

Mayor organización y mantenibilidad.

Programación declarativa.

Programacion III
Profe : Chocobar, Matias Sebastian

 2. Instalación de Node.js y CRA

✅Instalar Node.js:

Descargar desde: https://nodejs.org/ Verificar instalación:

node -v

npm -v

¿Qué es npm?

npm significa:

Node Package Manager (Es el gestor de paquetes que viene junto con Node.js.)

¿Para qué sirve npm?

Instalar librerías (como React, Express, etc.)

Manejar versiones de paquetes

Ejecutar scripts definidos en el package.json

Crear y publicar tus propios paquetes

Ejemplos de uso:

```
npm install react          # Instala react en tu proyecto
npm install -g nodemon     # Instala nodemon globalmente
npm run start              # Ejecuta un script del package.json
```

⚡ ¿Qué es npx?

npx significa:

Node Package eXecute

Es una herramienta que también viene con npm (desde la versión 5.2 en adelante).

Programacion III
Profe : Chocobar, Matias Sebastian

¿Para qué sirve npx?

Te permite ejecutar paquetes de npm sin instalarlos globalmente. Muy útil para herramientas que usás una sola vez, como generadores de proyectos.

Ejemplo típico:

```
npx create-react-app mi-app
```

Este comando:

Descarga temporalmente create-react-app

Ejecuta el generador

Y después descarta ese paquete (a menos que ya esté en caché)

Comparación resumida

Herramienta	Significa	Uso principal
`npm`	Node Package Manager	Instalar y gestionar paquetes
`npx`	Node Package Execute	Ejecutar paquetes sin instalarlos global

¿Qué es create-react-app?

Es una herramienta que permite crear proyectos React con una estructura preconfigurada, sin tener que configurar Webpack, Babel, etc.

✅Forma moderna y recomendada:

```
npx create-react-app nombre_proyecto
```

 No requiere instalación previa.

Programacion III
Profe : Chocobar, Matias Sebastian

 Usa la última versión directamente desde el repositorio.

 Recomendado por el equipo oficial de React.

 Forma antigua (no recomendada hoy):

```
npm install -g create-react-app
create-react-app nombre_proyecto
```

Instala el generador de forma global.

Puede quedar desactualado si no se actualiza manualmente.

En entornos compartidos (laboratorios, universidades) puede traer problemas de versiones.

¿Qué pasa si hacés ambas?

```
npm install -g create-react-app
npx create-react-app nombre_proyecto
```

 Técnicamente no hay error, pero el npx podría usar la versión global, lo que anula el beneficio de mantenerlo actualizado automáticamente.

 No aporta valor extra instalarlo globalmente si se usa con npx.

 Creación de Proyecto React con CRA (Create React App)

Comandos básicos para iniciar el proyecto:

npx create-react-app nombre_proyecto # Crea el proyecto React con configuración inicial

cd nombre_proyecto # Ingresa al directorio del proyecto

npm start # Inicia la aplicación en el navegador (localhost:3000)

Esto levanta un servidor local en http://localhost:3000

Programacion III
Profe : Chocobar, Matias Sebastian

Estructura recomendada del proyecto:

```
nombre_proyecto/
├── public/
│   └── index.html
├── src/
│   ├── assets/                # Recursos (imágenes, íconos, etc.)
│   ├── components/            # Componentes reutilizables
│   │   ├── Header.jsx
│   │   ├── Main.jsx
│   │   └── Footer.jsx
│   ├── pages/                 # Páginas principales (componentes hijos de App.js)
│   │   ├── Home.jsx
│   │   └── Login.jsx
│   ├── css/                   # Archivos de estilos CSS para cada componente
│   │   ├── header.css
│   │   ├── main.css
│   │   └── footer.css
│   ├── constants/             # Constantes generales (colores, rutas, textos)
│   ├── App.js                 # Componente raíz principal
│   ├── index.js               # Punto de entrada de la aplicación React
│   └── reportWebVitals.js
├── package.json
└── README.md
```

3. Estructura del Proyecto

Archivos principales:

- **index.js:** punto de entrada. Renderiza <App /> en el DOM.
- **App.js**: componente principal de la app.
- **App.test.js:** contiene tests automáticos (usualmente con Jest).
- **reportWebVitals.js:** mide rendimiento.
- **setupTests.js:** configura tests.
- **package.json:** dependencias y scripts.
- **package-lock.json:** exactas versiones instaladas.

Carpetas:

- **public/:** archivos estáticos (HTML, favicon, etc.).
- **src/:** código fuente React (JSX, CSS, componentes).
- **node_modules/:** todas las dependencias descargadas.

Conceptos clave en React

¿Qué es un componente?

Programacion III
Profe : Chocobar, Matias Sebastian

Un componente es una función o clase que retorna JSX.

Es la unidad básica de construcción en React.

Reutilizable, aislado, y puede tener estado (state).

¿Cómo se monta un componente?

Se monta al llamarlo dentro de otro componente, por ejemplo en App.js:

```
import Header from './components/Header';
function App() {
  return <Header />;
}
```

Ciclo de vida de un componente (en componentes de clase):

Montaje (mounting): constructor(), render(), componentDidMount()

Actualización (updating): componentDidUpdate()

Desmontaje (unmounting): componentWillUnmount()

 En componentes funcionales(son lo que usaremos a lo largo de la materia**) se reemplaza con Hooks.**

¿Por qué los componentes se escriben con mayúscula?

Porque React distingue entre elementos HTML ( ) y componentes (<Header>).

Componentes deben comenzar con mayúscula para que React los reconozca como tales.

Sugerencias:

Usar PascalCase: MiComponente.js

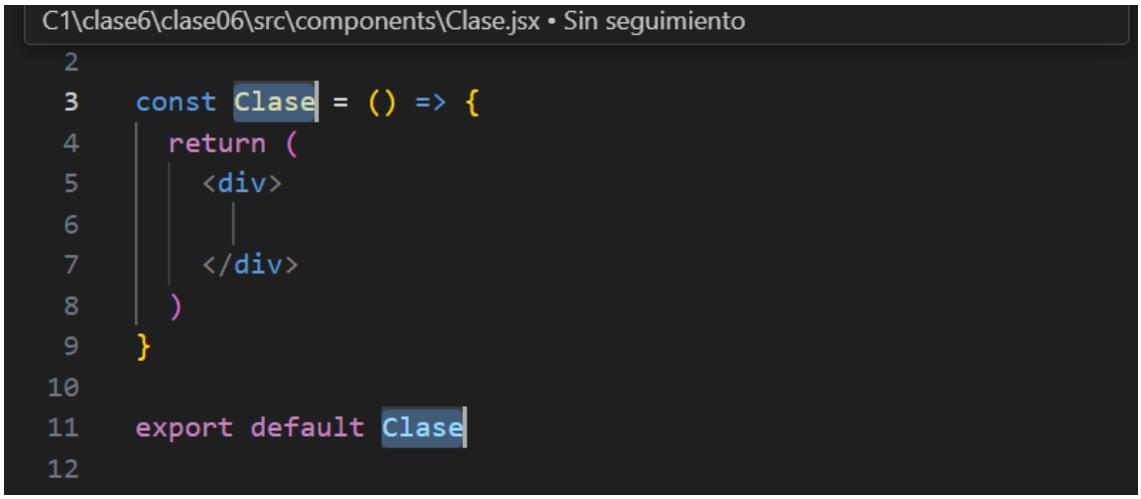
Archivos dentro de /src/components/

Separar lógica, estilo y test por archivo

Atajos comunes (Snippets - usando extensión ES7+ en VSCode):

rafce: React Arrow Function Component Export

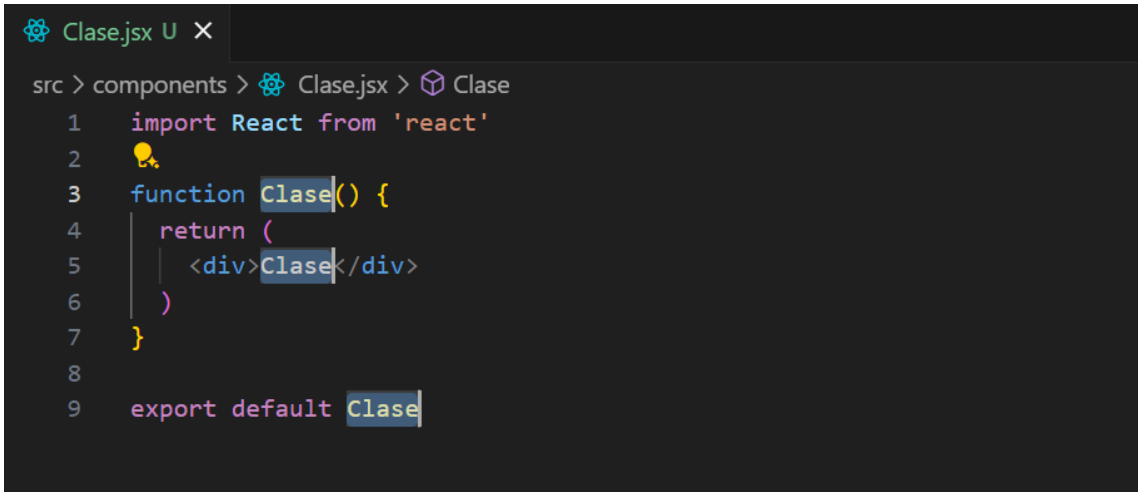
Programacion III
Profe : Chocobar, Matias Sebastian



rfc: React Functional Component

rfce: React Function Component Export

Programacion III
Profe : Chocobar, Matias Sebastian



sfc: Statless Function Component (crea la function del componente la cual queda a la espera que le coloquen el nombre del componente)

JSX vs JS

JSX es una extensión de sintaxis de JavaScript que permite escribir HTML dentro del JS.

JSX no es obligatorio pero mejora la legibilidad y productividad.

JSX se transforma en código JS puro con React.createElement().

¿Qué es App.js?

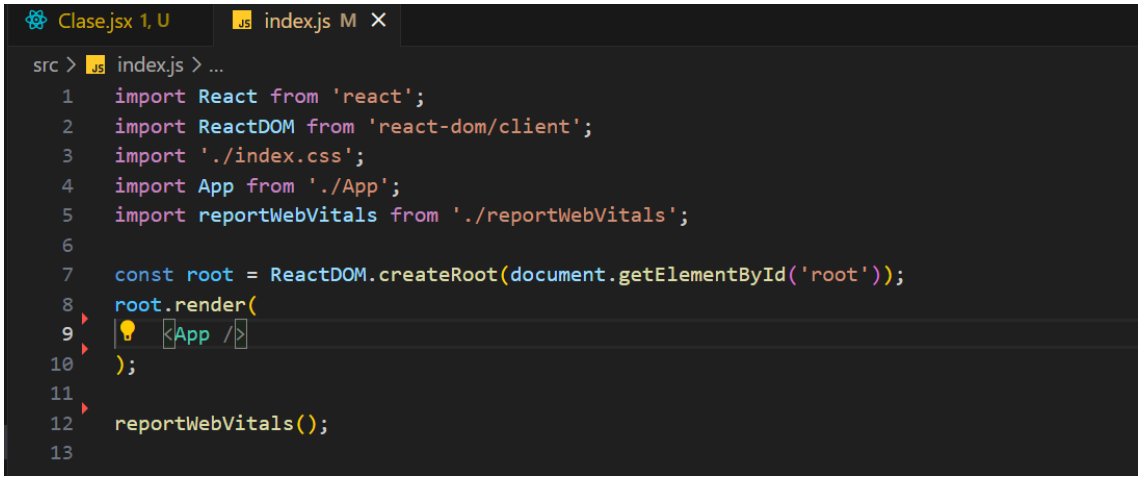
Es el componente principal donde se estructuran y montan los componentes hijos y rutas.

Es el centro de la jerarquía de componentes.

4. Relación con el DOM

El DOM (Document Object Model) representa la estructura de una página HTML. React trabaja con un DOM virtual que mejora el rendimiento actualizando solo lo necesario en el navegador.

// index.js



Programacion III
Profe : Chocobar, Matias Sebastian

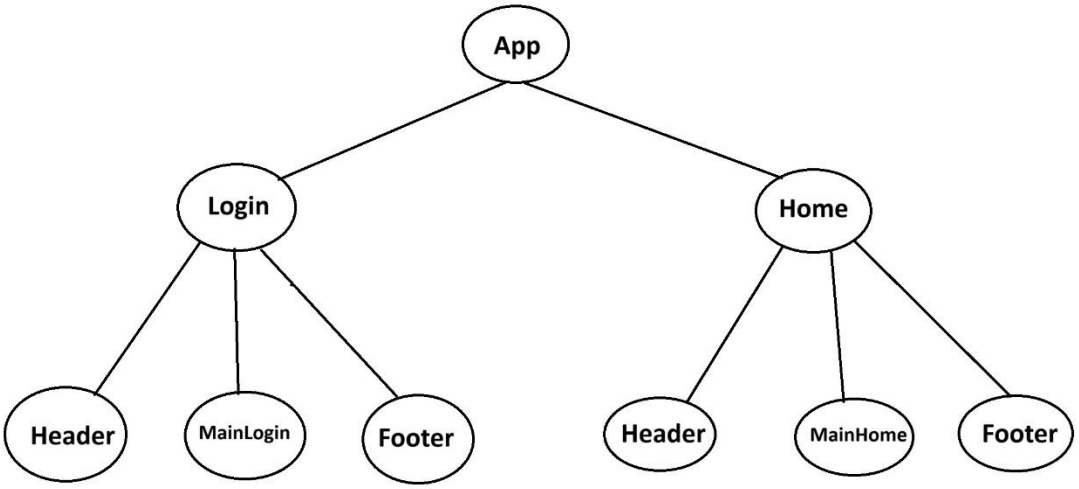
Virtual DOM

Es una **copia ligera del DOM real** en memoria.

React compara el Virtual DOM con el real y solo actualiza lo necesario.

Mejora el rendimiento.

diagrama jerárquico del Virtual DOM típico en una aplicación React.



¿Qué representa?

App como componente raíz.

Componentes hijos como **Home, Login**, y dentro de estos, **Header, Main, Footer**, etc.

Las líneas de conexión muestran la **estructura padre-hijo**, como un árbol o DOM virtual.

Narrativa de componentes

Padre → Hijo → Nieto → Bisnieto

Los datos fluyen de **arriba hacia abajo** mediante props.

Props

props (propiedades) son datos enviados desde un componente padre a uno hijo.

Ejemplo de props pasando de componente padre App.js a componente hijo Clase.jsx

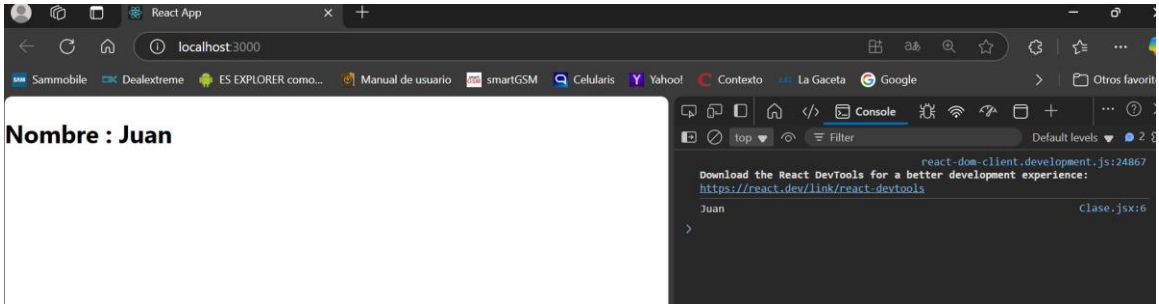
Programacion III
Profe : Chocobar, Matias Sebastian

Clase.jsx U

src > components > Clase.jsx > Clase

```jsx
import React from 'react'


const Clase = (props) => {

console.log(props.nombre)

  return (

    <div>

    <h2>Nombre : {props.nombre}</h2>

    </div>
  )
}

export default Clase

```

De esta forma se muestra en el navegador



Los props son objetos que se manda de componente padre a hijo, por lo tanto es posible hacer destructuring.

¿Que significa destructuring? Es una expression que te permite extraer o “destructurar” datos desde estrcuturas de datos como arreglos, objetos, mapas y sets y crear nuevas variables con ese dato en particular.

Los objetos se destructuran por propiedades, usando las que necesiten

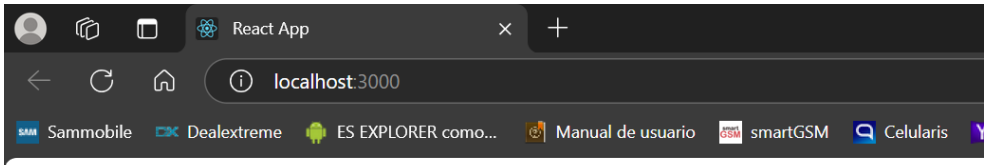
Programacion III
Profe : Chocobar, Matias Sebastian

Clase.jsx U

src > components > Clase.jsx > default

```jsx
import React from 'react'

const Clase = () => {
//Destructuring de objeto
//objeto persona
const persona = {
    nombre: 'Juan',
    edad: 30,
    ciudad: 'Madrid'
}
//Destructuring del objeto persona
const { nombre, edad, ciudad } = persona

  return (
    <div>
      <h2>Objeto sin Destructurar</h2>
      <br />
      <li>Nombre : {persona.nombre}</li>
      <li>Edad : {persona.edad}</li>
      <li>Ciudad : {persona.ciudad}</li>
      <br />
      <h2>Objeto con Destructurar</h2>
      <br />
      <li>Nombre: {nombre}</li>
      <li>Edad: {edad}</li>
      <li>Ciudad: {ciudad}</li>
      <br />
    </div>
  )
}
export default Clase
```

De esta forma se muestra en el navegador



Objeto sin Destructurar

- Nombre : Juan
- Edad : 30
- Ciudad : Madrid

Objeto con Destructurar

- Nombre: Juan
- Edad: 30
- Ciudad: Madrid

Programacion III
Profe : Chocobar, Matias Sebastian

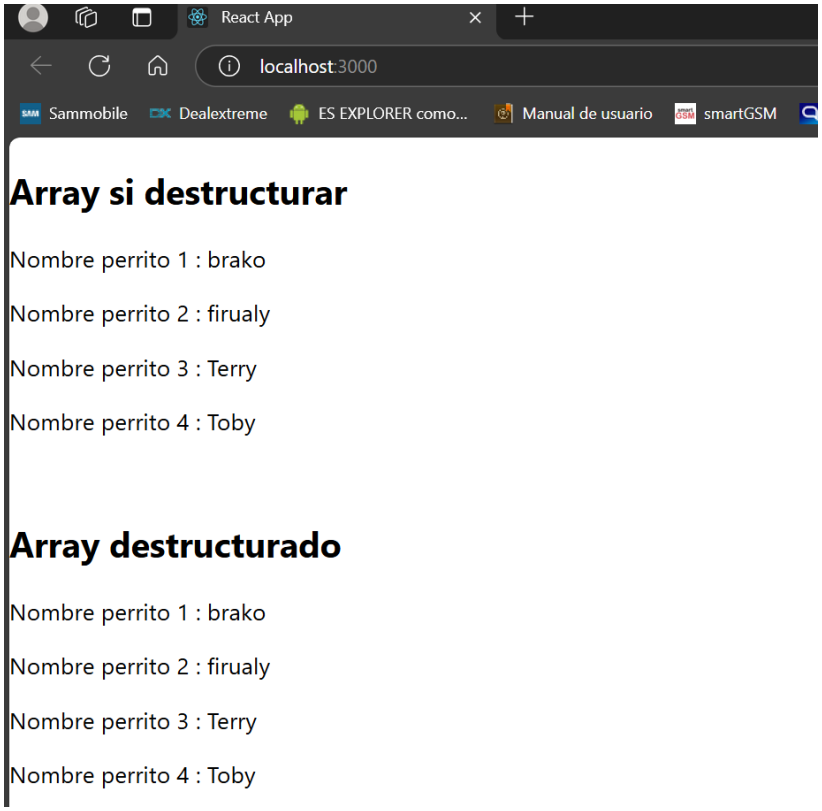
Los arrays se destructuran por posicion.

Clase.jsx U

src > components > Clase.jsx > Clase

```jsx
import React from 'react'

const Clase = () => {
//Destructuring de un array
//Arrays sin destructurar
const perrito = ['brako', 'firualy', 'Terry', 'Toby']
//Arrays destructurados
const [perrito1, perrito2, perrito3, perrito4] = perrito

  return (
    <div>
    <h2>Array si destructurar</h2>
    <p>Nombre perrito 1 : {perrito[0]}</p>
    <p>Nombre perrito 2 : {perrito[1]}</p>
    <p>Nombre perrito 3 : {perrito[2]}</p>
    <p>Nombre perrito 4 : {perrito[3]}</p>
    <br />
    <h2>Array destructurado</h2>
    <p>Nombre perrito 1 : {perrito1}</p>
    <p>Nombre perrito 2 : {perrito2}</p>
    <p>Nombre perrito 3 : {perrito3}</p>
    <p>Nombre perrito 4 : {perrito4}</p>
    </div>
  )
}
export default Clase
```

De esta forma se muestra en el navegador



Programacion III
Profe : Chocobar, Matias Sebastian

Retomando los props Podemos destructurar de la misma forma como se muesta en la imagen

```
const Saludo = ({ nombre }) => <h1>Hola {nombre}</h1>;
<Saludo nombre="Matías" />S
```

Props Drilling

Ocurre cuando se pasan props de componente en componente hasta el destino.

Solución: Context API o Redux para evitar propagar datos innecesariamente.

Hooks principales

Teoría useState React JS

¿Qué es useState?

Es un hook propio de React que te permite agregar estado local a un componente funcional. El estado es información que cambia a lo largo del tiempo o en respuesta a interacciones del usuario.

¿Qué hace useState?

useState te permite declarar una variable que React "vigila", de modo que cuando su valor cambia, el componente se vuelve a renderizar con el nuevo valor. Es una forma de almacenar y actualizar datos dinámicos dentro de un componente.

Ciclo de vida y useState

Cuando se monta un componente, useState inicializa su valor. Cada vez que se actualiza, React vuelve a renderizar el componente. No hay fase explícita de "desmontaje" con useState, pero puede combinarse con useEffect para gestionar esos casos.

useState: Permite tener variables reactivas (estado)

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

¿Cómo se utiliza useState?

Programacion III
Profe : Chocobar, Matias Sebastian

```jsx
import { useState } from "react";

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

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

Teoría de `useState` – Forma destructurada

Recuerden que `useState` devuelve un **array**, donde:

- La **primera posición** representa el **valor actual del estado**.
- La **segunda posición** es una **función que permite actualizar ese valor**.

Esta sintaxis se llama **desestructuración de arrays**.

¿Cómo se inicializa?

Dentro de los paréntesis de `useState` colocamos el **valor inicial** del estado.

Este valor puede ser **muy versátil**, ya que `useState` acepta múltiples tipos de datos:

- `useState(0)` → número
- `useState("")` → string vacío
- `useState(true)` → booleano
- `useState([])` → array
- `useState({})` → objeto
- `useState([{ id: 1, nombre: "Matías" }])` → array de objetos

Ejemplo:

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

contador es el valor actual.

setContador es la funcion que lo actualiza(puede mutar ese valor).

```
setContador(contador + 1); // actualiza el valor
```

Teoría UseEffect React JS

¿Qué es useEffect?

Es un hook(gancho) propio de react que te permite realizar efectos secundarios en un componente funcional. Los efectos secundarios son operaciones que ocurren fuera del flujo normal de la representación del componente, como obtener datos de una API, configurar suscripciones o modificar el DOM

¿Que es lo que hace useEffect?

Este hook nos permite definir efectos. Los efectos nos permiten ejecutar un trozo de código según el momento en el que se encuentre el ciclo de vida de nuestro componente.

Los ciclos de vida de un componente son:

Montaje (Mount)

Actualización (Update)

Desmontaje (Unmount)

Como se utiliza useEffect

Su uso es relativamente fácil de explicar: useEffect se ejecuta cuando el componente se monta, cada vez que cambia alguna parte del estado del componente y finalmente cuando el componente se desmonta. Una de las claves del correcto uso de useEffect está en evitar que se ejecute con cada renderizado.

¿Cuántos useEffect puedo usar en React?

Aunque normalmente los componentes de React solo cuentan con un useEffect lo cierto es que podemos tener tantos useEffect como queramos en un componente. Cada uno de ellos se ejecutará cuando se renderice el componente o cuando cambien las dependencias del efecto.

useEffect: Ejecuta código en ciertos momentos del ciclo de vida (al montar, actualizar o desmontar)

Teoría del `useEffect` – Estructura y dependencias

El hook `useEffect` se compone de **dos elementos principales**:

1. **Un callback (función)**:
   Dentro de esta función colocamos el **código que queremos ejecutar** cuando ocurra un efecto. Puede ser una llamada a API, un `console.log`, un cambio en el DOM, etc.

2. **Un array de dependencias** (segunda posición):
   Este array indica **cuándo debe ejecutarse el callback**. Actúa como un "observador" de variables o props.

Tipos de uso del array de dependencias

Sintaxis	Comportamiento
`useEffect(() => { ... })`	Se ejecuta **después de cada renderizado**
`useEffect(() => { ... }, [])`	Se ejecuta **una sola vez** al montar el componente
`useEffect(() => { ... }, [x])`	Se ejecuta **cada vez que x cambia**

Ejemplo

```
useEffect(() => {
  console.log("Se ejecuta solo una vez al montar");
}, []);
```

```
useEffect(() => {
  console.log("Se ejecuta cada vez que cambia el contador");
}, [contador]);
```

Eventos

```
<button onClick={() => alert('¡Clic!')}>Haz clic</button>
```

React Router DOM

Permite navegación entre páginas sin recargar el navegador.

Principales componentes:

Link: Enlace que reemplaza el <a> tradicional

```
<Link to="/login">Login</Link>
```

useNavigate: Hook para navegar programáticamente

```
const navigate = useNavigate();
navigate('/home');
```

useParams: Hook para obtener parámetros de la URL

```
const { id } = useParams();
```

React Bootstrap

Biblioteca de componentes con estilos de Bootstrap adaptados a React.

```
npm install react-bootstrap bootstrap
```

Ejemplo de uso:

```
import { Button } from 'react-bootstrap';
<Button variant="primary">Aceptar</Button>
```

⚡ Comandos útiles

npm start # Levanta el servidor

npm run build # Compila para producción

npm test # Corre los tests

npm install # Instala dependencias

🔗 Relación entre Archivos

index.js llama a **App.js**

App.js contiene componentes

App.test.js prueba funcionalidades

package.json declara dependencias como React

public/index.html contiene el div id="root" donde se monta todo

🔗 Conclusión

React permite separar lógica y presentación, crear interfaces modernas y escalables, y trabajar con herramientas modernas del ecosistema JavaScript.

Ideal para aprender desarrollo frontend profesional.