**Ruta de investigación React JS**

**Autor:** David Vaamonde Bueno

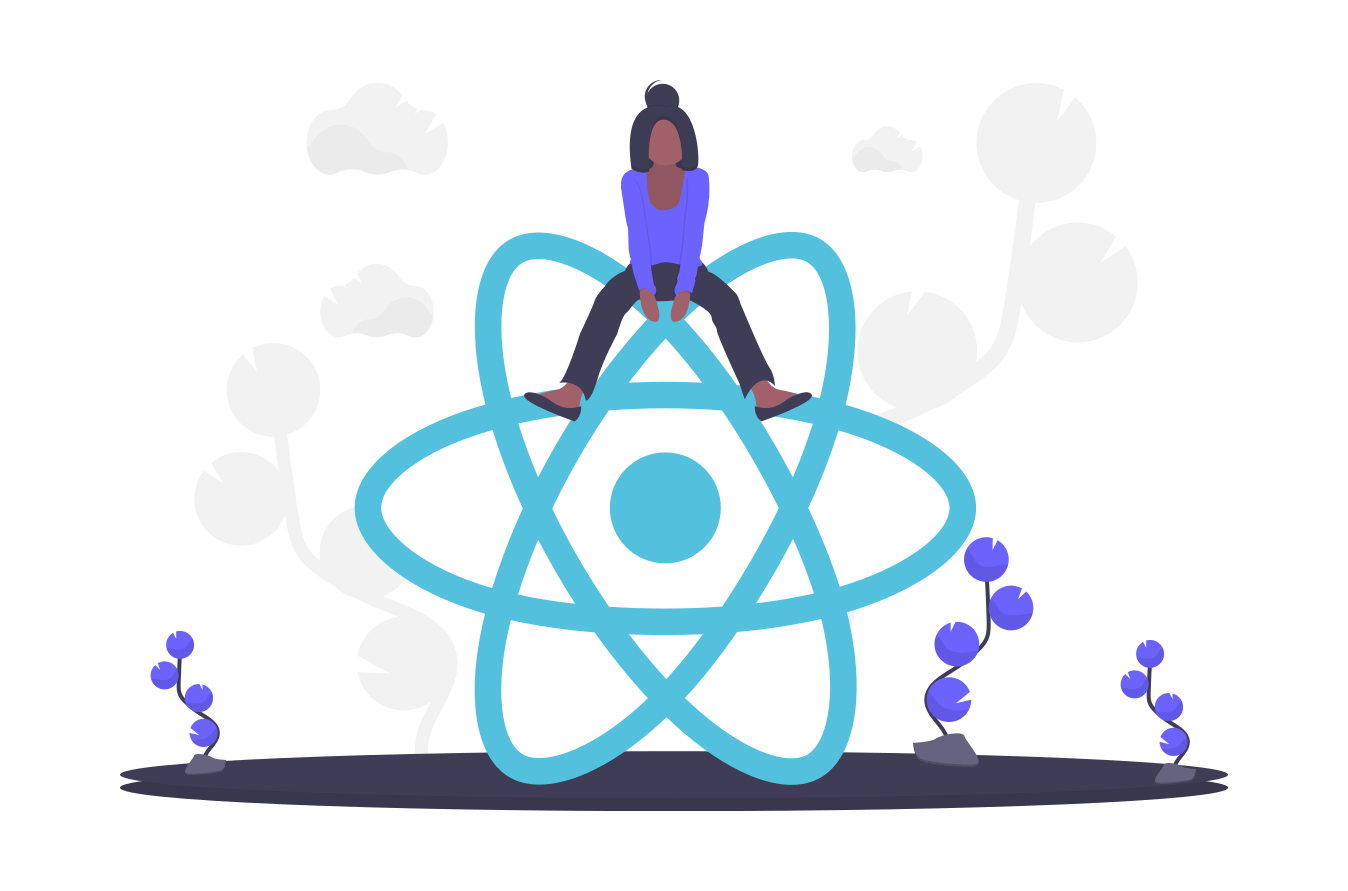
1. **✅ ¿Qué es React y para qué se utiliza?**

React (o también llamado **React.js** o ReactJS) es una biblioteca **JavaScript** de código abierto diseñada para crear **interfaces de usuario,** cuyo objetivo es facilitar el desarrollo de aplicaciones **en una sola página**. Es mantenido por [Facebook](https://es.wikipedia.org/wiki/Facebook) y la comunidad de **software libre**. En el proyecto hay más de mil desarrolladores libres.

React combina HTML con la funcionalidad de JavaScript para crear su propio lenguaje de marcado llamado **JSX**. Además, facilita la administración del flujo de datos a través de la aplicación.

React fue creado por [**Jordan Walke**](https://www.linkedin.com/in/jordwalke/), un ingeniero de software de Facebook, quien liberó un primer prototipo de React llamado "FaxJS".

Este fue influenciado por XHP de HTML una librería de componentes para PHP. Este fue usado por primera vez en el Feed de Noticias de Facebook en 2011 y después en Instagram en 2012.

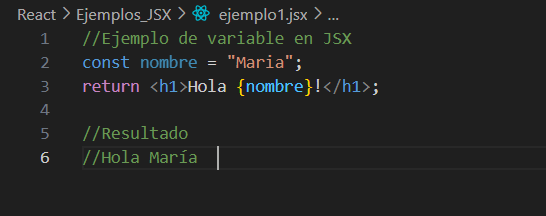


1. **🔤 ¿Qué es JSX y cómo se diferencia del HTML?**

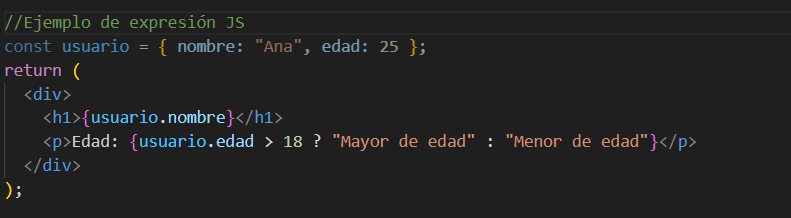
**JSX** (JavaScript XML) es una extensión de sintaxis de JavaScript que permite escribir código que parece HTML dentro de JavaScript. Es como una fusión entre HTML Y JavaScript que hace más fácil describir cómo debería verse la interfaz de usuario en React.

**Principales diferencias con HTML:**

* **Es JavaScript, no HTML**
  + JSX se transpira a JavaScript puro antes de ejecutarse.
  + Puedes usar variables, funciones y expresiones JavaScript dentro de las llaves **{}**

****

* **Los atributos son diferentes**
  + En React se utiliza **className** en lugar de **class** de HTML, porque **class** es una palabra reservada en **JavaScript**.
  + En React se utiliza **htmlFor** en lugar de **for**.
  + Los atributos usan camelCase: **onClick** en lugar de **onclick**.
* **Expresiones JavaScript embebidas**

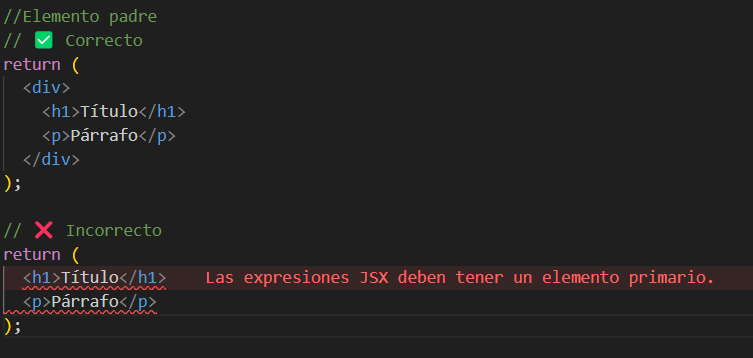
****

* **Componentes personalizados**: En JSX puedes usar componentes React como si fueran etiquetas HTML:

<MiComponente />

<Button color=”blue” onClick={handleClick} />

* **Debe tener un elemento padre:** Ejemplo:

****

1. **🧠 ¿Qué es Virtual DOM y porqué es importante en React?**

El Virtual DOM es una representación en memoria (como una copia ligera) del DOM real de la página web. Es básicamente un objeto JavaScript que describe cómo debería verse la interfaz de usuario, pero no es el DOM que el navegador realmente renderiza.

**¿Por qué es tan importante?**

1. **El DOM real es lento,** manipular el DOM directamente es costoso computacionalmente. Cada vez que cambias algo en el DOM, el navegador tiene que **recalcular estilos**, reorganizar elementos (**reflow**) y repintar la página (**repaint**).
2. **React usa un proceso inteligente**:

Estado cambia 🡪 Se crea nuevo Virtual DOM 🡪 Se compara con el anterior 🡪 Sólo se actualizan las diferencias en el DOM actual

**¿Qué hace el Virtual DOM?**

El Virtual DOM realiza varias tareas específicas que son clave para el funcionamiento de React:

1. **Actúa como una representación en memoria**
2. **Crea una “foto” del estado actual**: Cada vez que algo cambia en tu aplicación, React crea un nuevo Virtual DOM completo que representa cómo debería verse la interfaz en ese momento.
3. **Compara (Diffing)**: Esta es su tarea más importante:

* Toma el Virtual DOM anterior
* Toma el Virtual DOM nuevo
* Los compara elemento por elemento
* Identifica exactamente qué cambió

**Ejemplo práctico:**

// Estado inicial

<div>

<h1>Contador: 0</h1>

<button>Incrementar</button>

</div>

// Después de hacer click (nuevo estado)

<div>

<h1>Contador: 1</h1> ← Solo esto cambió

<button>Incrementar</button> ← Esto sigue igual

</div>

1. **Calcula los cambios mínimos necesarios:** El Virtual DOM determina que sólo necesita:

* Cambiar el texto del <h1> de “Contador: 0” a “Contador: 1”.
* NO tocar el <button> porque no cambió.

1. **Genera instrucciones de actualización:** Le dice a ReactDOM: “Solo actualiza el contenido de texto de este elemento específico”.
2. **Optimiza las actualizaciones:**

* Agrupa múltiples cambios en una sola actualización.
* Evita actualizaciones innecesarias.
* Minimiza las operaciones costosas en el DOM real.

1. **🚀 ¿Cómo puedo crear un proyecto nuevo en React utilizando Vite?**

**Vite** es una herramienta que ofrece un servidor de desarrollo muy rápido y un empaquetador optimizado. Es mucho más rápido que las herramientas tradicionales como Webpack.

**Pasos a seguir para crear un proyecto nuevo React con Vite**

1. Asegúrate de tener instalado en el ordenador el programa **Node.js**. Para comprobar si se ha instalado, abre el terminal y ejecuta el siguiente comando 🡪 “node -v”. Recomendable tener la versión v14.18 o superior.
2. Crear el proyecto. Abre el terminal y ejecuta:

“npm create vite@latest mi-proyecto-react -- --template react”

Donde cada parte del comando significa:

* **npm**: Es el gestor de paquetes de Node.js. Es la herramienta que usamos para instalar y gestionar dependencias.
* **create**: Descarga y ejecuta temporalmente el paquete especificado. Es equivalente a “npx créate-<paquete>”.
* **vite@latest:** “vite” es el nombre del paquete/herramienta que queremos usar, y “@latest” especifica que queremos la versión más reciente de Vite. Se puede usar versiones específicas como “**vite@4.5.0”**.
* **mi-proyecto-react**: Es el nombre del directorio/proyecto que se va a crear. Puedes cambiarlo por el nombre que quieras. Ejemplo: mi-app, tienda-online, restaurante-cachopo…
* **--**: Es un **separador de argumentos**. Todo lo que viene después se pasa directamente a Vite. Le dice a npm: “estos argumentos son para Vite, no para npm”.
* **–template react**: “--template” es una opción/flag de Vite que especifica qué plantilla usar, y “react” es el tipo de plantilla que queremos (proyecto React con Javascript).

También puedes utilizar:

#Con npm

“npm create vite@latest ‘ ‘ ‘ ‘ ”

# Con yarn

“yarn create vite ‘ ‘ ‘ ‘ “

# Con pnpm

“pnpm create vite ‘ ‘ ‘ ‘ “

1. Navegar al directorio del proyecto:

“cd mi-proyecto-react”

1. Instalar las dependencias:

“npm install”

1. Iniciar el servidor de desarrollo:

“npm run dev”

1. ¡Listo! Tu aplicación estará arrancando en [**http://localhost:5173**](http://localhost:5173).

**Estructura del proyecto creado:**

****

**Comandos que te serán útiles:**

“npm run dev” 🡪 Iniciar servidor de desarrollo

“npm run build” 🡪 Construir para producción

“npm run preview” 🡪 Previsualizar build de producción

**Ventajas de Vite sobre Create React App:**

* Mucho más rápido en el desarrollo
* Hot Module Replacement instantáneo
* Menos configuración necesaria
* Mejor optimización para producción
* Soporte nativo para TypeScript, CSS modules, etc.

1. **🧩 ¿Qué son los componentes en React y cómo creo uno básico?**
2. **🧱 ¿Cómo puedo descomponer la interfaz en varios componentes reutilizables?**
3. **📩 ¿Qué son los “*props*” y cómo se usan para pasar datos a los componentes?**
4. **🛠️ ¿Por qué necesitamos Vite o Create React App para empezar?**
5. **🖥️ ¿Cómo renderiza React un componente en el DOM de una página web?**
6. **⚙️ ¿Qué es “*ReactDOM.render()*” y cómo funciona en la inicialización de una aplicación React?**
7. **📦 ¿Por qué necesitamos un contenedor raíz (*root element*)?**