You may be wondering at this point...

Why would I use Axios when I can just use the Fetch API?

This is a great question.

Firstly, I want to be clear. **Axios is not the same as Fetch**. In fact, under the hood, Axios makes an HTTP request using the `XMLHttpRequest()` object.

That's right, it doesn't use the Fetch API.

What!

This doesn't make sense, does it? We all know that the old XHR object was not promise-based, so how was it possible that we used the `.then()` and `.catch()` blocks when we used Axios?

Answer: Axios also uses the Promise API. In other words, Axios combines the XHR object and Promises to make AJAX requests. This is why you will often hear developers say that Axios is a Fetch wrapper (but just remember, strictly speaking, it's not).

Now, let me get back to the main question—why do some developers use Axios when they could just use the Fetch API?

The short answer is that Axios gives us a lot of functionality that we can **easily** implement versus using Fetch (keyword being **easily**). Just like any library, all the heavy lifting is done for you.

For example, Axios makes life easier in the following ways:

- Axios automatically transforms the data received in the response. What this means, of course, is that you don't need to do anything special like *response.json()* to get the data.
- Axios automatically converts data to JSON when sending data to a server. When we used Fetch, we had to use JSON.stringify() manually, remember? Just in case you are wondering, you can override this default behavior in Axios if you want.
- It is easy to set a request timeout in Axios as compared to Fetch. In Axios, you

can use the optional `timeout` property in the config object to set the number of milliseconds before the request is aborted. If you want to do the same thing using Fetch, you have to write a lot more code and use the `AbortController` interface.

- Axios has the ability to intercept HTTP requests. HTTP interceptors are used when you want to set a global strategy for how you handle requests and

responses. Fetch does not support this natively, so you will have to be creative to implement it using Fetch.

- Fetch does not have an easy way for us to monitor download and upload progress. You can do it, but it requires working with Streams. Implementing a progress bar in Axios is a lot simpler.
- Axios makes it a little easier to handle simultaneous requests (through the Axios.all() method), whereas, with Fetch, we have to work with the Promise API.

Conclusion

Axios is an alternative to using Fetch.

Remember, it is entirely possible to reproduce the key features of the Axios library using the Fetch API provided by browsers—it will just require a little more code and finesse to do so.