Taller sobre componentes frontend. GA7-220501096-AA4-EV01

Aprendiz:

Valentina Vargas Sanchez

Instructor

Yerman Augusto Hernández Sáenz

CENTRO DE SERVICIOS FINANCIEROS

SENA- REGIONAL DISTRITO CAPITAL

ANALISIS Y DESARROLLO DEL SOFTWARE

FICHA: 2627062

2024

Diferencia entre React y JSX

React y JSX son dos conceptos que están estrechamente relacionados en el desarrollo de aplicaciones web con React.js, una biblioteca de JavaScript popular para la creación de interfaces de usuario. Aquí hay una explicación de cada uno y la diferencia entre ellos:

- React:

✓ React es una biblioteca de JavaScript desarrollada por Facebook para construir interfaces de usuario interactivas y reutilizables.
✓ React permite a los desarrolladores dividir la interfaz de usuario en componentes independientes y reutilizables, lo que facilita la construcción y el mantenimiento de aplicaciones web complejas.
✓ Utiliza un enfoque basado en componentes para el desarrollo de aplicaciones web, donde cada componente encapsula una parte específica de la interfaz de usuario y maneja su propio estado.

- JSX:

✓ JSX es una extensión de JavaScript que permite escribir código HTML dentro de JavaScript.
✓ JSX es una sintaxis de JavaScript que facilita la definición de la estructura del DOM en React de una manera más intuitiva y declarativa.
✓ Permite a los desarrolladores escribir código que combina la lógica de JavaScript con la estructura de los elementos de la interfaz de usuario, lo que hace que el código sea más legible y fácil de entender.

Diferencia:

La diferencia principal entre React y JSX radica en su propósito y naturaleza:

- React es la biblioteca en sí misma, proporcionando herramientas y funcionalidades para el desarrollo de aplicaciones web.
- JSX, por otro lado, es una sintaxis de JavaScript utilizada dentro de React para definir la estructura de los elementos de la interfaz de usuario. No es necesario utilizar JSX con React, pero es una práctica común debido a su conveniencia y legibilidad.

¿Qué son clases en React?

En React, las clases son un tipo de componente que se define utilizando la sintaxis de clases de JavaScript. Antes de la introducción de los Hooks en React, las clases eran la principal forma de definir componentes con estado y ciclo de vida en React.

Aquí hay algunas características y conceptos clave relacionados con las clases en React:

- ***Extensión de la clase Component:*** En React, las clases se extienden de la clase base Component proporcionada por la biblioteca React. Al extender la clase Component, se heredan funcionalidades y métodos que son esenciales para el funcionamiento de los componentes en React.

- ***Estado y Ciclo de vida:*** Las clases en React pueden contener un estado interno y pueden implementar los métodos del ciclo de vida de React, como componentDidMount, componentDidUpdate, y componentWillUnmount. Esto permite que los componentes manejen su estado interno y realicen acciones específicas en diferentes etapas del ciclo de vida de un componente.

- ***Método render():*** Todas las clases de React deben implementar un método render() que devuelve el JSX que representa el contenido del componente. Este método es responsable de describir cómo debería renderizarse el componente en la interfaz de usuario.

Principales eventos de React

En React, los eventos son acciones que ocurren en los elementos de la interfaz de usuario, como hacer clic en un botón, mover el ratón sobre un elemento, escribir en un campo de entrada, etc. Estos eventos son manejados por funciones conocidas como "manejadores de eventos". Aquí están algunos de los eventos más comunes en React:

1. ***onClick:*** Este evento se desencadena cuando se hace clic en un elemento.

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

2. ***onChange:*** Se activa cuando el valor de un elemento de entrada cambia, como un campo de texto o un select.

   <input type="text" onChange={handleChange} />

3. ***onMouseOver y onMouseOut:*** Estos eventos ocurren cuando el mouse se mueve sobre un elemento o sale de él, respectivamente.

    <div onMouseOver={handleMouseOver}
    *onMouseOut={handleMouseOut}>Pasa el mouse sobre mí *

4. ***onSubmit:*** Se activa cuando se envía un formulario.

    <form onSubmit={handleSubmit}>

    {/ Contenido del formulario */}*

    </form>

5. ***onFocus y onBlur:*** Se activan cuando un elemento obtiene o pierde el foco, respectivamente.

    <input type="text" onFocus={handleFocus} onBlur={handleBlur} />

6. ***onKeyPress y onKeyDown:*** Estos eventos ocurren cuando una tecla se presiona o se mantiene presionada, respectivamente.

    <input type="text" onKeyPress={handleKeyPress}
    onKeyDown={handleKeyDown} />

7. ***onLoad:*** Se activa cuando un elemento, como una imagen, se carga completamente.

    **

8. ***onScroll:*** Se desencadena cuando se desplaza un elemento, como una ventana o un contenedor.

    * Contenido que se puede desplazar *

Mapa conceptual