Servicio Nacional de Aprendizaje – SENA

Evidencia de Desempeño: GA7-220501096-AA4-EV01 taller sobre componentes frontend

Taller sobre componentes frontend

Análisis y desarrollo de software

Ana del pilar Bermeo Lozano

Fecha: 05 de Julio de 2025

Introducción

En este informe se exploran conceptos fundamentales sobre **React.js**, una biblioteca JavaScript de código abierto diseñada para desarrollar interfaces de usuario dinámicas y escalables, junto con su extensión sintáctica **JSX**. React y JSX permiten la creación de componentes reutilizables que facilitan la construcción y mantenimiento de aplicaciones web modernas.

Este taller se centra en:

- Analizar las **diferencias entre React y JSX**,
- Comparar **componentes en React**, tanto de clase como funcionales,
- Describir los **eventos principales utilizados en el desarrollo frontend con React**.

El propósito es consolidar los conocimientos teóricos mediante ejemplos y actividades prácticas, permitiendo al lector comprender cómo y cuándo usar cada enfoque en proyectos reales de desarrollo front-end.

Objetivo

El taller busca comprender de forma integral el uso de React y JSX, enfocándose en las diferencias entre ambos, el análisis de componentes de clase y funcionales, el manejo de eventos comunes en React y la visualización de estos conceptos mediante un mapa conceptual.

1.Diferencia entre React y JSX

React es una biblioteca frontend de JavaScript, de código abierto, diseñada específicamente para construir interfaces de usuario interactivas, reutilizables y fáciles de mantener. Su enfoque basado en componentes permite dividir la interfaz en piezas independientes que pueden reutilizarse en distintas partes de una aplicación, facilitando así el desarrollo escalable y organizado.

Por otro lado, JSX (JavaScript XML) es una extensión de la sintaxis de JavaScript que permite escribir estructuras similares a HTML dentro del mismo archivo JS. Fue desarrollado como complemento para React, con el propósito de hacer que el código de los componentes sea más claro, visual y declarativo.

Mientras que React proporciona la lógica y estructura para construir interfaces, JSX se encarga de representar visualmente la estructura del componente, haciendo que el código sea más intuitivo para los desarrolladores, sobre todo cuando se manejan jerarquías complejas de elementos.

Una de las grandes ventajas del uso combinado de React y JSX es la capacidad de reutilizar componentes con diferentes propiedades, evitando la repetición de código. Por ejemplo, puedes crear un único componente de botón en React y usarlo múltiples veces en tu sitio web, simplemente cambiando sus propiedades (props), como el color o la acción al hacer clic, en lugar de escribir el mismo bloque de código repetidamente.

En conclusión, React y JSX forman un dúo poderoso en el desarrollo web moderno. React aporta la lógica y el manejo de estados, mientras que JSX mejora la legibilidad y la forma en que estructuramos visualmente los componentes. Juntos permiten construir interfaces ricas, dinámicas y mantenibles con una experiencia de desarrollo mucho más fluida y eficiente.

2. ¿Qué son clases en React?

En React, una clase es una forma de definir un componente utilizando la sintaxis de clases de JavaScript. Estas clases extienden la clase base React.Component, lo que les permite acceder a funcionalidades avanzadas como el manejo de estado interno (state) y los métodos del ciclo de vida del componente.

Un componente de clase puede controlar qué sucede antes, durante y después de su renderizado, gracias a métodos como componentDidMount (cuando se monta) o componentWillUnmount (cuando se elimina del DOM).

```
class Biology extends React.Component {
  render() {
    return <h1>Hola desde el componente Biology</h1>;
  }
}
```

En este ejemplo, Biology es un componente que hereda todo el comportamiento necesario desde React.Component.

3. ¿Cuáles son los principales eventos en React?

Los eventos en React funcionan de forma similar a los del DOM tradicional, pero están normalizados a través de SyntheticEvent, una envoltura compatible entre navegadores.

Algunos de los eventos más comunes son:

EventoDescripción

onClick Se activa al hacer clic en un elemento

onChange Se activa cuando cambia el valor de un input

onSubmit Se activa al enviar un formulario

onMouseOver Se activa al pasar el mouse sobre un elemento

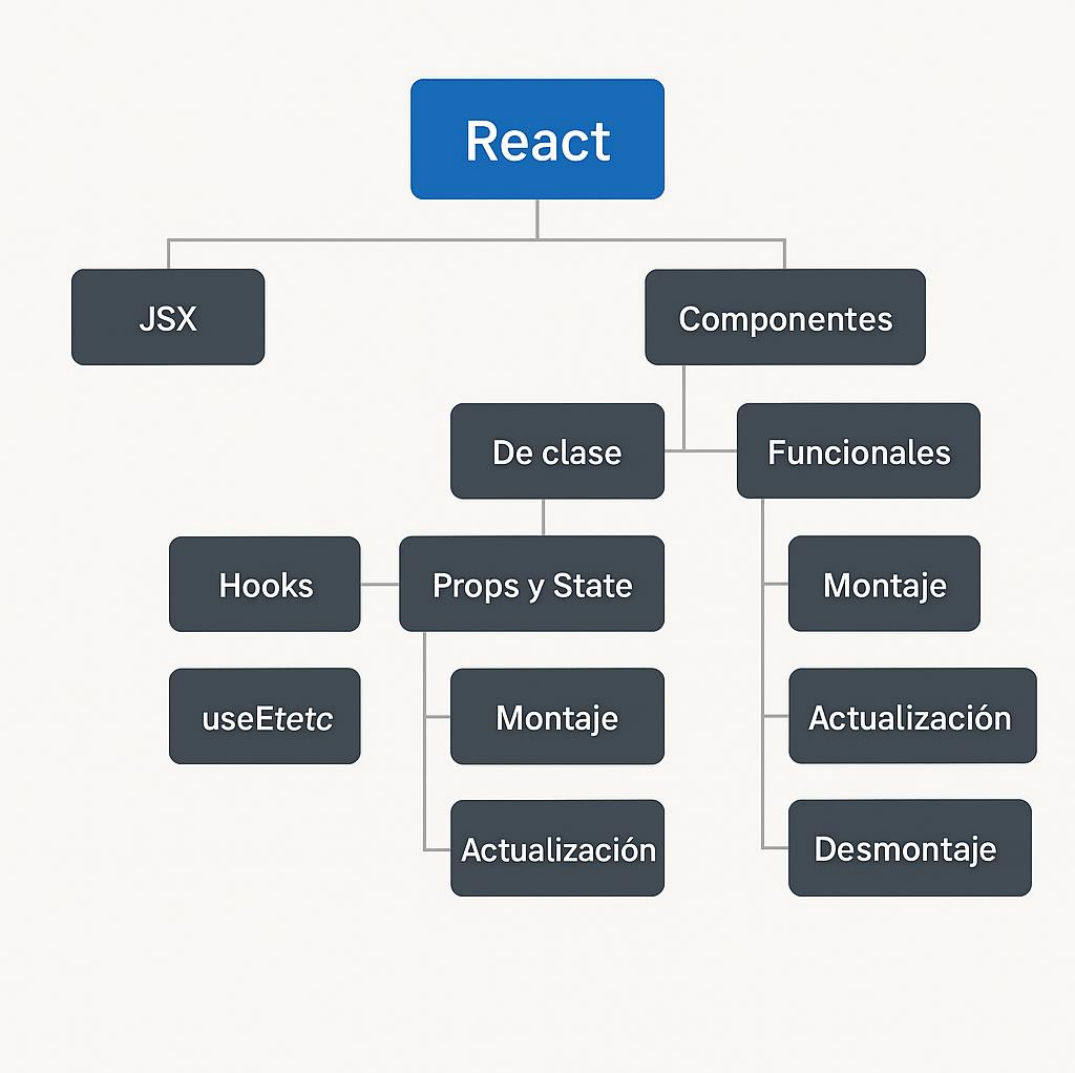
onKeyDown Se activa al presionar una tecla

Ejemplo:

```
<button onClick={handleClick}>Haz clic</button>
```

Estos eventos permiten manejar interacciones del usuario de forma declarativa y efectiva dentro de los componentes.

4. Mapa conceptual de React



Mapa Conceptual de React – Explicación

Nodo principal: React

Es el núcleo del mapa. React es una biblioteca de JavaScript enfocada en construir interfaces de usuario basadas en componentes.

- **JSX**

Este nodo está directamente conectado a React. JSX es una extensión de la sintaxis de JavaScript que permite escribir código similar a HTML dentro de componentes React. Mejora la legibilidad del código y facilita estructurar las interfaces.

- **Componentes**

React se basa en componentes. Estos se dividen en dos tipos:

▪ De clase

- Se crean con class y extienden React.Component.
- Usan this.state y métodos de ciclo de vida como:
  - **Montaje**: componentDidMount
  - **Actualización**: componentDidUpdate
  - **Desmontaje**: componentWillUnmount
- Se apoyan en **Props y State** para manejar datos.

▪ Funcionales

- Se definen como funciones y retornan JSX.
- Para manejar lógica compleja usan **Hooks** como:
  - useState, useEffect, useContext, etc. (en el mapa está como "useEtetc", posiblemente por un error tipográfico).
- También tienen su propio flujo de:
  - **Montaje**
  - **Actualización**
  - **Desmontaje**

🔹 Hooks

Los Hooks son funciones especiales que permiten a los componentes funcionales tener estado y lógica de ciclo de vida, sin usar clases.

🔹 Props y State

Aunque están asociados en el mapa con componentes de clase, en realidad **Props** (propiedades) son externas y se reciben desde el componente padre, mientras que **State** (estado) es interno del componente. Ambos permiten que los componentes sean dinámicos y reactivos.