REACT 1

INTRO MANEJADORES DE PAQUETES NPM YARN PNPM

Los **manejadores de paquetes** son herramientas que permiten gestionar dependencias en proyectos, facilitando la instalación, actualización, y mantenimiento de bibliotecas o módulos. Aquí tienes una breve descripción de **pnpm**, **yarn** y **npm**

**Manejadores de paquetes:**<https://www.npmjs.com/> (clásico)

**npm (Node Package Manager):**

* **Descripción:** Es el gestor de paquetes oficial de Node.js y el más ampliamente utilizado.
* **Características:**
  + Viene preinstalado con Node.js.
  + Maneja dependencias en un directorio llamado node\_modules.
  + Su enfoque inicial era simplicidad, pero con el tiempo ha incorporado optimizaciones como el package-lock.json.
  + Más lento comparado con **pnpm** o **yarn**, pero ha mejorado mucho en versiones recientes.

<https://classic.yarnpkg.com/lang/en/>

**2. yarn (Yet Another Resource Negotiator):**

* **Descripción:** Lanzado por Facebook como una alternativa a npm, ofreciendo mayor velocidad y confiabilidad en su tiempo.
* **Características:**
  + Uso de un archivo yarn.lock para asegurar versiones consistentes entre instalaciones.
  + Descarga paquetes de manera paralela, haciéndolo más rápido que npm en sus inicios.
  + Proporciona comandos más intuitivos (yarn add, yarn remove) y características como "plug-and-play" para evitar conflictos de dependencias.

<https://pnpm.io/es/>

Los **manejadores de paquetes** son herramientas que permiten gestionar dependencias en proyectos, facilitando la instalación, actualización, y mantenimiento de bibliotecas o módulos. Aquí tienes una breve descripción de **pnpm**, **yarn** y **npm**:

**1. npm (Node Package Manager):**

* **Descripción:** Es el gestor de paquetes oficial de Node.js y el más ampliamente utilizado.
* **Características:**
  + Viene preinstalado con Node.js.
  + Maneja dependencias en un directorio llamado node\_modules.
  + Su enfoque inicial era simplicidad, pero con el tiempo ha incorporado optimizaciones como el package-lock.json.
  + Más lento comparado con **pnpm** o **yarn**, pero ha mejorado mucho en versiones recientes.

**2. yarn (Yet Another Resource Negotiator):**

* **Descripción:** Lanzado por Facebook como una alternativa a npm, ofreciendo mayor velocidad y confiabilidad en su tiempo.
* **Características:**
  + Uso de un archivo yarn.lock para asegurar versiones consistentes entre instalaciones.
  + Descarga paquetes de manera paralela, haciéndolo más rápido que npm en sus inicios.
  + Proporciona comandos más intuitivos (yarn add, yarn remove) y características como "plug-and-play" para evitar conflictos de dependencias.

**3. pnpm (Performant npm):**

* **Descripción:** Una alternativa moderna que optimiza el almacenamiento y manejo de dependencias, enfocándose en eficiencia.
* **Características:**
  + Usa un almacenamiento global compartido para las dependencias, lo que evita duplicación de archivos en múltiples proyectos.
  + Más rápido y ahorra espacio en disco en comparación con npm y yarn.
  + Es compatible con scripts y paquetes existentes de npm.

node +18

instalar:

 PNPM npm install -g pnpm

VERIFICAR VERSION pnpm -v

ACTUALIZAR pnpm add -g pnpm

PNPM CREATE VITE   
nombre

react   
js

pnpm install

PNPM DEV

CAMBIA =>pack look pnpm lock

tenemos que empezar a distinguir lo importante de lo secundario

CUANDO SE RENDERIZA EN REACT  
SON 3

al cargar la pagina

cuando cambia un estado

cuando cambia una prop

cuando cambia el componente padre  
  
tenemos varias tareas por hacer   
mostrar el spinner es la tarea principal los demás son side effects

peticiones

cambiar el renderizado

cambiar url

side effects:

se producen como consecuencia de otras acciones.. cambio de estado

ejemplo realizo una busqueda y atras vienen los side effects…  
muestro un spinner realizó peticion cambio el renderizado

en react el renderizado es lo más importante

el use effect es un nuevo hook que vamos a ver se importa desde react

regla principal un hook se usa dentro de un componente

es una función que no retorna nada  
recibe DOS PARAMETROS  un callback  y un array de despnmdencias

que se ejecuta:  
1- siemrpe

useEffect=(()=>{

console.log(‘use effect’)})

2- [] cuando inicia la app solo una vez

useEffect=(()=>{

console.log(‘use effect’)},[])

MOSTRAR UN EJEMPLO DE COUNT EN LOCAL STORAGE

HAY QUE ACTUALIZAR CADA VEZ POR CADA ACCIÓN

3- array de dependencias con dependencias

useEffect=(()=>{

console.log(‘use effect’)},[count])

solo cuando cambia count

EL USEeFFECT SE EJECUTA DESPUES DEL RENDERIZADO ENTONCES NO HAY DESFASES

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

CREO DOS ESTADOS PARA MOSTRAR QUE CADA VEZ QUE CAMBIO UNO SE EJECUTA EL USE EFFECT Y SI PONGO UNO EN EL ARRAY DE DEPENDENCIAS SE EJECUTA CADA VEZ QUE CAMBIA UNO

NO PONER VALOR FIJO PORQUE NO SE VA A VOLVER A RENDERIZAR SOLO EN LA PRIMERA EJECUCIÓN DE CARGA

cuidado con los loops infinitos

no puedo cambiar el estado que está en la dependencia.. si cambia se vuelve a renderizar

<https://es.react.dev/reference/react/useEffect>

HACEMOS EJERCICIO DE CONTADOR CON LOCAL STORAGE , input

—----------------------

EMPEZAMOS A USAR UNA API https://pokeapi.co/

-CREAMOS API

URL BASE  [https://pokeapi.co/api/v2/pokemon](https://pokeapi.co/api/v2/pokemon/ditto)SOLO UNO  <https://pokeapi.co/api/v2/pokemon/ditto>

USAMOS:

 USE STATE guarda la data en el estado Y USE EFFECTS hace la petición a la api

no hacer peticiones demás

meter clg para ver cuando se ejecuta

las api generalmente son gratuitas péro tienen un limite de pet diarias

Captura de pantalla de un celular

Descripción generada automáticamente

REACT 2

**Axios**: Una biblioteca para realizar solicitudes HTTP de forma sencilla. crud cuales son   
<https://www.npmjs.com/package/axios>

fetch es nativo de js

tenemos una ventaja en la sintaxis mas facil

debemos mantener la consistencia con el manejador de paquete que uso

USO AXIOS IMPORTO DESDE AXIOS

no hay mucha ventaja en un get pero si en un post quizas

Las principales diferencias entre hacer una solicitud POST con fetch y con axios radican en la facilidad de uso, compatibilidad y características integradas. Aquí tienes un desglose:

1. **Facilidad de uso**:
   * fetch: Es nativo de JavaScript y no necesita instalación, pero requiere más configuración para manejar errores y datos JSON.
   * axios: Es una biblioteca externa que simplifica las solicitudes HTTP, manejando automáticamente la transformación de datos JSON y errores.
2. **Compatibilidad con navegadores antiguos**:
   * fetch: No es compatible con navegadores antiguos como Internet Explorer, a menos que uses un polyfill.
   * axios: Compatible con una mayor variedad de navegadores gracias a su soporte incorporado.
3. **Configuración de encabezados y datos**:
   * fetch: Requiere más pasos para configurar encabezados y convertir datos de/para JSON.
   * axios: Simplifica este proceso, ya que los encabezados comunes y la conversión de datos se manejan automáticamente.
4. **Manejo de errores**:
   * fetch: Solo lanza errores si hay un problema de red; necesitas verificar manualmente el estado de la respuesta.
   * axios: Lanza errores para respuestas con códigos de estado fuera del rango 2xx, lo que lo hace más intuitivo para manejar errores.

**Ejemplo de POST con fetch:**

javascript

Copiar código

const postDataWithFetch = async () => {

    try {

        const response = await fetch('https://jsonplaceholder.typicode.com/posts', {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json'

            },

            body: JSON.stringify({

                title: 'Fetch Example',

                body: 'This is an example using fetch',

                userId: 1

            })

        });

        if (!response.ok) {

            throw new Error(`HTTP error! status: ${response.status}`);

        }

        const data = await response.json();

        console.log('Response with fetch:', data);

    } catch (error) {

        console.error('Error with fetch:', error);

    }

};

postDataWithFetch();

**Ejemplo de POST con axios:**

javascript

Copiar código

import axios from 'axios';

const postDataWithAxios = async () => {

    try {

        const response = await axios.post('https://jsonplaceholder.typicode.com/posts', {

            title: 'Axios Example',

            body: 'This is an example using axios',

            userId: 1

        });

        console.log('Response with axios:', response.data);

    } catch (error) {

        console.error('Error with axios:', error);

    }

};

postDataWithAxios();

**Diferencias clave en los ejemplos:**

* **Con fetch**: Es necesario agregar explícitamente los encabezados y convertir la respuesta a JSON.
* **Con axios**: Todo esto se maneja automáticamente, haciendo que el código sea más compacto y legible.

Ambos métodos tienen sus ventajas según el contexto. Para proyectos más simples, fetch puede ser suficiente, mientras que axios es ideal para aplicaciones más complejas.

podemos añadir un config/axios.js instanciar y exportar las funciones

USE CONTEXT

Manejador de estado global:

Zustan   
dredux  
context

use context es un hook que nos va a permitir acceder al estado de manera global   
sin  necesidad de tener que pasar por props de componente a componente  
  
**Resumen del flujo:**

**Paso 1: Crea el contexto del carrito**

Definimos un contexto

export const UserContext = createContext() SE IMPORTA  
  
2-envolvemos la app con el <UserContext.provider  value={ todo lo que uso aca va a ser global, si es mas de una cosa debemos usar un objeto}> pueden haber mas de uno

creo el estado   
const [user, setUser] = useState(null)

const handleLogin = () =>{

setUser({

name: “cintia”,

email:”[cintia@gmail.com](mailto:cintia@gmail.com)”

})

}

QUE SOLEMOS PONER EN UN CONTEXT:

* INFO DE USUARIO
* TEMA CLARO/OSCURO
* CARRITOS

3- desde los demás componentes puedo acceder creando una instancia de la fucnión useContetx

cont user = useContext(paso por parametro que useconetx voy a usar )

REACT HOOK FORM

React hook form es un paquete para validar formularios:

FORMIK

[REACT HOOK FORM](https://www.react-hook-form.com/get-started/)

INSTALAMOS DEPENDENCIA   
npm install react-hook-form  
yarn add react-hook-form

pnpm add react-hook-form

VERIFICAMOS INTSTALACIÓN

es un paquete muy completo y se puede complementar con otros.  
tiene su propio context  
podemos hacer formularios en pasos componetizando el form   
  
con este paquete podemos usar el context para englobar la data

Creamos el form

creamos la constante methods e importamos de react hook form lo que vamos a usar

//USO DESTRUCTURING PARA TRAER SOLO LO QUE NECESITO

  const  {register, handleSubmit}  = useForm()

 REGISTER:registra cada input para y manejarlo con el hook

HANDLESUBMIT : envía el submit

el evento on submit mediante el handle submit ejecuta la función de mi login

HAY PAQUETES DE VALIDACIÓN COMO ZOD Y YUP SE ARMA EL ESQUEMEA DE VALIDACIÓN