Taller Sobre Componentes Frontend

Evidencia: GA7-220501096-AA4-EV01

Formación: Tecnología en Análisis y Desarrollo de Software

Aprendiz: Jefferson Harbey Mendez Castellanos

Instructor: John Alejando Niño Tambo

Ficha: 2977395

Fecha: 27-10-2025

Contenido

Introducción

El siguiente documento tiene como objetivo central desarrollar un taller que incluya las bases principales y fundamentos de React. Al final del taller se espera desarrollar un conocimiento profundo en cada uno de los conceptos claves de React, que nos permitan entender a profundidad que es y cómo funciona esta importante tecnología, la cual nos permite crear interfaces modernas, mediante componentes reutilizables creados con HTML, CSS JavaScript y JSX.

Objetivo

Realizar una investigación profunda sobre lo que es React, sus principales fundamentos y ejemplos de cómo funciona, definir las diferencias entre React y JSX, incluir las diferencias entre componentes funcionales y componentes de clases entre otros conceptos claves de React, y finalmente crear un mapa conceptual de React.

¿Qué es React?

React es una biblioteca de JavaScript de código abierto para construir interfaces de usuario (UI), no es un framework completo sino una librería que se puede combinar con otras herramientas de desarrollo. Fue creada por Facebook en el año 2013, y se basa en componentes reutilizables para construir interfaces complejas de manera eficiente. Su principal característica es el uso del Virtual DOM, que mejora el rendimiento al actualizar solo las partes necesarias de la interfaz

¿Como funciona React?

Todo en React es un componente: es decir, piezas independientes y reutilizables de código, cada componente es desarrollado con su propia lógica promoviendo principios como el (no te repitas), de esta manera React se encarga de actualizar y renderizar los componentes creando interfaces muy poderosas y modernas.

Ventajas de utilizar React

- Rendimiento optimizado: React nos da un excelente rendimiento gracias a el Virtual DOM, esto debido a que se mantiene una representación virtual DOM en memoria, así cuando hay cambios React solo actualiza las partes específicas que se cambiaron y no todo el árbol DOM.
- Reutilización de código: cada componente se puede reutilizar en múltiples lugares de la aplicación.
- Mantenibilidad: y es que en React organizamos el código en componentes aislados, lo cual lo hace más fácil de probar, gracias a su estructura modular, clara y predecible.

Diferencias entre React Y Jsx

La tabla a continuación presenta las principales diferencias entre Reat y JSX

Aspecto	React	JSX
¿Que son?	Biblioteca de JavaScript completa	Extensión de sintaxis de JavaScript
Naturaleza	Herramienta/Framework	Lenguaje de marcado dentro de JS
Función Principal	Construir interfaces de usuario y gestionar componentes	Escribir estructura HTML dentro de JS
Uso	Se importa y utiliza para crear componentes (import React from "react")	Se usa dentro de React para definir cómo se ve la interfaz (return Hola)
Dependencia	Puede funcionar sin JSX (usando React.createElement()	Depende de React para funcionar
Obligatoriedad	Obligatorio para usar React	Opcional pero altamente recomendado
compilación	Se ejecuta directamente en el navegador	Se transpila a React.createElement() mediante babel antes de ejecutarse
Sintaxis	JavaScript estándar con funciones y métodos	HTML-like dentro de código JavaScript
Ejemplo	React.createElement('div', null, 'hola')	Hola

Ejemplo

Tarjeta de producto

Sin JSX “React puro”

```
function TarjetaProducto() {
  return React.createElement(
    'div',
    { className: 'producto-card' },
    React.createElement('img', {
      src: 'laptop.jpg',
      alt: 'Laptop'
    }),
    React.createElement('h2', null, 'Laptop HP'),
    React.createElement('p', { className: 'precio' },
'$1500000'),
    React.createElement(
      'button',
      { onClick: () => console.log('Añadido al carrito') },
      'Añadir al carrito'
    )
  );
}
```

Ahora con JSX

```
function TarjetaProducto() {
  return (
    <div className="producto-card">
      <img src="laptop.jpg" alt="Laptop" />
      <h2>Laptop HP</h2>
      <p className="precio">$1.500.000</p>
```

```
      <button onClick={() => console.log('Añadido al
carrito')}>
        Añadir al carrito
      </button>
    </div>
  );
}
```

Se puede analizar que la tarjeta que creamos con jsx es más concisa y un poco más fácil de leer a simple vista que la tarjeta creada con React puro.

Componente Funcional vs Componente de Clase

Aspecto	Componente Funcional	Componente de Clase
Definición	Función de JS que retorna JSX	Clase de ES6 que extiende de React.Component
Sintaxis	Mas simple y concisa	Mas verbosa y compleja
Estado	Usa Hooks (useState)	Usa this.state
Ciclo de vida	Usa useEffect	Usa métodos como componentDidMount, componentDidUpdate
Uso de this	No usa this	Requiere this para acceder a props y estado
Hooks	Puede usar Hooks	No puede usar hooks

Ejemplo componente funcional

Un componente funcional es simplemente una función de JavaScript recibe props como argumento y retorna un JSX, usa hooks para manejar estado y efectos secundarios.

```
function Contador() {
  return (
    <div>
      <h2>Contador Simple</h2>
      <p>Valor inicial: 0</p>
    </div>
  );
}
```

Ejemplo componente de clase

Es una clase ES6 que extiende de React.Component, requiere de un método render() el cual retorna el JSX, usa this.state para el estado y this.props para las props. Requiere un constructor para inicializar el estado y necesita .bind(this) o funciones flecha para métodos.

```
import React, { Component } from "react";

class Contador extends Component {
    render () {
        return (
            <div>
                <h1>Contador simple</h1>
                <p>valor inicial: 0</p>
            </div>
        )
```

```
    }
}

export default Contador;
```

Elementos vs Componentes en React

A continuación, se presenta una tabla comparativa con el objetivo de aclarar las diferencias entre elementos y componentes en React.

Aspecto	Elementos	Componentes
¿Qué son?	Objetos simples que describen lo que uno quiere ver en pantalla	Funciones o clases que precisamente retornan esos elementos
Naturaleza	Datos(objetos JS)	Codigo reutilizable (funciones/clases)
Mutabilidad	Inmutables	Pueden tener estado y lógica
Creación	Se crean con JSX o React.createElement()	Se definen como funciones o clases
Instancias	Representan un momento especifico en el tiempo	Pueden renderizarse múltiples veces
Ejemplo	Hola	function Saludo() { return Hola }

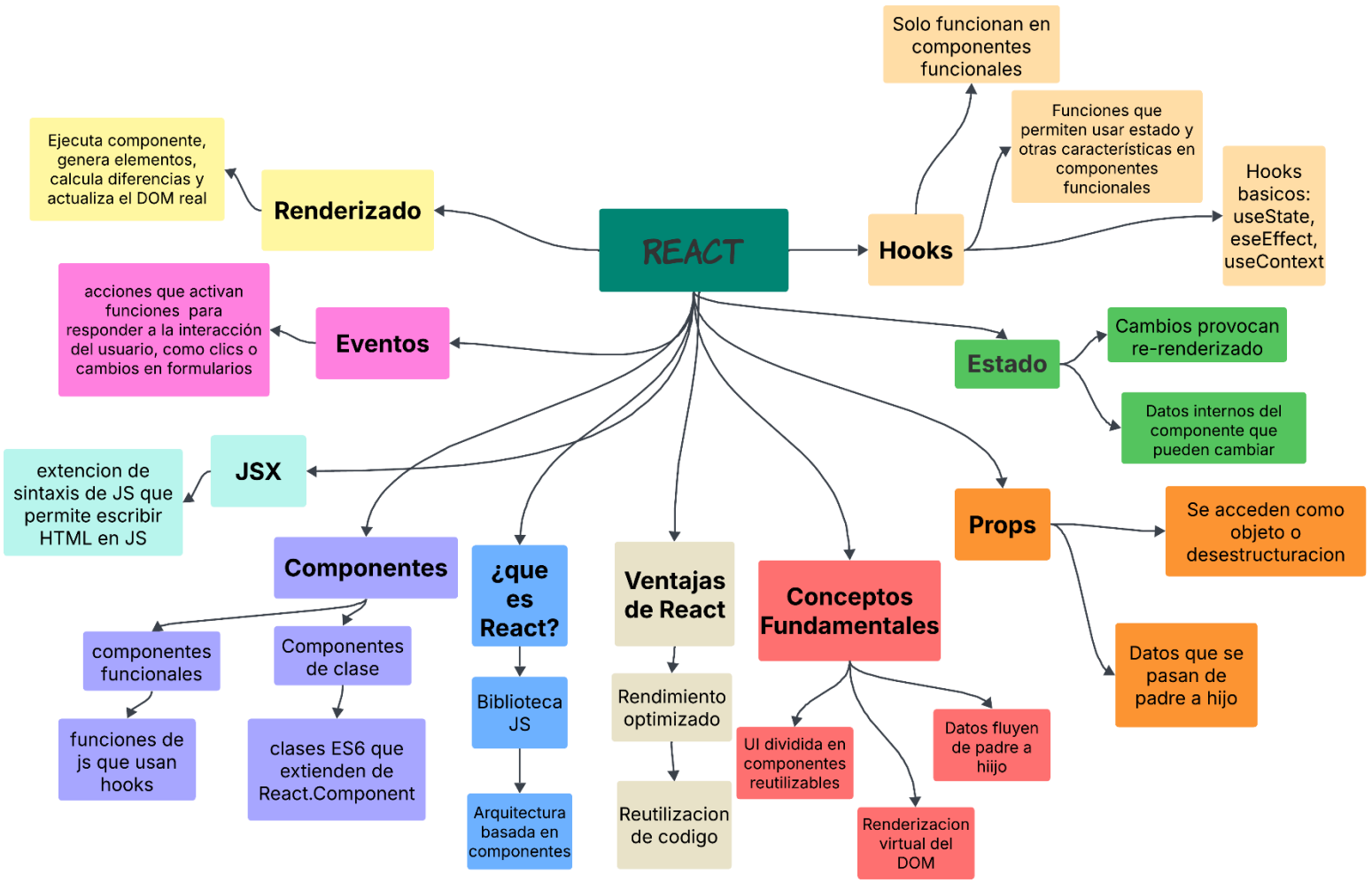
En conclusión, un elemento es un objeto plano que describe lo que el usuario quiere ver en pantalla, describen un nodo del DOM y es la unidad más pequeña en React. Por su parte un componente es una función o clase que opcionalmente acepta entradas (props) y retorna un elemento de React que describe como debe verse una sección de la interfaz.

Ejemplo

```
//elemento (no reutilizable)
Const miElemento = <h1>Hola</h1>;

//componente(reutilizable)
Function MiComponente() {
Return <h1>Hola</h1>;
}
```

Mapa coneptual sobre React



Conclusiones

Se ha desarrollado un taller sobre React, se buscó profundizar en cada uno de los conceptos más importantes, como que es React y para que se utiliza, porque su importancia para el desarrollo de componentes Frontend, su funcionamiento mediante conceptos claves como: componentes, elementos, props, estado, eventos, renderizado, JSX, Hooks, entre muchos conceptos más, esto con el fin de profundizar y obtener un conocimiento solido en esta importante tecnología. Después de realizar la investigación y desarrollar el taller, podemos concluir que React es una importante y poderosa biblioteca que está presente en muchas de las interfaces de muchos sitios webs y plataformas de las que utilizamos día a día, es una tecnología clave en el mundo del desarrollo moderno y por lo tanto estudiarla y empezar a implantarla en proyectos es clave, no solo para aumentar nuestro conocimiento técnico y habilidades, sino también para optimizar y aumentar nuestro rendimiento como desarrolladores.