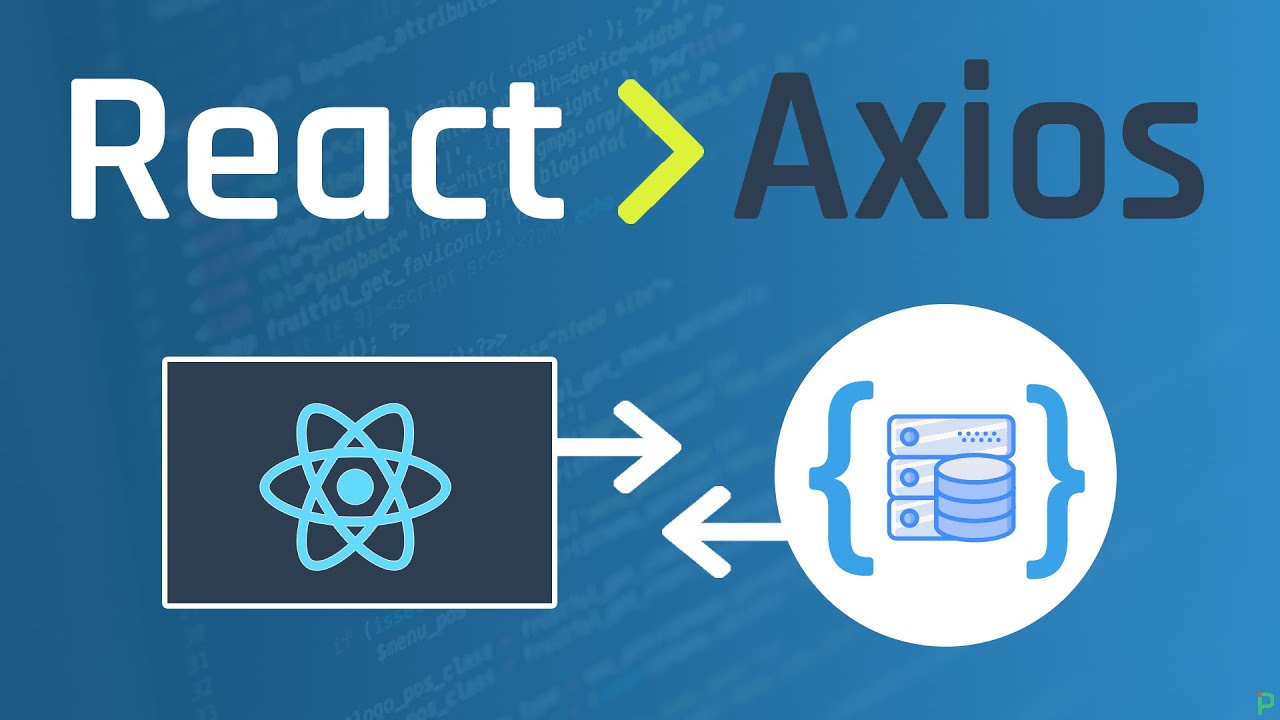
**Axios With React**



In ReactJS, Axios is a library that serves to create HTTP requests that are present externally. It is evident from the fact that we may sometimes in React applications need to get data from the external source. It is quite difficult to fetch such data so that they can be normally shown on the website.

## 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

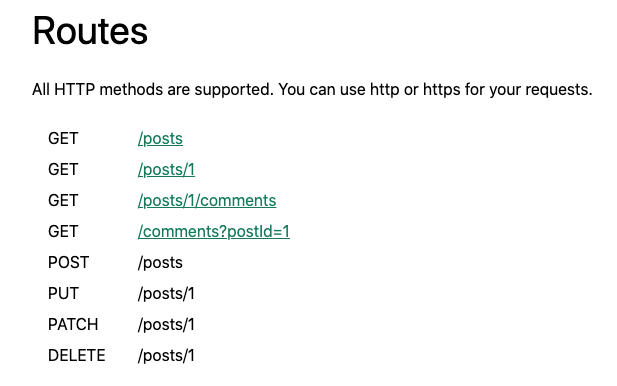
Using Axios with React is a very simple process. You need three things:

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

npm install axios

In this guide, you'll use the JSON Placeholder API to get and change post data.

Here is a list of all the different routes you can make requests to, along with the appropriate HTTP method for each:



## GET Request

<https://jsonplaceholder.typicode.com/posts/>

To fetch data or retrieve it, make a GET request.

import axios from "axios";

import React from "react";

const baseURL = "https://jsonplaceholder.typicode.com/posts/1";

export default function App() {

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

React.useEffect(() => {

axios.get(baseURL).then((response) => {

setPost(response.data);

});

}, []);

if (!post) return null;

return (

<div>

<h1>{post.title}</h1>

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

</div>

);

}

To perform this request when the component mounts, you use the useEffect hook. This involves importing Axios, using the .get() method to make a GET request to your endpoint, and using a .then() callback to get back all of the response data.

## POST Request

import axios from "axios";

import React from "react";

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

export default function App() {

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

React.useEffect(() => {

axios.get(`${baseURL}/1`).then((response) => {

setPost(response.data);

});

}, []);

function createPost() {

axios

.post(baseURL, {

title: "Hello World!",

body: "This is a new post."

})

.then((response) => {

setPost(response.data);

});

}

if (!post) return "No post!"

return (

<div>

<h1>{post.title}</h1>

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

<button onClick={createPost}>Create Post</button>

</div>

);

}

## PUT Request

To update a given resource, make a PUT request.

In this case, you'll update the first post.

To do so, you'll once again create a button. But this time, the button will call a function to update a post:

import axios from "axios";

import React from "react";

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

export default function App() {

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

React.useEffect(() => {

axios.get(`${baseURL}/1`).then((response) => {

setPost(response.data);

});

}, []);

function updatePost() {

axios

.put(`${baseURL}/1`, {

title: "Hello World!",

body: "This is an updated post."

})

.then((response) => {

setPost(response.data);

});

}

if (!post) return "No post!"

return (

<div>

<h1>{post.title}</h1>

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

<button onClick={updatePost}>Update Post</button>

</div>

);

}

In the code above, you use the PUT method from Axios. And like with the POST method, you include the properties that you want to be in the updated resource.

Again, using the .then() callback, you update the JSX with the data that is returned.

## How to Make a DELETE Request

Finally, to delete a resource, use the DELETE method.

As an example, we'll delete the first post.

Note that you do not need a second argument whatsoever to perform this request:

import axios from "axios";

import React from "react";

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

export default function App() {

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

React.useEffect(() => {

axios.get(`${baseURL}/1`).then((response) => {

setPost(response.data);

});

}, []);

function deletePost() {

axios

.delete(`${baseURL}/1`)

.then(() => {

alert("Post deleted!");

setPost(null)

});

}

if (!post) return "No post!"

return (

<div>

<h1>{post.title}</h1>

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

<button onClick={deletePost}>Delete Post</button>

</div>

);

}

In most cases, you do not need the data that's returned from the .delete() method.

But in the code above, the .then() callback is still used to ensure that your request is successfully resolved.

In the code above, after a post is deleted, the user is alerted that it was deleted successfully. Then, the post data is cleared out of the state by setting it to its initial value of null.

Also, once a post is deleted, the text "No post" is shown immediately after the alert message.

## How to Handle Errors with Axios

What about handling errors with Axios?

What if there's an error while making a request? For example, you might pass along the wrong data, make a request to the wrong endpoint, or have a network error.

To simulate an error, you'll send a request to an API endpoint that doesn't exist: /posts/asdf.

This request will return a 404 status code:

import axios from "axios";

import React from "react";

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

export default function App() {

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

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

React.useEffect(() => {

// invalid url will trigger an 404 error

axios.get(`${baseURL}/asdf`).then((response) => {

setPost(response.data);

}).catch(error => {

setError(error);

});

}, []);

if (error) return `Error: ${error.message}`;

if (!post) return "No post!"

return (

<div>

<h1>{post.title}</h1>

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

</div>

);

}

In this case, instead of executing the .then() callback, Axios will throw an error and run the .catch() callback function.

