**What is Axios?**

Axios is an HTTP client library that allows you to make requests to a given endpoint(API)

This could be an external API or your own backend Node.js server, for example.

By making a request, you expect your API to perform an operation according to the request you made.

For example, if you make a GET request, you expect to get back data to display in your application.

## Why Use Axios in React

There are a number of different libraries you can use to make these requests, so why choose Axios?

Here are **five reasons** why you should use Axios as your client to make HTTP requests:

1. It has good defaults to work with JSON data. Unlike alternatives such as the Fetch API, you often don't need to set your headers. Or perform tedious tasks like converting your request body to a JSON string.
2. Axios has function names that match any HTTP methods. To perform a GET request, you use the .get() method.
3. Axios does more with less code. Unlike the Fetch API, you only need one .then() callback to access your requested JSON data.
4. Axios has better error handling. Axios throws 400 and 500 range errors for you. Unlike the Fetch API, where you have to check the status code and throw the error yourself.
5. Axios can be used on the server as well as the client. If you are writing a Node.js application, be aware that Axios can also be used in an environment separate from the browser.

## How to Set Up Axios with React

Using Axios with React is a very simple process. You need three things:

1. An existing React project
2. To install Axios with npm/yarn
3. An API endpoint for making requests

The quickest way to create a new React application is by going to [react.new](https://react.new/).

If you have an existing React project, you just need to install Axios with npm (or any other package manager):

npm install axios

## How to Create an Axios Instance

If you look at the previous examples, you'll see that there's a baseURL that you use as part of the endpoint for Axios to perform these requests.

However, it gets a bit tedious to keep writing that baseURL for every single request. Couldn't you just have Axios remember what baseURL you're using, since it always involves a similar endpoint?

In fact, you can. If you create an instance with the .create() method, Axios will remember that baseURL, plus other values you might want to specify for every request, including headers:

The one property in the config object above is baseURL, to which you pass the endpoint.

The .create() function returns a newly created instance, which in this case is called client.

Then in the future, you can use all the same methods as you did before, but you don't have to include the baseURL as the first argument anymore. You just have to reference the specific route you want, for example, /, /1, and so on.

## How to Use the Async-Await Syntax with Axios

A big benefit to using promises in JavaScript (including React applications) is the async-await syntax.

Async-await allows you to write much cleaner code without then and catch callback functions. Plus, code with async-await looks a lot like synchronous code, and is easier to understand.

But how do you use the async-await syntax with Axios?

However in useEffect, there's an async function called getPost.

Making it async allows you to use the await keword to resolve the GET request and set that data in state on the next line without the .then() callback.

Note that the getPost function is called immediately after being created.

Additionally, the deletePost function is now async, which is a requirement to use the await keyword which resolves the promise it returns (every Axios method returns a promise to resolve).

After using the await keyword with the DELETE request, the user is alerted that the post was deleted, and the post is set to null.

As you can see, async-await cleans up the code a great deal, and you can use it with Axios very easily.

## How to Create a Custom useAxios Hook

Async-await is a great way to simplify your code, but you can take this a step further.

Instead of using useEffect to fetch data when the component mounts, you could create your own custom hook with Axios to perform the same operation as a reusable function.

While you can make this custom hook yourself, there's a very good library that gives you a custom useAxios hook called use-axios-client.

First, install the package:

npm install use-axios-client

To use the hook itself, import useAxios from use-axios-client at the top of the component.

Because you no longer need useEffect, you can remove the React import:

Now you can call useAxios at the top of the app component, pass in the URL you want to make a request to, and the hook returns an object with all the values you need to handle the different states: loading, error and the resolved data.

In the process of performing this request, the value loading will be true. If there's an error, you'll want to display that error state. Otherwise, if you have the returned data, you can display it in the UI.

The benefit of custom hooks like this is that it really cuts down on code and simplifies it overall.

If you're looking for even simpler data fetching with Axios, try out a custom useAxios hook like this one.