Yes, there are several methods to get data from APIs in web development using JavaScript. Some of the commonly used methods include Axios, AJAX (using XMLHttpRequest), and the Fetch API. Here's a brief overview of each:

1. \*\*Axios\*\*:

Axios is a popular JavaScript library for making HTTP requests from the browser. It provides a simple and consistent API for performing asynchronous operations. You need to include Axios in your project, and you can make GET, POST, PUT, DELETE, and other HTTP requests easily.

Example of making a GET request with Axios:

```javascript

axios.get('https://api.example.com/data')

.then(function (response) {

// Handle the response data here

console.log(response.data);

})

.catch(function (error) {

// Handle any errors here

console.log(error);

});

```

2. \*\*AJAX (XMLHttpRequest)\*\*:

The XMLHttpRequest object is the traditional way to make asynchronous HTTP requests in JavaScript. While it's not as user-friendly as modern libraries like Axios or Fetch, it's still widely used.

Example of making a GET request with AJAX:

```javascript

var xhr = new XMLHttpRequest();

xhr.open('GET', 'https://api.example.com/data', true);

xhr.onload = function () {

if (xhr.status === 200) {

// Handle the response data here

console.log(xhr.responseText);

} else {

// Handle errors

console.log('Request failed');

}

};

xhr.send();

```

3. \*\*Fetch API\*\*:

The Fetch API is a modern alternative to XMLHttpRequest and provides a more straightforward and promise-based interface for making network requests. It is built into modern browsers and doesn't require additional libraries.

Example of making a GET request with the Fetch API:

```javascript

fetch('https://api.example.com/data')

.then(function (response) {

if (response.ok) {

return response.json(); // Parse the response as JSON

}

throw new Error('Network response was not ok');

})

.then(function (data) {

// Handle the parsed data here

console.log(data);

})

.catch(function (error) {

// Handle any errors here

console.log(error);

});

```

To access the JSON API using Axios, Fetch API, or AJAX, you can follow the examples you provided. I'll provide you with code examples for each method using the URL you specified:

1. \*\*Using Axios\*\*:

```javascript

const axios = require('axios'); // If you're using Node.js

// If you're using a browser, include Axios in your HTML

axios.get(

`https://api.trace.moe/search?url=${encodeURIComponent(

"https://images.plurk.com/32B15UXxymfSMwKGTObY5e.jpg"

)}`

)

.then(function (response) {

// Handle the response data here

const jsonData = response.data;

console.log(jsonData);

})

.catch(function (error) {

// Handle any errors here

console.log(error);

});

```

2. \*\*Using Fetch API\*\*:

```javascript

fetch(`https://api.trace.moe/search?url=${encodeURIComponent("https://images.plurk.com/32B15UXxymfSMwKGTObY5e.jpg")}`)

.then(function (response) {

if (response.ok) {

return response.json();

}

throw new Error('Network response was not ok');

})

.then(function (data) {

// Handle the parsed JSON data here

console.log(data);

})

.catch(function (error) {

// Handle any errors here

console.log(error);

});

```

3. \*\*Using AJAX (XMLHttpRequest)\*\*:

```javascript

var xhr = new XMLHttpRequest();

xhr.open('GET', `https://api.trace.moe/search?url=${encodeURIComponent("https://images.plurk.com/32B15UXxymfSMwKGTObY5e.jpg")}`, true);

xhr.onload = function () {

if (xhr.status === 200) {

try {

var jsonData = JSON.parse(xhr.responseText);

// Handle the parsed JSON data here

console.log(jsonData);

} catch (error) {

// Handle JSON parsing error

console.log(error);

}

} else {

// Handle errors

console.log('Request failed');

}

};

xhr.send();

```

In all three examples, you are making a GET request to the specified URL, and the response is expected to be in JSON format. You can then handle the JSON data as needed in your JavaScript code.  
  
To display the JSON data in an HTML document and access it via JavaScript, you can follow these steps:

1. \*\*HTML Structure\*\*: Create an HTML structure where you want to display the JSON data.

```html

<!DOCTYPE html>

<html>

<head>

<title>Display JSON Data</title>

</head>

<body>

<div id="jsonDataDisplay">

<!-- JSON data will be displayed here -->

</div>

<button id="fetchDataButton">Fetch Data</button>

<script src="script.js"></script>

</body>

</html>

```

In this HTML code, we have a `<div>` element with the ID "jsonDataDisplay" where we will display the JSON data, and a "Fetch Data" button with the ID "fetchDataButton" to trigger the API request.

2. \*\*JavaScript (script.js)\*\*: Create a JavaScript file to fetch the data and update the HTML content when the "Fetch Data" button is clicked.

```javascript

// script.js

document.getElementById("fetchDataButton").addEventListener("click", function() {

// Fetch data from the API using Fetch API

fetch(`https://api.trace.moe/search?url=${encodeURIComponent("https://images.plurk.com/32B15UXxymfSMwKGTObY5e.jpg")}`)

.then(function(response) {

if (response.ok) {

return response.json();

}

throw new Error("Network response was not ok");

})

.then(function(data) {

// Display the JSON data in the specified <div>

const jsonDataDisplay = document.getElementById("jsonDataDisplay");

jsonDataDisplay.innerHTML = JSON.stringify(data, null, 2); // Format JSON for display

})

.catch(function(error) {

console.log(error);

});

});

```

In this JavaScript code, we add a click event listener to the "Fetch Data" button. When the button is clicked, it makes an API request, retrieves the JSON data, and then updates the content of the "jsonDataDisplay" `<div>` with the JSON data. We use `JSON.stringify(data, null, 2)` to format the JSON for a more readable display.

3. \*\*Accessing the Page\*\*: To access this HTML page and run the JavaScript, you can simply open the HTML file in your web browser. Make sure the HTML file and the "script.js" file are in the same directory.

4. \*\*Running the Code\*\*: Open the HTML file in a web browser and click the "Fetch Data" button to trigger the API request and display the JSON response.

This setup will allow you to fetch data from the API and display it in your HTML page when the button is clicked.