La función fetchPosts() es una función asíncrona diseñada para obtener y mostrar posts desde una API. Vamos a desglosarla paso por paso:

* async function fetchPosts() {
* const url = "https://jsonplaceholder.typicode.com/posts";
* const container = document.getElementById("post-container");
* Se declara una función asíncrona llamada fetchPosts().
* Se define una constante url que contiene la dirección de la API de la cual se obtendrán los posts. En este caso, se está utilizando JSONPlaceholder, una API de prueba que proporciona datos ficticios.
* Se obtiene una referencia al elemento del DOM con el id "post-container", que es donde se mostrarán los posts.
* if (!container) {
* console.error("Container element not found");
* return;
* }
* Se declara una función asíncrona llamada fetchPosts().
* Se define una constante url que contiene la dirección de la API de la cual se obtendrán los posts. En este caso, se está utilizando JSONPlaceholder, una API de prueba que proporciona datos ficticios.
* Se obtiene una referencia al elemento del DOM con el id "post-container", que es donde se mostrarán los posts.
* if (!container) {
* console.error("Container element not found");
* return;
* }
* Se verifica si se encontró el elemento contenedor. Si no se encuentra, se muestra un error en la consola y se termina la función.
* try {
* // Mostrar un indicador de carga
* container.innerHTML = "<p>Cargando posts...</p>";
* const response = await fetch(url);
* if (!response.ok) {
* throw new Error(`HTTP error! status: ${response.status}`);
* }
* Se inicia un bloque try para manejar posibles errores.
* Se muestra un mensaje de carga en el contenedor.
* Se realiza una petición a la API usando fetch() y se espera la respuesta con await.
* Se verifica si la respuesta es exitosa (status 200-299). Si no lo es, se lanza un error.

**Usar un array local con tus posts**

Puedes definir tus propios posts directamente en el código como un array de objetos JavaScript. Esta es la opción más sencilla y no requiere una conexión a Internet ni una base de datos.

**Pasos:**

**Crea un array con tus posts :**

Define un array que contenga tus posts. Cada post debe ser un objeto con propiedades como id, title, body, y userId.

**Modifica la función fetchPosts :**

En lugar de realizar una solicitud HTTP a una API, simplemente usa el array local.

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Mis Propios Posts</title>

<style>

.post {

border: 1px solid #ccc;

padding: 15px;

margin-bottom: 15px;

border-radius: 5px;

}

.post h2 {

font-size: 1.2em;

margin-bottom: 10px;

}

.post p {

font-size: 1em;

color: #333;

}

.post footer {

display: flex;

justify-content: space-between;

font-size: 0.9em;

color: #666;

}

</style>

</head>

<body>

<h1>Lista de Mis Posts</h1>

<!-- Contenedor donde se mostrarán los posts -->

<div id="post-container"></div>

<script>

// Array con tus propios posts

const myPosts = [

{

id: 1,

userId: 101,

title: "Mi primer post",

body: "Este es el contenido de mi primer post. ¡Es emocionante!"

},

{

id: 2,

userId: 102,

title: "Mi segundo post",

body: "Aquí hay más información sobre lo que quiero compartir."

},

{

id: 3,

userId: 103,

title: "Un post sobre tecnología",

body: "La tecnología está cambiando el mundo de muchas maneras."

}

];

async function fetchPosts() {

const container = document.getElementById("post-container");

if (!container) {

console.error("Container element not found");

return;

}

try {

// Mostrar un indicador de carga

container.innerHTML = "<p>Cargando posts...</p>";

// Simular una carga asíncrona (opcional)

await new Promise(resolve => setTimeout(resolve, 1000));

// Limpiar el contenedor

container.innerHTML = "";

// Crear y añadir elementos para cada post

myPosts.forEach(post => {

const postElement = document.createElement("article");

postElement.className = "post";

postElement.innerHTML = `

<h2>${post.title}</h2>

<p>${post.body}</p>

<footer>

<span>ID del Post: ${post.id}</span>

<span>ID del Usuario: ${post.userId}</span>

</footer>

`;

container.appendChild(postElement);

});

} catch (error) {

console.error("Error fetching posts:", error);

container.innerHTML = "<p>Error al cargar los posts. Por favor, intente más tarde.</p>";

}

}

// Llamar a la función cuando se carga la página

document.addEventListener('DOMContentLoaded', fetchPosts);

</script>

</body>

</html>

El array **myPosts** contiene tus propios posts.

La función **fetchPosts** ya no realiza una solicitud HTTP. En su lugar, utiliza el array **myPosts**.

Se simula una carga asíncrona con **setTimeout** para que el mensaje "Cargando posts..." aparezca durante 1 segundo (esto es opcional).

**Usar una base de datos o backend personalizado**

Si necesitas una solución más avanzada (por ejemplo, si los posts cambian dinámicamente o si quieres permitir que los usuarios agreguen nuevos posts), puedes crear un backend con Node.js, Python, PHP, etc., y almacenar los posts en una base de datos.

En este caso, tu backend proporcionará una API que devuelva los posts en formato JSON, y tu frontend hará una solicitud HTTP a esa API (similar a cómo funciona el ejemplo original con **https://jsonplaceholder.typicode.com/posts**).

async function fetchPosts() {

    const url = "https://jsonplaceholder.typicode.com/posts ";

    const container = document.getElementById("post-container");

    if (!container) {

    console.error("Container element not found");

    return;

    }

    try {

    // Mostrar un indicador de carga

    container.innerHTML = "<p>Cargando posts...</p>";

    const response = await fetch(url);

    if (!response.ok) {

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

    }

    const posts = await response.json();

    // Limpiar el contenedor

    container.innerHTML = "";

    // Crear y añadir elementos para cada post

    posts.forEach(post => {

    const postElement = document.createElement("article");

    postElement.className = "post";

    postElement.innerHTML = `

    <h2>${post.title}</h2>

    <p>${post.body}</p>

    <footer>

    <span>ID del Post: ${post.id}</span>

    <span>ID del Usuario: ${post.userId}</span>

    </footer>

    `;

    container.appendChild(postElement);

    });

    } catch (error) {

    console.error("Error fetching posts:", error);

    container.innerHTML = "<p>Error al cargar los posts. Por favor, intente más tarde.</p>";

    }

    }

    // Llamar a la función cuando se carga la página

    document.addEventListener('DOMContentLoaded', fetchPosts);

**HTML**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Fetch Posts</title>

<style>

.post {

border: 1px solid #ccc;

padding: 10px;

margin: 10px 0;

}

</style>

</head>

<body>

<div id="post-container"></div>

<script src="Untitled-1.js"></script>

</body>

</html>

**Usar un archivo JSON local**

Si tienes muchos posts o prefieres separar los datos del código, puedes guardar tus posts en un archivo JSON y cargarlos desde allí.

**Pasos:**

**Crea un archivo JSON** : Guarda tus posts en un archivo llamado **posts.json**.