Trabajo Práctico N°4 – React

1- Responder las siguientes preguntas:

a- ¿Qué es JSX?

JSX es una extensión de sintaxis de JavaScript que se utiliza en React para definir la estructura de la interfaz de usuario (UI). JSX permite escribir código HTML y JavaScript en el mismo archivo, lo que hace que sea más fácil y rápido desarrollar componentes de UI en React.

function MiComponente() {

return (

<div className="mi-clase">

<h1>Mi Título</h1>

<p>Mi contenido de ejemplo</p>

</div>

);

}

b- ¿Qué es un estado en React?

El estado se utiliza para almacenar información importante que debe mantenerse actualizada en una aplicación de React.

Para utilizar el estado en un componente de React, se utiliza la función useState que se proporciona en el paquete react. Esta función devuelve un array con dos elementos: el primer elemento es el estado actual y el segundo elemento es una función para actualizar el estado.

import { useState } from 'react';

function Contador() {

const [count, setCount] = useState(0);

return (

<div>

<p>Has presionado el botón {count} veces</p>

<button onClick={() => setCount(count + 1)}>Presioname</button>

</div>

);

}

c- ¿Qué es un componente?

En React, un componente es una pieza reutilizable e independiente de código que define una parte de la interfaz de usuario.

Son reutilizables y escalables.

Puede ser un botón, un menú, un campo de entrada, un encabezado, un pie de página o cualquier otra cosa que se pueda representar en la interfaz de usuario.

El componente se puede utilizar en cualquier lugar de la aplicación, simplemente importándolo y llamándolo como cualquier otra función.

import React from 'react';

function Encabezado(props) {

return (

<header>

<h1>{props.titulo}</h1>

<p>{props.descripcion}</p>

</header>

);

}

export default Encabezado;

d- ¿Para qué sirve el hook useState?

El hook useState es uno de los más utilizados en React y sirve para agregar estado a los componentes de función en React.

El hook useState acepta un argumento que es el valor inicial del estado y devuelve una matriz con dos elementos: el valor actual del estado y una función que permite actualizar el valor del estado.

import React, { useState } from 'react';

function Contador() {

const [contador, setContador] = useState(0);

function incrementar() {

setContador(contador + 1);

}

return (

<div>

<p>El contador es: {contador}</p>

<button onClick={incrementar}>Incrementar</button>

</div> );}

e- ¿Para qué sirve el hook useEffect?

El hook useEffect es otro de los hooks más importantes en React y sirve para ejecutar efectos secundarios en un componente de función, tales como la actualización del título de la página, la obtención de datos de una API, la suscripción a un evento del navegador, entre otros.

El hook useEffect acepta dos argumentos: una función y una lista de dependencias. La función que se pasa como primer argumento se ejecuta después de que el componente se haya renderizado por primera vez, y luego se ejecuta de nuevo cada vez que alguna de las dependencias cambia. Si no se especifican dependencias, la función se ejecutará cada vez que el componente se renderice.

En resumen, el hook useEffect es una herramienta muy útil en React que permite ejecutar efectos secundarios en un componente de función, tales como la actualización del título de la página, la obtención de datos de una API, la suscripción a un evento del navegador, entre otros.

import React, { useState, useEffect } from 'react';

function Contador() {

const [contador, setContador] = useState(0);

useEffect(() => {

document.title = `Contador: ${contador}`;

}, [contador]);

function incrementar() {

setContador(contador + 1);

}

return (

<div>

<p>El contador es: {contador}</p>

<button onClick={incrementar}>Incrementar</button>

</div>

);

}

f- ¿Qué es el contexto en React?

En resumen, el contexto en React es una forma de compartir datos entre componentes sin tener que pasar explícitamente las props de un componente a otro a través de la jerarquía de componentes.

El contexto se define mediante la creación de un objeto Context utilizando la función createContext(). Este objeto Context proporciona dos componentes de React: Provider y Consumer.

El componente Provider se utiliza para envolver los componentes que necesitan acceder a los datos del contexto. Este componente acepta un prop llamado value que contiene los datos que se desean compartir. Cada vez que el valor del contexto cambia, todos los componentes que lo consumen se vuelven a renderizar.

El componente Consumer se utiliza para acceder a los datos del contexto en un componente secundario. Este componente acepta una función como su hijo y proporciona el valor actual del contexto como argumento.

import React, { createContext, useState } from 'react';

const ContadorContext = createContext();

function Contador() {

const [contador, setContador] = useState(0);

function incrementar() {

setContador(contador + 1);

}

return (

<ContadorContext.Provider value={contador}>

<div>

<p>El contador es: {contador}</p>

<button onClick={incrementar}>Incrementar</button>

<ContadorHijo />

</div>

</ContadorContext.Provider>

);

}

function ContadorHijo() {

return (

<ContadorContext.Consumer>

{contador => <p>El contador desde el hijo es: {contador}</p>}

</ContadorContext.Consumer>

);

}

2- Consumir una API (a elección) o utilizando alguna de las siguientes:

Pokimons: https://pokeapi.co/api/v2/pokemon

Libros: https://www.etnassoft.com/api/v1/get/?any\_tags=[html,css,javascript]

Perros Pug: https://dog.ceo/api/breed/pug/images

Personajes de Disney: https://api.disneyapi.dev/characters

Juegos: https://www.freetogame.com/api/games

Ricky Martin: https://rickandmortyapi.com/api/character

Yugioh:https://db.ygoprodeck.com/api/v7/cardinfo.php?type=%27Normal%20Mon

ster%27

MMO games: https://www.mmobomb.com/api1/games

Star Wars: https://swapi.dev/api/people

Mas...: https://github.com/public-apis/public-apis

Y crear una aplicación que realice lo siguiente:

Listar los elementos obtenidos de la API

Eliminar un elemento de la lista (opcional)

Detalles a tener en cuenta:

La aplicación debe contar con una sola pagina

Debe tener un botón que al presionarlo realice la petición y cargue la lista

El sistema debe contener al menos un componente hijo en la carpeta components

El sistema debe hacer uso de los hooks de React (el que sea necesario)

No usar chat GPT para hacer todo...