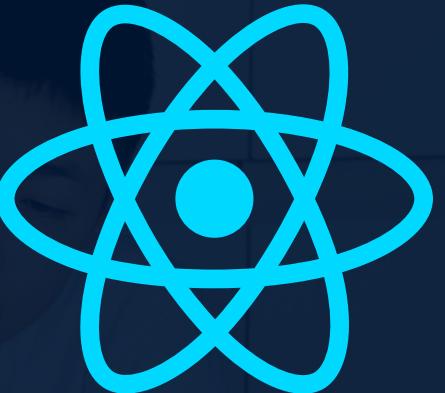


IW

REACT



**Cabral Machuca Alejandro Aaron
Díaz Canales José Lino
Hernández Rodríguez César Uriel
Valdez Ponce Jose Alberto**

ITTEPIC



Historia

¿QUE ES REACT?

React (también conocido como ReactJS) es una biblioteca de JavaScript de código abierto diseñada para construir interfaces de usuario interactivas, especialmente en aplicaciones de una sola página (SPA).



¿POR QUÉ USAR REACT?



La principal ventaja de React es poder generar el DOM ("Modelo de Objetos del Documento", estructura de los elementos que se generan en el navegador web al cargar una página")

Amplia comunidad. Al estar basado en JavaScript, cuenta con una amplia comunidad que dispone de un gran número de librerías externas.

React Native. Facilidad para generar aplicaciones móviles usando el mismo código JavaScript de la aplicación web.

COMPONENTES EN REACT

Los componentes son como las funciones de JavaScript. Aceptan entradas arbitrarias (llamadas “props”) y retornan elementos de React que describen lo que debe aparecer en la pantalla

La forma más sencilla de definir un componente es escribir una función de JavaScript

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

Esta función es un componente de React válido porque acepta un solo argumento de objeto “props” (que proviene de propiedades) con datos y devuelve un elemento de React

ESTADO (STATE) Y PROPS

Los props son un objeto que se pasan como argumentos de un componente padre a un componente hijo

State es un valor que se define dentro de un componente.

```
constructor(props) {
  super(props);
  this.state = {
    street: "Cracker St.108",
    type: "Family House",
    color: "white",
    yearOfConstruction: 1968
  };
}
render() {
  return (
    <div>
      <h1>Introduction Example</h1>
    </div>
  );
}
```

COMPONENT

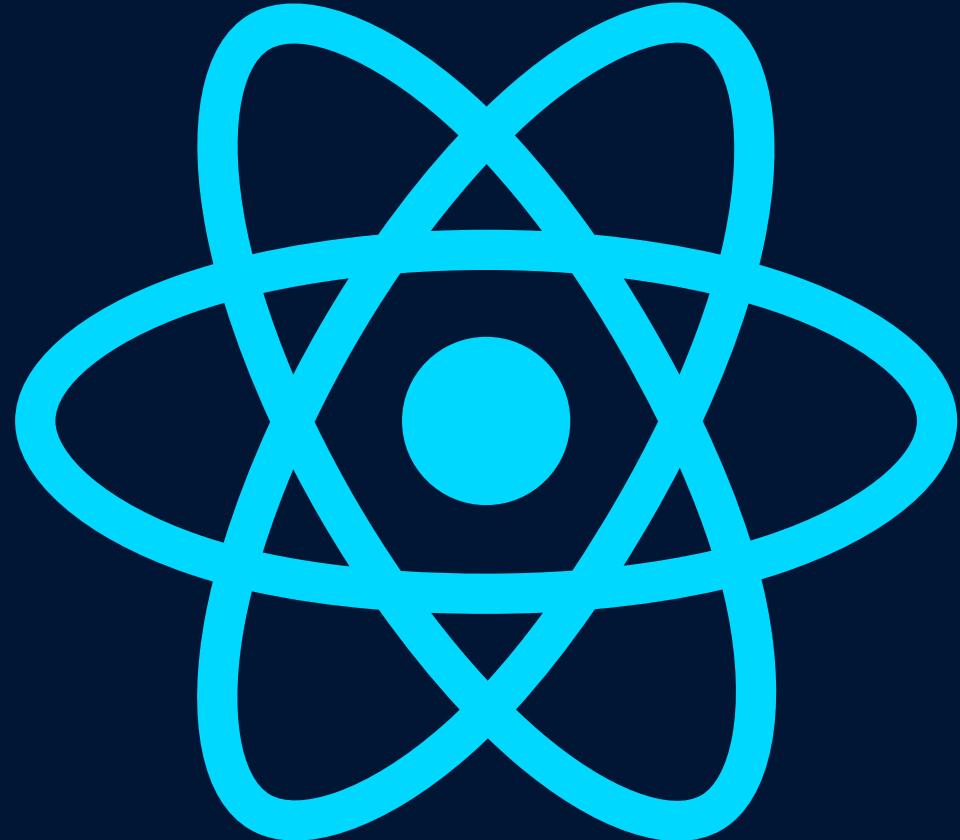


state is us

REACT VS OTRAS TECNOLOGÍAS

Característica	React	Angular	Vue
Tipo	Biblioteca	Framework	Framework
Fundador	Facebook	Google	Comunidad
Curva de aprendizaje	Media	Alta	Baja
Virtual DOM	Sí	No (usa otro método)	Sí

COMO USAR REACT



1. Instalar Node.js y crear un proyecto:

Primero, asegúrate de tener instalado Node.js (versión 14 o superior) y npm (versión ≥ 5.6) en tu sistema. Luego, crea un nuevo proyecto con Create React App (una herramienta oficial que configura todo lo necesario):

```
npx create-react-app mi-app  
cd mi-app  
npm start  
con vite  
npm create vite@latest  
cd vite  
npm install  
npm run dev
```

2. Explorar la estructura básica:

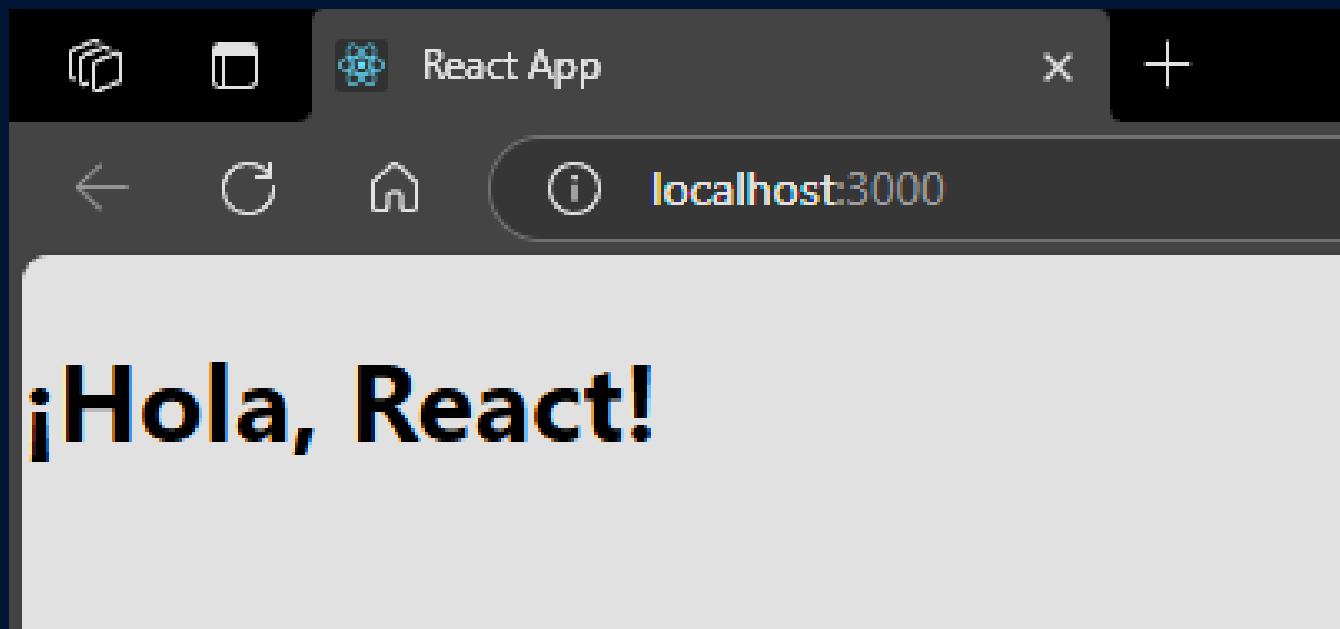
El proyecto generado tiene esta estructura típica

```
mi-app/  
  public/  
    index.html (plantilla HTML principal)  
  src/  
    index.js (punto de entrada JavaScript)  
    App.js (componente principal)  
    App.css (estilos del App)  
    index.css (estilos globales) ...otros archivos...
```

COMO USAR REACT

Luego en src/index.js debes importar y renderizar este componente en el DOM:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
ReactDOM.createRoot(document.getElementById('root')).render(<App />);
```



3. Crear el primer componente: Abre src/App.js y modifica su contenido para probar tu componente. Por ejemplo:

```
import React from 'react';
function App() {
  return <h1>¡Hola, React!</h1>;
}
export default App;
```

Esto mostrará “¡Hola, React!” en la página web. Así se puede comprobar que React está funcionando correctamente.

COMO USAR REACT

Y en App.js

```
import Saludo from './saludos';
function App() {
return (
<div>
<Saludo nombre="Carlos" />
<Saludo nombre="Ana" />
</div>
);
}
```

4. Uso de props: Los componentes pueden recibir datos mediante props. Por ejemplo, crea un componente Saludo.js:

```
import React from 'react';
function Saludo(props) {
return <h2>¡Hola, {props.nombre}!</h2>;
}
export default Saludo;
```

Cada instancia de muestra un mensaje diferente según la prop nombre. Las props permiten parametrizar componentes y utilizarlos con distintos valores.

COMO USAR REACT

```
import React, { useState } from 'react';
function Contador() {
  const [cuenta, setCuenta] = useState(0);
  return (
    <div>
      <p>Has hecho clic {cuenta} veces</p>
      <button onClick={() => setCuenta(cuenta + 1)}>Haz clic</button>
    </div>
  );
}
```

En React, el estado de un componente es simplemente un objeto o variable JavaScript que refleja parte de la interfaz y cambia con la interacción del usuario.

5. Manejo de estado:

Para componentes que necesitan datos cambiantes (por ejemplo, contadores, formularios, peticiones asíncronas), React usa el concepto de estado. Con hooks como useState, puedes definir variables de estado locales en un componente. Por ejemplo, un simple contador:

En este ejemplo, cuenta es el estado (un entero), inicializado en 0. Al presionar el botón se invoca setCuenta para actualizar el estado.