Solucion Taller de Nivelación

MÓDULO SOBRE REACT JS

1. React es una biblioteca de javascript que se utiliza para construir interfaces de usuario interactivas y eficientes. Sus principales características son los componentes,virtual DOM,JSX,estados,Reac Router.
2. Un componente es una unidad independiente y reutilizable de interfaz de usuario que puede contener su propio estado,propiedades y lógica. Estos son la base fundamental para construir aplicaciones en REACT.los componentes se dividen en **Componentes Funcionales:**Son funciones de JavaScript que aceptan propiedades (props) como argumentos y devuelven elementos React (normalmente JSX). No tienen estado interno ni métodos de ciclo de vida. Se recomienda utilizar componentes funcionales siempre que sea posible debido a su simplicidad y rendimiento.
   Componentes de Clases: Son clases de JavaScript que extienden la clase base React.Component. Pueden tener su propio estado interno, métodos de ciclo de vida y se utilizan para manejar lógica más compleja y manejar estados internos.

3. el Virtual DOM en ReactJS actúa como una capa intermedia que mejora la eficiencia en la actualización del DOM real. Permite que React realice comparaciones eficientes y determine las actualizaciones necesarias antes de manipular el DOM real, contribuyendo a un rendimiento óptimo en aplicaciones web. Esta técnica es especialmente valiosa en aplicaciones con interfaces de usuario dinámicas y cambiantes.
4. JSX es una parte integral de ReactJS y contribuye significativamente a la facilidad de desarrollo y la legibilidad del código. Facilita la creación de interfaces de usuario, mejora la integración de lógica dinámica y ayuda a construir componentes reutilizables de manera más intuitiva. Aunque opcional, JSX se ha convertido en una práctica común y altamente recomendada al trabajar con React.
5. Los hooks en React simplifican la lógica y la gestión del estado en los componentes de función, eliminando la necesidad de convertirlos en componentes de clase para utilizar características avanzadas. Permiten un código más modular, reutilizable y fácil de entender. Su adopción ha cambiado la forma en que se estructuran y desarrollan las aplicaciones React.
6.
   - **useState:**Permite que los componentes de función tengan estado local. Este hook devuelve un array con dos elementos: el estado actual y una función para actualizar ese estado. Se utiliza para gestionar variables de estado en componentes de función.
   - **useEffect:**se utiliza para realizar efectos secundarios en componentes de función. Puede emular el ciclo de vida de los componentes de clase, como componentDidMount y componentDidUpdate. Se ejecuta después de que la renderización se ha completado y puede realizar operaciones asíncronas, suscripciones a eventos, etc.
   - **useContext:**se utiliza para suscribir un componente de función al contexto de React y acceder a los valores proporcionados por un Context.Provider

superior. Permite a los componentes acceder a valores globales sin pasar propiedades a través de múltiples niveles de componentes.

7. El uso de Hooks en ReactJS está sujeto a ciertas reglas y convenciones que garantizan su correcto funcionamiento y evitan problemas comunes y las podemos definir como:
    - Solo en Componentes de Función
    - Solo en el Nivel Superior de los componentes
    - Solo llamarse desde componentes de react
8. React Router DOM es una librería que proporciona herramientas de enrutamiento para aplicaciones React de una sola página (Single Page Applications - SPAs). Permite la navegación y gestión de la URL en la aplicación, lo que facilita la creación de aplicaciones con múltiples vistas. y sus principales componentes son:
    - BrowserRouter
    - Route
    - Link
    - useParams
    - useHistory
9.
10.
11. Se ingresa a la carpeta donde queremos crear el proyecto ReactJS con Vite, abrimos el cmd de windows y colocamos el siguiente comando **npm create vite@latest <nombre-de-mi-proyecto>** luego con las flechas del teclado seleccionamos la opción de vite y seguido de ellos la de javascript
    y ya nuestro proyecto estará creado.
12. Primero debemos clonar el proyecto de forma remota en github una vez ya tengamos nuestro repositorio, accedemos a render donde nos logueamos con nuestra cuenta de gitHub y el nos permitira crear un servicio web. ahí seleccionamos el repositorio donde tenemos nuestro JSON y el lo desplegará y nos dará la ruta que podemos usar para acceder a ello.
13. Para desplegar un proyecto con github pages primero debemos subir nuestro repositorio y subir el proyecto, luego vamos a las configuraciones del repositorio y seleccionamos la opción pages donde nos permitira elegir la rama que queremos despegar una vez escojamos la de nuestro gusto volvemos a la página de nuestro repositorio y la esquina derecha seleccionamos el icono de tuerca y chuleamos la segunda opción..

MÓDULO SOBRE GESTION DE ESTADOS Y DATOS CON REACT CONTEXT Y USEREDUCER

1. Es un hook de react el cual nos permite pasar información de un componente a otro a través de un árbol componentes. Se puede decir que los datos se consideran globales y son usados por cualquier componente que pertenezca al árbol.
2. Para la creación de un contexto en React. depende de la distribución de carpetas que manejamos en nuestro caso iremos a Routes.jsx

```
import React, { createContext, useState } from 'react'
export const AppContext = createContext({});
```

definimos la constante e importamos a la función createContext de react.

Para proveer información a el contexto y permitir que todos los componentes que pertenecen al arbol puedan acceder a esta utilizamos la siguiente etiqueta

```
<AppContext.Provider value={globalState}>
```

En este caso la variable que asignamos es globalState.

y para poder consumir esta información desde otros componentes debemos definir importamos el userState y definimos una constante especificando qué valores queremos recuperar en este componente en especifico.

```
import React, { useContext, useState } from 'react';
```

```
const { setIsUserLogged, setUserLogged } = useContext(AppContext);
```

3. Es importante destacar que React Context no siempre es la mejor opción para todos los casos. Para aplicaciones pequeñas o componentes que no necesitan compartir mucho estado, pasar props puede ser más simple y adecuado. Además, el uso excesivo de Context puede hacer que tus componentes sean menos reutilizables y más acoplados, por lo que es importante evaluar cada situación de manera individual.
4. useReducer es una herramienta poderosa cuando necesitas manejar estados más complejos y acciones específicas en tu componente. Sin embargo, para casos más simples, useState sigue siendo una opción válida y más concisa. La elección entre useReducer y useState depende de la complejidad de los requerimientos del estado en tu aplicación.
5. **reducer**: Función que determina cómo cambia el estado en respuesta a una acción.
   initialArg: Valor a partir del cual se calcula el estado inicial.
   init (opcional): Función que especifica cómo se calcula el estado inicial. Si no se proporciona, se utiliza directamente initialArg.
6. useReducer es una herramienta poderosa en React que simplifica la gestión del estado en situaciones más complejas, promoviendo un código más claro, modular y fácil de mantener. Su uso es especialmente beneficioso cuando el estado y las acciones en tu aplicación son más complejos y están interrelacionados.
7. useReducer y useContext se utilizan juntos para proporcionar una solución eficiente y organizada para gestionar el estado global en aplicaciones de React, especialmente cuando la lógica de manejo del estado es más compleja y necesita ser compartida entre varios componentes.
8. tener un sistema de gestión de estado global en aplicaciones React más grandes mejora la organización del código, facilita la comunicación entre componentes, promueve la reutilización de código y contribuye a una aplicación más mantenible y escalable.
9. Gestión de Estado Global:

   Ventaja: React Context junto con useReducer facilita la creación de un sistema de gestión de estado global. Puedes compartir el estado entre diferentes componentes sin tener que pasar props manualmente a través de múltiples niveles de la jerarquía de componentes.

Beneficios: Evita la prop drilling, simplifica la comunicación entre componentes y permite un acceso más sencillo a datos globales desde cualquier parte de la aplicación.
Lógica de Reducción Centralizada:

Ventaja: useReducer proporciona una manera de centralizar la lógica de reducción (cambio de estado) en una única función reductora. Esto es especialmente beneficioso cuando el estado es más complejo y puede tener múltiples transiciones en respuesta a diferentes acciones.
Beneficios: La lógica de reducción está encapsulada, mejora la legibilidad del código, evita la dispersión de la lógica en diferentes partes de la aplicación y facilita la mantenibilidad.
Facilita Pruebas Unitarias:

Ventaja: useReducer y React Context facilitan las pruebas unitarias al separar la lógica de manejo del estado del componente. Al utilizar funciones puras para el reductor, puedes realizar pruebas fácilmente, sin necesidad de interactuar con el DOM o el ciclo de vida del componente.
Beneficios: Mejora la capacidad de realizar pruebas unitarias específicas del estado sin complicaciones, lo que contribuye a la calidad y robustez del código.

10. La migración a un enfoque de estado global con React Context y useReducer suele ser beneficiosa cuando la aplicación crece en complejidad y se presentan desafíos relacionados con la gestión del estado y la comunicación entre componentes. Sin embargo, siempre es importante evaluar si la complejidad adicional de un estado global es justificada para los requisitos específicos de tu aplicación.