Lexpin

React

¿Qué es React?

React es una biblioteca de JavaScript de código abierto utilizada para construir interfaces de usuario. Fue desarrollada por Facebook y es mantenida por Facebook y una gran comunidad de desarrolladores. React es muy popular y se utiliza en la construcción de aplicaciones web y móviles (React Native), así como en la creación de componentes reutilizables.

Hay varias razones por las que React es una herramienta popular para el desarrollo de aplicaciones.

- Utiliza un modelo de programación declarativo, lo que significa que los desarrolladores solo necesitan especificar qué debe hacer la interfaz de usuario y React se encarga de actualizarla en consecuencia.

- Permite la creación de componentes reutilizables, lo que acelera el proceso de desarrollo y reduce la duplicación de código.

- Cuenta con una gran comunidad de desarrolladores, una amplia documentación y recursos disponibles en línea.

¿Cómo se Instala?

Para instalar React en Windows y Mac, se requiere tener **Node.js** y **NPM** (Node Package Manager) instalados.

A continuación, sigue los siguientes pasos para crear un proyecto React:

- Abre una terminal y navega hasta la carpeta donde desea crear su proyecto.
- Ejecuta el siguiente comando para crear un nuevo proyecto React:
  npx create-react-app my_proyect

  Nota: No uses letras mayúsculas, "***my_proyect***" es el nombre del proyecto, puedes reemplazarlo con el nombre que desees.

- Una vez que se complete la instalación, navega hasta el directorio del proyecto:
  cd my_proyect

- Ahora, puede ejecutar el proyecto utilizando el siguiente comando:
  npm start
  Esto abrirá una nueva ventana del navegador en la dirección http://localhost:3000, donde podrás ver tu aplicación React y comenzar a desarrollar.

¿Qué es JSX?

JSX es una extensión de sintaxis para JavaScript que permite escribir código HTML en un archivo de JavaScript. JSX se utiliza para definir la estructura de la interfaz de usuario de una aplicación.

JSX se utiliza para definir componentes de React mediante la sintaxis de etiquetas similares a HTML. Ejemplo: Componente.jsx

¿Qué son Componentes?

Los componentes en React son bloques de código reutilizables que permiten a los desarrolladores dividir una interfaz de usuario en partes pequeñas y manejables. Cada componente es independiente y se puede utilizar en diferentes partes de una aplicación sin afectar el comportamiento de otros componentes.

Los componentes pueden contener HTML, CSS, JavaScript, además de otros componentes, y pueden aceptar propiedades (props) y estado (state) para personalizar su comportamiento.

```
function MiPrimerComponente(){
    return (
        <p>Este es mi primer componente!</p>
    );
} <- #1-5 function MiPrimerComponente()
export default MiPrimerComponente;
```

Se puede escribir scf (stateless functional component) para crear la estructura básica de un componente

Lexpin

¿Qué son Vistas?

Las vistas son componentes que se utilizan para mostrar una sección específica de la interfaz de usuario. Se utilizan comúnmente en aplicaciones de una sola página para mostrar diferentes secciones de contenido según la URL de la página actual. Por ejemplo, una aplicación de comercio electrónico podría tener diferentes vistas para mostrar los productos, el carrito de compras y la página de pago. Las vistas también pueden contener lógica y funcionalidades específicas, como formularios y botones.

¿Qué son props?

Las "props" (es la abreviatura de "propiedades") son un mecanismo para pasar datos de un componente padre a un componente hijo. Las props se utilizan para comunicar datos y funciones entre componentes de React y permiten que los componentes sean reutilizables y modularizados.

Para pasar props de un componente padre a un componente hijo, primero se deben definir en el componente padre. Las props se pueden definir como atributos en el elemento de React en el árbol de componentes. Por ejemplo, Para utilizar la prop "value" en un componente parrafo, simplemente debes pasar la prop a cada uno de los párrafos que se renderizaran como una propiedad de cada elemento "p".

¿Qué son Rutas?

Son direcciones URL que los usuarios pueden visitar para acceder a diferentes partes de una aplicación. Permiten a los usuarios navegar por diferentes secciones de la aplicación sin necesidad de cargar una nueva página.

Las rutas se definen utilizando la biblioteca de enrutamiento React Router DOM. Esta biblioteca proporciona herramientas para definir rutas y enlazarlas a componentes. Se instala con el comando: ***npm i react-router-dom***

```
import React from 'react';
import { BrowserRouter, Routes, Route } from "react-router-dom";
import ReactDOM from 'react-dom/client';
import Home from './pages/Home';
import Login from './pages/Login';
import Carrito from './pages/Carrito';
import About from './pages/About'
import Register from './pages/Register';
import Recovery from './pages/Recovery';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />
        <Route path="/carrito" element={<Carrito />} />
        <Route path="/recovery" element={<Recovery />} />
        <Route path="/about" element={<About />} />
    </Routes>
  </BrowserRouter>
); <- #24-47 root.render
```

Lexpin

¿Qué son Hooks?

Los Hooks en React son funciones que permiten a los desarrolladores utilizar el estado y otros aspectos del ciclo de vida de los componentes en los componentes funcionales. Los Hooks permiten a los desarrolladores agregar comportamientos a los componentes sin necesidad de convertirlos en componentes de clase o utilizar patrones de renderizado propios de las clases. Los Hooks más utilizados son useState, useEffect, useRef.

- **useState():** Permite a los desarrolladores agregar estado local a los componentes funcionales. Esto significa que se puede definir un estado inicial en un componente y actualizarlo en función de ciertas acciones o eventos.

- **useEffect():** Permite a los desarrolladores agregar efectos secundarios a los componentes funcionales. Es comúnmente usado para procesos en segundo plano.

- **useRef():** Se utiliza para crear una referencia mutable que se puede usar para almacenar cualquier valor. A diferencia de los estados, los valores almacenados en una referencia no provocarán una actualización del componente al cambiar.

¿Cómo consumir una API?

Dependiendo del tipo de api que vayas a consumir puede que necesites una key(llave) para poder obtener los datos o no.

Consumiendo API Publica:

- **Crear un componente en React**: Se puede crear un nuevo componente en React para mostrar los datos de la API.

- **Realizar la solicitud HTTP**: Dentro del componente, se utiliza **useEffect** para realizar una solicitud HTTP a la API.
    - Para lograr esto se utiliza la función **fetch** para obtener los datos de la API.
    - Luego los datos se deben transformar en un objeto JSON usando el método **json()**.
    - Los datos se establecen en el estado del componente usando (en este ejemplo) setCharacters.

- **Mostrar los datos de la API**: Finalmente, se pueden renderizar los datos de la API en el componente. Para ello usamos el método **map()**.

Lexpin

Ejemplo:

```jsx
import React, { useState, useEffect } from "react";

function APINoKeyFetch() {
    const [personajes, serPersonajes] = useState([]);

    useEffect(() => {
        fetch("https://rickandmortyapi.com/api/character")
            .then(response => response.json())
            //
            .then(datos => {
                serPersonajes(datos.results);
            })
            .catch(error => {
                console.log(error);
            });
    }, []); <- #6-16 useEffect

    return (
        <>
            <h1>Api Fetch (No-Key)</h1>
            <ul>
                {personajes.map(personaje => (
                    <li key={personaje.id}>
                        Nombre: {personaje.name}
                    </li>
                ))} <- #22-26 <ul>
            </ul>
        </>
    ); <- #18-29 return
} <- #3-30 function APINoKeyFetch()

export default APINoKeyFetch;
```

Consumiendo API con Key(llave):

- **Obtén una clave API**: Para consumir la API con key, se necesita una clave API válida. Puedes registrarte para obtener una clave API gratuita en el sitio web de The Movie Database.

- **Crea un componente en React**: crea un nuevo componente en React para mostrar los datos de la API.

- **Realiza la solicitud HTTP**: Dentro del componente, se utiliza **useEffect()** para realizar una solicitud HTTP a la API. En este ejemplo, se utiliza la función **fetch()** para obtener los datos de la API. Se concatena las constantes *llaveDeApi* que contiene la key, *idDePelicula* que contiene el id de la película a mostrar, y por último *idioma* que contiene el idioma del sitio, a la URL utilizando la cadena de consulta api_key.
  Los datos se transforman en un objeto JSON usando el método **json()**. Establece los datos en el estado del componente usando (en este ejemplo) setMovie.

- **Muestra los datos de la API**: Finalmente, se pueden renderizar los datos de la API en el componente. En este ejemplo, se muestra el título y la descripción de la película en un elemento h1 y un elemento p, respectivamente.

Ejemplo:

```
import React, { useState, useEffect } from "react";

function APIWithKeyFetch() {
    const [pelicula, setPelicula] = useState({});

    useEffect(() => {
        const idDePelicula = "315162"
        const llaveDeApi = "1a8c3b228d63655b9dbd957026da4b84";
        const idioma = "es-MX";

        fetch(`https://api.themoviedb.org/3/movie/${idDePelicula}?api_key=${llaveDeApi}&language=${idioma}`)
            .then(response => response.json())
            .then(datos => {
                setPelicula(datos);
            })
            .catch(error => {
                console.log(error);
            });
    }, []); <- #6-19 useEffect

    return (
        <div>
            <h1>{pelicula.title}</h1>
            <p >{pelicula.overview}</p>
            <img src={"https://image.tmdb.org/t/p/w500/"+pelicula.poster_path} alt="Poster"/>
        </div>
    ); <- #21-27 return
} <- #3-28 function APIWithKeyFetch()

export default APIWithKeyFetch;
```