# Understanding useEffect in React

The `useEffect` Hook is one of the most powerful and frequently used hooks in React. It enables functional components to perform side effects, such as fetching data, updating the DOM, and setting up timers. The `useEffect` hook is crucial for creating dynamic and responsive applications. This document explains `useEffect` from the basics to advanced usage, complete with a detailed code example.

## Basics of useEffect

In its simplest form, `useEffect` is a function that takes two parameters: an effect callback function and a dependency array. It allows a function to run after the component renders.  
  
Syntax:  
  
**useEffect(() => {  
 // Side effect logic here  
}, [dependencies]);**The dependency array is optional and can contain variables that trigger the effect callback when they change. If no dependency array is provided, `useEffect` runs after every render. If an empty array `[]` is provided, `useEffect` runs only once after the first render.

## Example: Fetching Posts Using useEffect

In the example below, `useEffect` is used to fetch posts from an API when the component loads. The dependency array ensures that the effect runs only once. Additional states manage loading and error statuses to provide feedback to the user. This demonstrates the use of `useEffect` with asynchronous data fetching.

# The use state posts are different and json posts are different

{

"posts": [

{

"id": 1,

"title": "His mother had always taught him",

"body": "His mother had always taught him not to ever think of himself as better than others. He'd tried to live by this motto. He never looked down on those who were less fortunate or who had less money than him. But the stupidity of the group of people he was talking to made him change his mind.",

"tags": [

"history",

"american",

"crime"

],

"reactions": {

"likes": 192,

"dislikes": 25

},

"views": 305,

"userId": 121

},

{

"id": 2,

"title": "He was an expert but not in a discipline",

"body": "He was an expert but not in a discipline that anyone could fully appreciate. He knew how to hold the cone just right so that the soft server ice-cream fell into it at the precise angle to form a perfect cone each and every time. It had taken years to perfect and he could now do it without even putting any thought behind it.",

"tags": [

"french",

"fiction",

"english"

],

"reactions": {

"likes": 859,

"dislikes": 32

},

"views": 4884,

"userId": 91

},

### Code Example

import axios from 'axios';  
import React, { useEffect, useState } from 'react';  
  
const PostDetail = () => {  
 const [posts, setPosts] = useState([]); // State to hold posts data  
 const [loading, setLoading] = useState(true); // State to manage loading status  
 const [error, setError] = useState(null); // State to handle errors  
 const [lim, setLim] = useState(100);  
  
 useEffect(() => {  
 const fetchPosts = async () => {  
 try {  
 setLoading(true); // Start loading  
 const response = await axios.get(`https://dummyjson.com/posts?limit=${lim}`);  
 console.log('Fetched Posts:', **response.data.posts**); // Log fetched posts that is list is extracted json list name is posts

//donot get confused bcs set post is not related to json posts object  
 setPosts(**response.data.posts**); // Update state with fetched posts  
 } catch (err) {  
 console.error('Error fetching posts:', err.message); // Log error message  
 setError(err.message); // Update error state  
 } finally {  
 setLoading(false); // Stop loading  
 }  
 };  
  
 fetchPosts();  
 }, [lim]); // Runs whenever `lim` changes  
  
 return (  
 <div className='container'>  
 {loading ? (  
 <p>Loading...</p>  
 ) : error ? (  
 <p>Error: {error}</p>  
 ) : (  
 posts.map((post) => (  
 <div key={post.id}>  
 <h3>{post.title}</h3>  
 <p>{post.body}</p>  
 </div>  
 ))  
 )}  
 </div>  
 );  
};  
  
export default PostDetail;

## Advanced Techniques with useEffect

**Some advanced applications of `useEffect` include using cleanup functions, handling dependencies carefully to avoid infinite loops, and managing effects that rely on external state or props. Cleanup functions are especially useful for actions like canceling timers or removing event listeners when a component unmounts.**

Example with Cleanup:

useEffect(() => {  
 const timer = setInterval(() => console.log('Tick'), 1000);  
 return () => clearInterval(timer); // Cleanup on unmount  
}, []);

# now wen you fetch single post ,just response.data

import axios from 'axios';

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

import { useParams } from 'react-router-dom';

**const** PostDetail = () **=>** {

**const** { id } = useParams();

**const** [post, setPost] = useState(null);

**const** [loading, setLoading] = useState(true);

**const** [error, setError] = useState(null);

  useEffect(() **=>** {

**const** fetchPost = **async** () **=>** {

      try {

        setLoading(true);

**const** response = await axios.get(`https://dummyjson.com/posts/${id}`);

        console.log('Fetched Post:', response.data); *// Log fetched post*

        setPost(response.data); *// Access the post object directly*

      } catch (err) {

        console.error('Error fetching post:', err.message); *// Log error message*

        setError(err.message); *// Set error message*

      } finally {

        setLoading(false); *// Stop loading*

      }

    };

    fetchPost();

  }, [id]); *// Add `id` as a dependency to refetch on id change*

  return (

    <div className="max-w-full max-h-full bg-violet-950 flex justify-center gap-3 flex-wrap pt-20">

      {loading ? (

        <p className="text-3xl text-white">Loading...</p>

      ) : error ? (

        <p>Error: {error}</p>

      ) : post ? (

        <div className="p-2 text-justify text-white bg-violet-700 w-[30rem] h-fit rounded-md bg-opacity-[20%] border-violet-400 border-2" key={post.id}>

          <h3 className="bg-violet-800 text-center first-letter:uppercase rounded-md shadow-sm shadow-violet-400 font-thin">

            <span className="bg-violet-900 p-1 text-left rounded-md shadow-sm shadow-violet-300 border-spacing-2 border-2">Title:</span>

            <> </>

            {post.title}

          </h3>

          <p className="bg-blue-900 bg-opacity-[40%] text-center text-sm mt-2 p-1 rounded-md border-4 border-double border-blue-600">

            {post.body}

          </p>

        </div>

      ) : null}

      <img src={`https://dummyjson.com/image/400x200/282828?fontFamily=pacifico&text=${toString(post.title)}`} alt="" />

    </div>

  );

};

export default PostDetail;