**EVIDENCIA DE CONOCIMIENTO**

GA7-220501096-AA4-EV01

TALLER SOBRE COMPONENTES FRONTEND

Aprendices

JUAN JOSE REYES CUERO

ALISON RIVERA

Instructora

TATIANA ALEXANDRA FORERO GRANADA

TECNOLOGÍA EN ANÁLISIS Y DESARROLLO DE SOFTWARE

CENTRO DE INDUSTRIA Y CONSTRUCCIÓN – REGIONAL TOLIMA

SENA AMBIENTE VIRTUAL

2024

**Tabla de Contenido**

|  |  |  |
| --- | --- | --- |
|  | | *Pág.* |
| ***Introducción*** | | *3* |
| ***Objetivos*** | | *4* |
| ***Desarrollo Frontend con React JS*** | | *5* |
| ***1.*** | ***Diferencia entre React y JSX*** | *5* |
| ***2.*** | ***¿Qué son clases en React?*** | *18* |
| ***3.*** | ***Principales eventos de React*** |  |
| ***4.*** | ***¿Qué es SPA (Aplicación de una sola página)?*** |  |
| ***5.*** | ***Mapa conceptual de React*** |  |
| ***6.*** | ***Conclusiones*** | *20* |

**Introducción**

En la era moderna, el desarrollo web ha evolucionado significativamente y, como resultado, la demanda de desarrolladores web capacitados se ha disparado. La introducción a la programación web proporciona una descripción general de los conceptos y tecnologías clave que definen el panorama del desarrollo web contemporáneo. Esto incluye una exploración de marcos de interfaz de usuario populares como React, que ha obtenido una adopción generalizada en los últimos años debido a su eficiencia y flexibilidad. En esta evidencia profundizaremos en la importancia del diseño web responsivo, la accesibilidad web y la creciente importancia de la experiencia del usuario. Como base para nuestro proyecto de Venta en Línea, este contexto sienta las bases para una comprensión integral de las herramientas y técnicas necesarias para el éxito en el campo del desarrollo web moderno.

**Objetivos**

* Implementar componentes Frontend aplicando de manera adecuada los framework disponibles para mejorar la parte visible del usuario final.
* Codificar el frontend utilizando framework.

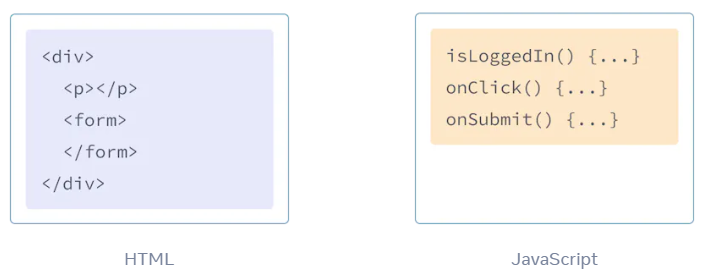
# 

# Desarrollo de Frontend con React JS

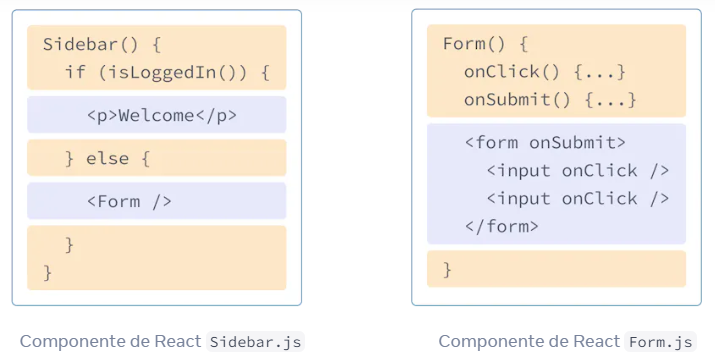
# [Aspectos Preliminares]

1. **Diferencia entre React y JSX**

La Web se ha construido sobre HTML, CSS, y JavaScript. Durante muchos años, los desarrolladores web mantuvieron el contenido en HTML, el diseño en CSS, y la lógica en JavaScript, a menudo en archivos separados. El contenido se marcó dentro del HTML mientras que la lógica de la página vivía por separado en JavaScript:



Pero, a medida que la Web se volvió más interactiva, la lógica determinó cada vez más el contenido. JavaScript estaba a cargo del HTML esto es la razón por la que en React, la lógica de renderizado y el marcado viven juntos en el mismo lugar: componentes.



Mantener juntas la lógica de renderizado y el marcado de un botón, garantiza que permanezcan sincronizados entre sí en cada edición. Por el contrario, los detalles que no están relacionados, como el marcado de un botón y el marcado de una barra lateral, están aislados entre sí, haciendo que sea más seguro cambiar cualquiera de ellos por su cuenta.

Cada componente de *React* es una función de *JavaScript* que puede contener algún marcado que React muestra en el navegador. Los componentes de React usan una extensión de sintaxis llamada *JSX* para representar el marcado. JSX se parece mucho a HTML, pero es un poco más estricto y puede mostrar información dinámica. La mejor manera de comprender esto es convertir algunos marcados HTML en marcado JSX.

***JSX*** *y* ***React*** *son independientes. A menudo se usan en conjunto, pero se pueden usar de forma separada.* ***JSX*** *es una extensión de sintaxis, mientras* ***React*** *es una biblioteca de JavaScript.*

1. **¿Qué son las clases en React?**

Las clases son una forma tradicional de crear componentes reutilizables. Se basan en la sintaxis de clases de JavaScript para definir el comportamiento y el estado del componente. Las clases definen componentes en React, se caracterizan por tener propiedades y hacen uso del paradigma denominado programación orientada a componentes. Dichos componentes se representan como clases que heredan de la clase Component cuyo único requerimiento especial es especificar el método render que define cuál será el contenido del mismo.

Dentro de los componentes de clase existe una función constructora que se utiliza para definir el estado inicial del componente. Es muy importante llamar a la función super() dentro del constructor para que los parámetros funcionen adecuadamente.



1. **Principales eventos en React**

Los eventos en React son una parte fundamental para crear interfaces de usuario interactivas. Permiten que los componentes respondan a las acciones del usuario, como clics, movimientos del mouse, entradas de teclado y envíos de formularios.

Manejar eventos en elementos de React es muy similar a manejar eventos con elementos del DOM. Hay algunas diferencias de sintaxis:

* Los eventos de React se nombran usando camelCase, en vez de minúsculas.
* Con JSX se pasa una función como el manejador del evento, en vez de un string.

**Tipos de eventos en React**

React admite una amplia variedad de eventos, incluyendo:

**Eventos de mouse:** onClick, onMouseMove, onMouseOver, onMouseOut, onDrag.

**Eventos de teclado:** onKeyDown, onKeyUp, onKeyPress.

**Eventos de formulario:** onChange, onSubmit, onInput, onFocus.

**Eventos de ventana:** onLoad, onResize, onScroll.

**Eventos sintéticos vs. eventos nativos**

React utiliza un sistema de eventos sintéticos que abstrae las diferencias entre los navegadores y proporciona una interfaz consistente para manejar eventos. Esto significa que no necesitamos preocuparnos por las diferentes implementaciones de eventos en cada navegador.

Sin embargo, también se puede acceder a los eventos nativos del navegador utilizando la API ReactDOM.Events. Esto puede ser útil en algunos casos, como cuando necesitas acceder a información específica del evento que no está disponible en el evento sintético.

React permite añadir controladores de eventos a JSX. Los controladores de eventos son funciones que se ejecutarán en respuesta a interacciones como hacer clic, hover, enfocar inputs en formularios, entre otras. Para agregar un controlador de evento, primero definirá una función y luego se pasará como una prop a la etiqueta JSX apropiada.

Puedes hacer que muestre un mensaje cuando un usuario haga clic siguiendo estos tres pasos:

1. Declara una función llamada handleClick *dentro* de tu componente Button.
2. Implementar la lógica dentro de esa función (utiliza alert para mostrar el mensaje).
3. Agrega onClick={handleClick} al JSX del <button>.



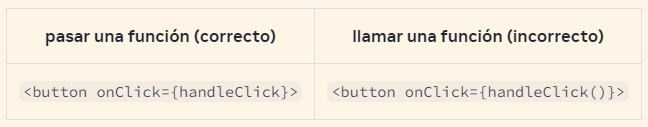
Se definió la función handleClick y luego se pasó como una prop al <button>. handleClick es un **controlador de evento.** Las funciones controladoras de evento:

* Usualmente están definidas *dentro* de los componentes.
* Tienen nombres que empiezan con handle, seguido del nombre del evento.

Por convención, es común llamar a los controladores de eventos como handle seguido del nombre del evento. A menudo se verán así:

onClick={handleClick}, onMouseEnter={handleMouseEnter}

Las funciones que se pasan a los controladores de eventos deben ser pasadas, no llamadas. Por ejemplo:



*La diferencia es sutil. En el primer ejemplo, la función handleClick es pasada como un controlador de evento onClick. Esto le dice a React que lo recuerde y solo llama la función cuando el usuario hace clic en el botón.*

*En el segundo ejemplo, los () al final del handleClick() ejecutan la función inmediatamente mientras se renderiza, sin ningún clic. Esto es porque el JavaScript dentro de { y } en JSX se ejecuta de inmediato.*

1. **¿Qué es SPA [Aplicación de una Sola Página]**
2. **Mapa conceptual de React**

***3. Conclusiones***

Con la realización de esta practica se lograr afianzar los conocimientos hacia el tema de lo importante que es el buen uso de la codificación y probación de elementos esenciales para el desarrollo de software de alta calidad. Su uso permite crear sistemas modulares, reutilizables, confiables y fáciles de mantener. Estos espacios nos permiten mantener y organizar nuestro proyecto para poder estructurar de una manera adecuada los módulos de éste y refinar su código para que cada vez que se vaya avanzando, la finalización sea exitosa. Los módulos de software codificados y probados representan un hito importante en el desarrollo del proyecto. Se ha demostrado la capacidad del equipo para trabajar de manera efectiva y entregar soluciones de software de alta calidad. Las lecciones aprendidas durante el proceso servirán para optimizar el desarrollo de futuros proyectos.