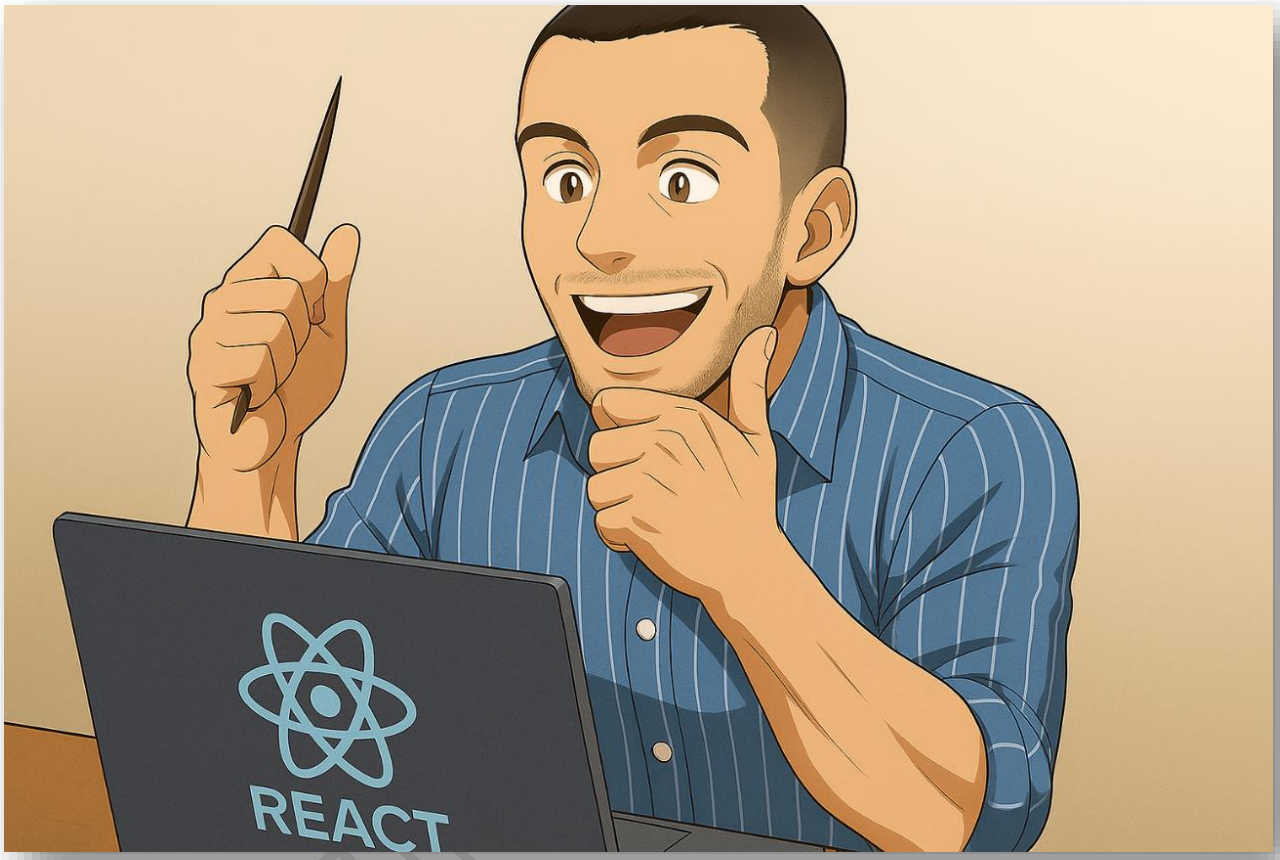
Introducción a React: Qué es, cómo instalarlo y primeros pasos prácticos.



GUIA PRACTICA PARA PRINCIPIANTE

Autor: Titogeremito

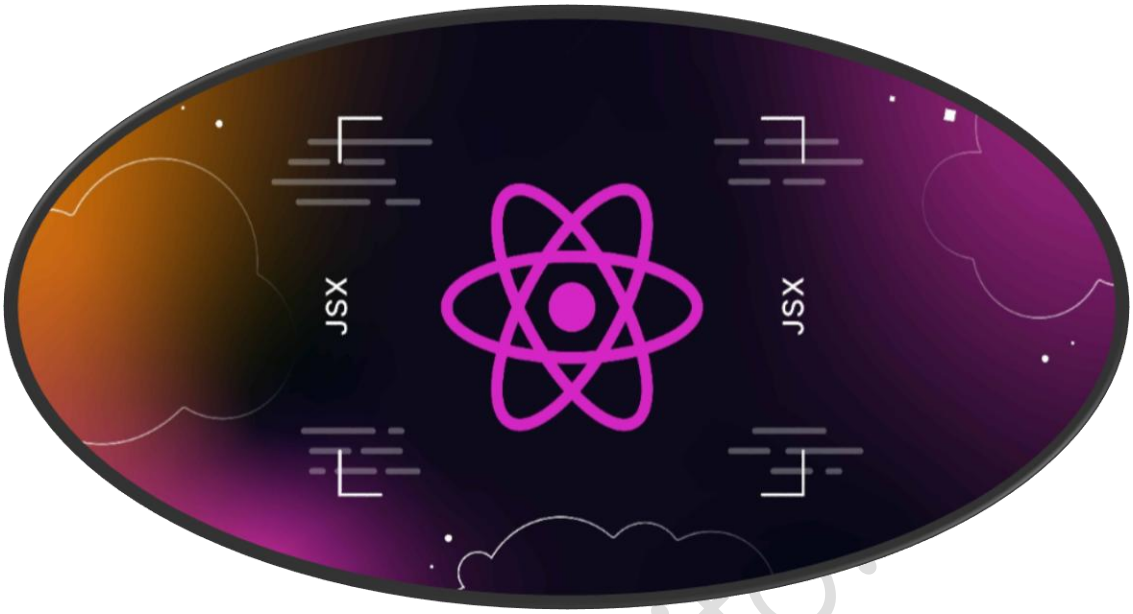
Web: www.titogeremito.com

Explora más contenido gratuito sobre React, desarrollo web y apps.
➞■ Aprende a crear tu primera app React paso a paso

Visita: www.titogeremito.com/recursos-react

Contenido

¿QUE ES REACT?



React es una librería de JavaScript que sirve para crear interfaces web de forma rápida y organizada. Su objetivo es simplificar el trabajo del desarrollador, haciendo más fácil crear páginas web dinámicas sin tener que escribir tanto código complicado en JavaScript puro.

Para lograrlo, React introduce una forma diferente de pensar y trabajar en el desarrollo web. Hace muchas tareas automáticamente por detrás, lo que permite centrarse más en construir que en configurar. Sin embargo, aunque React te ahorre trabajo, **es importante conocer bien los conceptos básicos del desarrollo web**. ***(Leer mis otros Ebooks de HTML, CSS,JS para aprender los principios del desarrollo web)***

Si no entiendes cómo funciona todo por dentro, podrás crear cosas simples, pero te costará avanzar hacia proyectos más complejos o profesionales.

Funciones de React

Cuando se desarrolla una página web simple, usar HTML, CSS y JavaScript por separado suele ser suficiente. Puedes crear una estructura básica, añadir estilos y dar algo de interactividad con JavaScript.

Sin embargo, a medida que el proyecto crece, por ejemplo, si necesitas mostrar contenido dinámico, gestionar formularios, actualizar partes concretas de la página sin recargarla o trabajar en equipo, este enfoque

empieza a mostrar sus límites. El código se vuelve difícil de mantener, repetitivo y propenso a errores.

React surge como solución a estos problemas. Es una biblioteca que permite dividir tu aplicación en pequeños componentes reutilizables, donde cada uno contiene su propia estructura, estilo y lógica.

Además, ofrece una gestión eficiente del estado: cuando algo cambia (como el número de "me gusta" o el contenido de una lista), React actualiza automáticamente solo lo necesario en pantalla, de forma rápida y ordenada. Esto no solo mejora el rendimiento y la experiencia de usuario, sino que también facilita el mantenimiento, el trabajo colaborativo y la escalabilidad del proyecto.

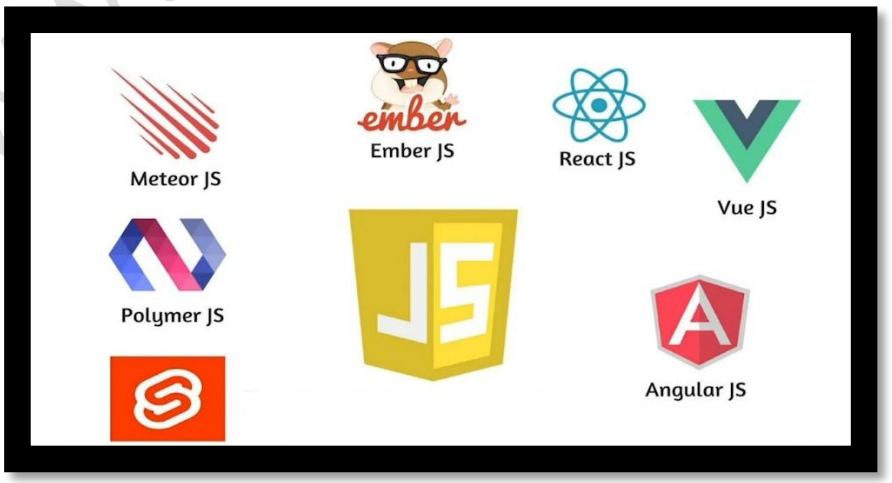
¿Por qué estudiar React?

React es utilizado por empresas como **Meta, Netflix, Instagram, Airbnb o Spotify**, y permite desarrollar desde **páginas web simples** hasta **aplicaciones empresariales complejas** y **apps móviles** con React Native.

Aprender React no solo te da acceso a un ecosistema tecnológico inmenso, sino que te enseña **conceptos clave de la programación moderna**: componentes, estados, props, enrutamiento, hooks… Todo ello mejora tu lógica y tus habilidades como desarrollador web profesional.

Al principio tiene una curva de dificultad alta pero una vez entiendes su sistema por modulo y componentes además de apoyarse en múltiples bibliotecas entenderás porque todas las empresas apuestan por esta herramienta.

FRAMEWORKS



Un *framework* o una *librería* de frontend es una herramienta que facilita el desarrollo de interfaces web, organizando el código y automatizando tareas comunes. Estas tecnologías permiten crear aplicaciones interactivas de forma más eficiente, manteniendo el código limpio, reutilizable y escalable.

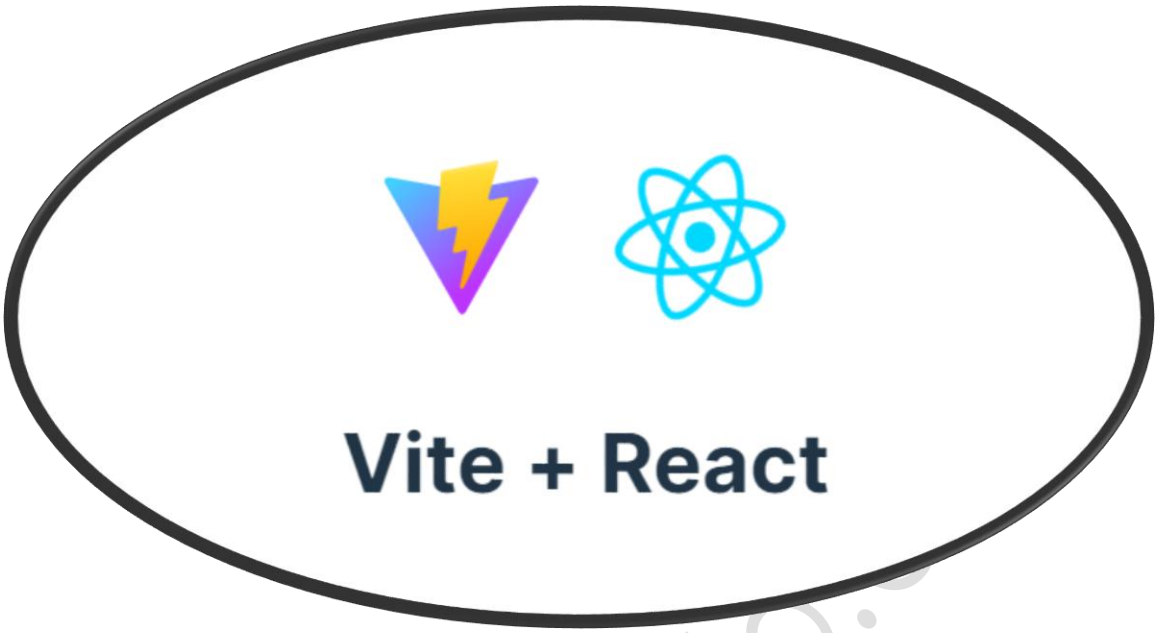
Actualmente, existen cuatro opciones muy populares para crear interfaces de usuario:

- **Angular**: Es un *framework* completo (*todo-en-uno*) mantenido por Google. Suele gustar a quienes vienen del backend, ya que utiliza patrones estructurados y fuertes convenciones. Eso sí, requiere trabajar con TypeScript, lo cual puede suponer una curva de aprendizaje extra.
- **React**: Es una librería enfocada en construir interfaces web mediante componentes. Si ya manejas JavaScript (incluso del lado del servidor, como con Node.js), pero no tienes demasiada experiencia en HTML y CSS, React puede resultarte muy natural. Su filosofía es minimalista y flexible, permitiéndote escoger las herramientas que acompañarán a tu proyecto.
- **VueJS**: Es un *framework progresivo* que busca ser accesible desde el principio, especialmente para quienes dominan HTML y CSS. Tiene una curva de aprendizaje suave y resulta muy cómodo para quienes vienen del diseño web o frontend clásico.
- **Svelte**: Es una librería moderna que combina ideas de React y Vue, pero con un enfoque diferente: en lugar de hacer todo en tiempo de ejecución, compila los componentes en JavaScript optimizado antes de que lleguen al navegador. Es muy eficiente, pero su comunidad aún es más pequeña.

React se ha convertido en la opción más usada por empresas y desarrolladores. Y si se combina con **Vite**, la experiencia mejora aún más. ¿Por qué?

Vite es una herramienta de desarrollo ultrarrápida que permite crear y lanzar proyectos React con una configuración mínima. Gracias a su sistema de recarga instantánea (*hot reload*) y su tiempo de arranque casi inmediato, Vite ha reemplazado con creces a herramientas antiguas como *create-react-app*.

Conclusión:
React + Vite es hoy la combinación preferida para desarrollar interfaces modernas. React aporta la solidez y comunidad de una librería madura, mientras que Vite ofrece una experiencia de desarrollo mucho más ágil y actual. Juntos forman un entorno ideal para aprender, prototipar y construir proyectos profesionales.



BASES DE REACT (CONCEPTOS BASICOS)

Vamos a comentar una lista de conceptos que son esenciales para entender cómo funciona React.

JSX

El lenguaje usado en React es JSX, que a grandes rasgos es la sintaxis de HTML, pero leída e interpretada por JS. (Este coge el texto y lo convierte en HTML).

```
function hola() {
 const name = "TitoGeremito";
 return (
  <div>
   <h1>¡Hola, {name}!</h1>
  </div>
 );
}
```

vDOM

React suele trabajar con lo que llamamos Virtual DOM, un tipo de DOM ligero al que React accede directamente, por lo cual el desarrollador se evita preocuparse por el DOM.

Componentes

React trabaja con componentes y herramientas, lo que busca el desarrollador es tener una biblioteca de componentes los cuales puede reutilizar de forma sencilla facilitando el trabajo.

```
function component() {
  return <h1>¡Hola, Buenos dias!</h1>;
}
```

Ejemplo de **CONTADOR**, pequeño componente con botón que suma +1 a un párrafo.

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

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

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

export default Contador;
```

Luego este contador lo podremos usar en todas las páginas web que queramos solamente añadiendo lo siguiente a nuestro código:

```
<Contador />
```

Propagación Unidireccional

En React los datos solo pueden ir de padres a hijos y no al revés.

- Facilita la gestión de código
- Esto hace que el código sea menos flexible

PRIMER CONTACTO CON REACT

Aunque React es una librería frontend debemos instalar ciertas herramientas para que JavasCript pueda ser correctamente compilado.

- NodeJS (Página oficial de Node.js)
    - node -–version
    - npm --version

- NPM o cualquier gestor de paquetes de NodeJS.
- Automatizador como Vite, podemos usar similares (Al crear el proyecto)

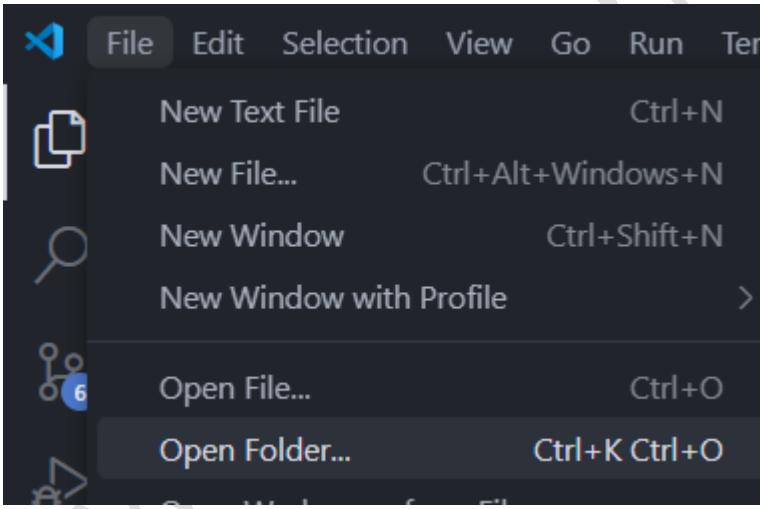
Instalación de React

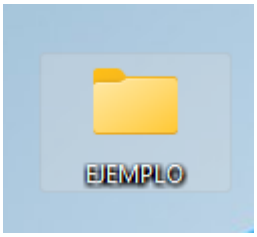
1. Instalamos NodeJS

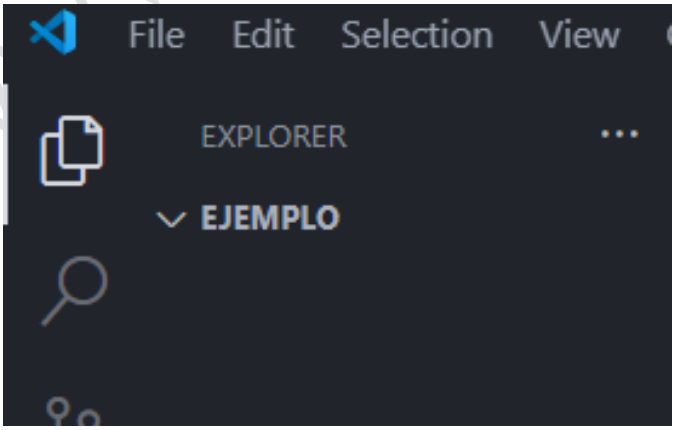
Ante de nada instalamos NodeJS con el enlace de arriba.

2. Abrimos nuestra carpeta y la terminal

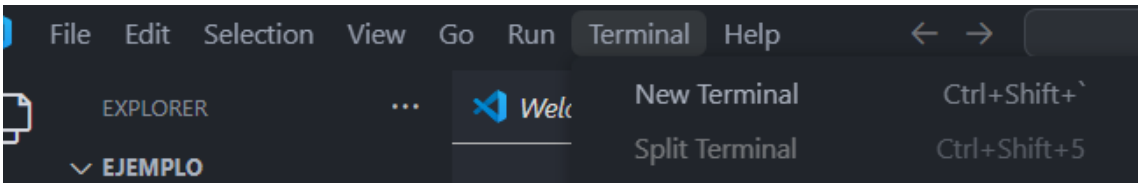
Ahora abrimos en nuestro VSCode la carpeta o el sitio donde vamos a generar nuestro proyecto de React-Vite.

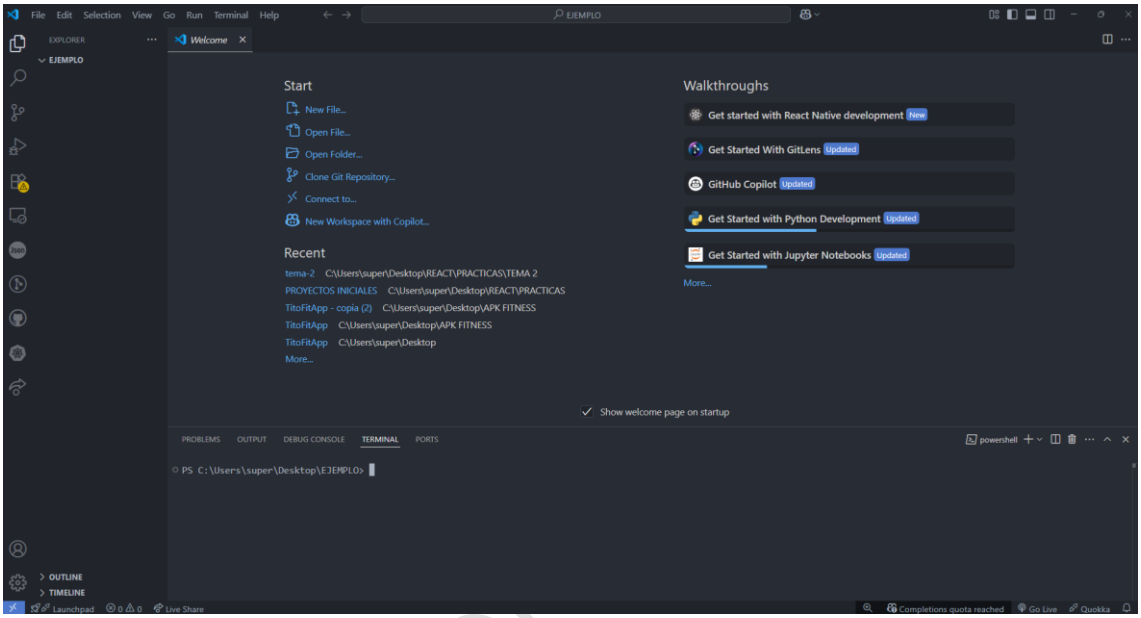






Ya estando dentro de nuestra carpeta abrimos la terminal.

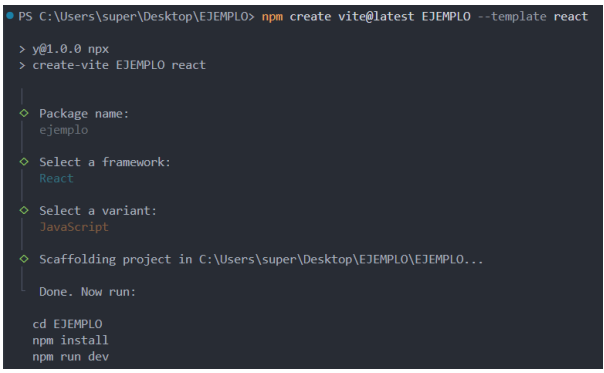




3. Creamos nuestro proyecto con Vite

```
npm create vite@latest
```

Esto generará:



- Crea una carpeta con el nombre del proyecto.
- Instala Vite + React automáticamente.
- Configura todo lo necesario (build, servidor de desarrollo, etc.).
    - **Framework**: React
    - **Variante**: JavaScript

4. Instalamos las dependencias con npm

```
cd nombre-de-tu-proyecto
```

```
npm install
```

```
PS C:\Users\super\Desktop\EJEMPLO> cd ejemplo
PS C:\Users\super\Desktop\EJEMPLO\ejemplo> npm install

added 13 packages, and audited 14 packages in 4s

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

5. Estructura básica que debemos de tener

📁 public
└── vite.svg → Archivos estáticos accesibles desde la app

📁 src
├── App.jsx → Componente principal de tu app
└── style.css → Estilos globales

📄 index.html → HTML base con el
📄 package.json → Configuración del proyecto y dependencias
📄 package-lock.json → Configuración de Vite
📌 React + Vite usa archivos .js desde, pero lo ideal es cambiarlos por archivos. jsx.

6. Finalmente, para lanzar nuestro proyecto

npm run dev

```
PS C:\Users\super\Desktop\EJEMPLO\ejemplo> npm run dev

> ejemplo@0.0.0 dev
> vite


  VITE v7.0.0  ready in 218 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

Clicamos en el enlace y …

PRIMER HOLA MUNDO

Abre el archivo app.jsx.

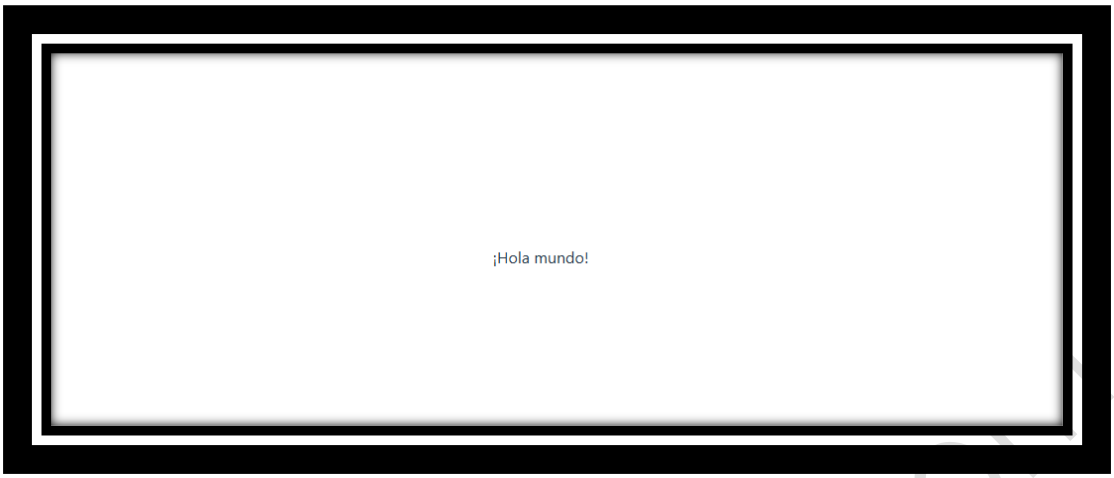
Como puedes ver en el app.jsx ya tenemos un código que es lo que podemos ver al abrir la página de referencia.

Bórralo todo y vamos a escribir nuestro primer '¡Hola mundo!

```
import './App.css'

function App() {
 return (
  <>
   <p>
    ¡Hola mundo!
   </p>
  </>
 )
}
export default App
```

Así quedaría nuestro código y nuestra página web.



Primer componente y sus usos.

Como ya hemos estudiado antes uno de los mayores beneficios de React es construir a través de componentes ya diseñados o que podemos diseñar nosotros.

Vamos a generar nuestro primer componente y lo vamos a añadir a nuestra página.

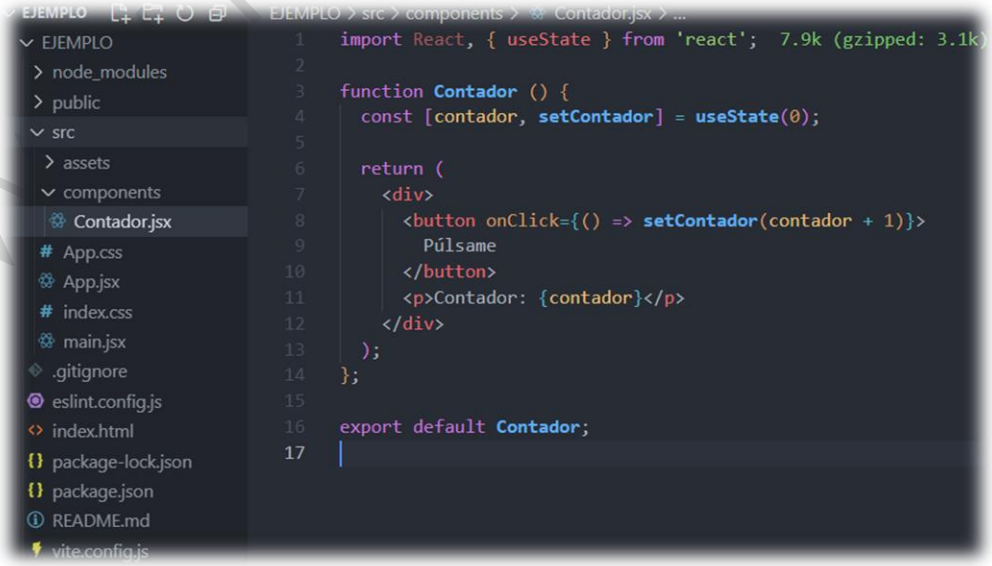
Creación del componente

Lo primero que tenemos que hacer es crear una carpeta de componentes dentro de nuestro src.

/src/components/ (aquí guardamos nuestros componentes.jsx)

Vamos a crear el componente *contador* que he puesto antes más arriba. (enlace contador)

Creamos un nuevo archivo llamado Contador.jsx y copiamos el código en él.



Lo que debéis saber hasta ahora es que lo que creamos es un componente que se basa en una función que nos devuelve en el `return` un JSX el cual cumple unas funcionalidades, usamos el `export default Contador` para que otros archivos puedan importarlo.

Una vez tenemos creado el componente podemos volver a nuestra página inicial `src/app.jsx` y añadirlo.

Añadir componente

Ahora en nuestro `app.jsx` podemos añadir fácilmente cualquier componente generado donde queramos y las veces que queramos.

Primero lo importamos a nuestra página:

```
import Contador from './components/Contador'
```

Luego ya podemos añadirlo como una etiqueta en nuestro app.jsx.

IMPORTANTE

Cada `return` de cada uno de nuestros archivos solo puede tener un contendor.

ESTO BIEN

```
function App() {
 return (
  <div>
   <p>HOLA MUNDO</p>
  </div>
 )
}
```

ESTO CACA

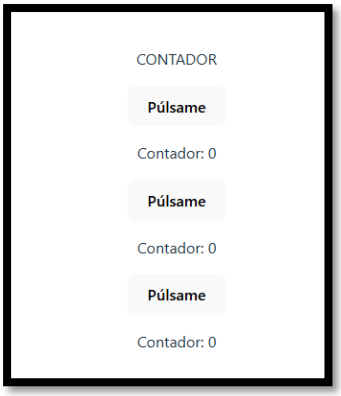
```
function App() {
 return (
  <div>
   <p>HOLA MUNDO</p>
  </div>
  <div>
   <p>HOLA MUNDO</p>
  </div>
 )
}
```

App con el *component*

```
function App() {
 return (
  <div>
   <p>CONTADOR</p>
   <Contador />
  </div>
 )
}
```

Ahora lo divertido es que esto lo podemos añadir las veces que queramos como si fuera una etiqueta más.

```
function App() {
 return (
  <div>
   <p>CONTADOR</p>
   <Contador />
   <Contador />
   <Contador />
  </div>
 )
}
```



Esto tiene infinidad de usos y de aplicaciones.

- **Contadores personalizados**
  Para contar clics, visitas, ítems añadidos al carrito o repeticiones de ejercicios.
- **Mensajes de bienvenida personalizados**
  Mostrar saludos o avisos distintos según el nombre o perfil del usuario.
- **Tarjetas de información**
  Mostrar datos de usuarios, productos o artículos en un formato visual atractivo y reutilizable.

- **Listas dinámicas**
  Generar listas automáticas de tareas, mensajes, productos o actividades desde un array de datos.
- **Muchos otros**

Intenta hacer tu propio componente desde cero.

Tu misión es crear un componente que salude utilizando una variable personalizada con tu nombre.

CONCLUSION

React es una de las herramientas más potentes y versátiles para crear interfaces web modernas. Aprender sus fundamentos te abre las puertas a múltiples oportunidades en el mundo del desarrollo: desde simples páginas web hasta aplicaciones empresariales y móviles.

Dominar React no solo implica aprender su sintaxis, sino entender su filosofía: **dividir para conquistar**, **reutilizar para optimizar** y **gestionar el estado para escalar**.

Este primer acercamiento es solo el comienzo. A medida que avances, descubrirás hooks como useEffect, sistemas de rutas, contextos globales y frameworks que extienden aún más las capacidades de React.

BIBLIOTECAS PARA REACT

Te dejo unas cuantas bibliotecas de componentes de React para que pruebes a hacer tus propios proyectos, como siempre digo, prueba, toca y rompe que es la única manera de aprender

Mucho ánimo con tu estudio.

- https://react-bootstrap.netlify.app/
- https://mui.com/
- https://tailwindcomponents.com/
- https://react-hook-form.com/
- https://headlessui.com/
- https://www.radix-ui.com/
- https://github.com/brillout/awesome-react-components
- https://react-icons.github.io/react-icons/
- https://mantine.dev/
- https://storybook.js.org/

¿Te ha gustado esta guía?

Entonces te encantará todo lo que te espera en www.titogeremito.com

- Cursos prácticos de React y desarrollo de apps
- Ebooks descargables sobre programación y productividad
- Mi blog personal con estrategias, motivación y tecnología
- Visita ya: www.titogeremito.com/react